ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra řídící techniky

Prostředí pro modelování a optimalizaci provozu kogeneračních provozů

Environment for modelling and optimization of combined heat and power plant operations

Bakalářská práce

Vedoucí práce: Ing. Michal Dvořák

Jan Prášek

Praha 2013

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Prášek**

Studijní program: Kzbernetika a robotika
Obor: Systémy a řízení

Název tématu: **Prostředí pro modelovaní a optimalizaci provozu kogeneračních provozů**

Pokyny pro vypracování:

1. Najděte vhodné existující grafické prostředí, pomocí kterého bude možné vytvořit schematický popis parního cyklu teplárny. Vlastnosti tohoto schematického popisu jsou dány jeho účelem, kterým je formulace optimalizačního problému pro plánování provozu a obchodu teplárny, viz (Dvořák & Havel, 2012).
2. Navrhněte způsob, jak uživatelsky přívětivě připojit ke schematickému popisu teplárny vstupní data pro optimalizaci, tj. popisy produktů elektřiny, časové řady s predikcí odběru tepla, cenou paliva atp.
3. Navrhněte a v prostředí .NET naprogramujte překladač vstupních dat, který převede zadání problému do formy zpracovatelné externím solverem pro řešení smíšených celočíselných optimalizačních problémů. Využijte pro to nástroj JD, viz (Hák, 2012).
4. Proveďte napojení všech komponent nástroje a výsledek otestujte na jednoduchém příkladu, který dodá vedoucí práce.
5. Součástí bakalářské práce bude dokumentace a základní návod.

Seznam odborné literatury:

[1] Dvořák, M., & Havel, P. (2012). Combined heat and power production planning under liberalized market conditions. Applied Thermal Engineering, stránky 163-173.
[2[Hák, J. (2012). Návrh SW platformy pro formulaci a řešení optimalizačních úloh v energetice. Diplomová práce.
[3] Schamai, W. (2009). Modelica Modeling Language (ModelicaML) : A UML Profile for Modelica. Linköping University Electronic Press.

Vedoucí: Ing. Michal Dvořák

Platnost zadání: do konce zimního semestru 2012/2013

L.S.

Prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 7. 11. 2012

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, dne 2. ledna 2013 .....

podpis

Poděkování

Chtěl bych poděkovat především panu Ing. Michalu Dvořákovi za jeho trpělivost a skvělé vedení. Dále děkuji své rodině a přátelům za významnou podporu během celých studií.

Abstrakt

Cílem práce je vytvoření uživatelsky přívětivého prostředí pro modelování a optimalizaci provozu kogeneračních tepláren, neboť současná metodika je neumožňuje rychle a přehledně vytvářet nebo měnit. Práce se zabývá výběrem vhodného existujícího grafického prostředí pro jednoduchou tvorbu schématického popisu, kde je hlavní důraz kladen na akademickou i komerční dostupnost a použitelnost v problematice, a implementací programu Uml2Solver. Jako nejvhodnější grafické prostředí se ukazuje prostředí Visual Paradigm for UML, zejména vzhledem ke svým možnostem, dostupnosti a plné podpory UML. Uml2Solver kombinuje vstupní data, která jsou zadávána pomocí souboru v Microsoft Excel, a schématický popis systému k formulaci optimalizačního problému, který je následně řešen solverem Gurobi. Program je napsán s využitím frameworku JD a platformy .NET.

Klíčová slova: kogenerační provoz, modelovací prostředí, optimalizace

Abstract

This work aims to create a user-friendly environment for modelling and optimization of combined heat and power plant operations, because current methodology does not allow user to create or alter models effectively. Firstly, the work deals with finding suitable graphical environment which would provide a simple way of model scheme creation and fine academical and commercial availability. The most suitable environment apeared to be Visual Paradigm for UML which satisfied all demands. Second part of the work was implementation of program Uml2Solver. Uml2Solver combines all input data – scheme created in Visual Paradigm and input time vector with required parameters stored in MS Excel file, to create and to solve an optimization problem. This problem is created using JD framework and solved by Gurobi solver. The program was implemented using C# .NET framework.

Keywords: combined heat and power plant, modelling environment, optimalization

Obsah

1. Úvod

Cílem práce je vytvoření kompaktního řešení pro modelování provozu kogeneračních provozů. Problematiku lze rozdělit do několika částí, které jsou ve výsledném nástroji zkombinovány:

- Vytvoření modelu provozu v grafickém prostředí
- Export z grafického popisu do univerzálního formátu – XML.
- Vytvoření souboru se vstupními daty.
- Zpracování navrženého modelu a vstupních dat.
- Spuštění solveru Gurobi, výpis a uložení výsledků.

Programová část využívá framework JD, který byl navržen v rámci diplomové práce Ing. Josefem Hákem [1] k univerzálnímu formulaci optimalizačních problémů bez ohledu na použitý solver. Ze závěrů této práce vyplývá, že solver Gurobi [16] je v kombinaci s JD velmi efektivní jak z hlediska výpočetního času, tak i potřebné paměti. Proto jej budu používat pro řešení optimalizačních úloh.

Jednotlivým etapám řešení se budu postupně věnovat v následujících kapitolách.

1.1 Motivace a cíle práce

Hlavní motivací k tvorbě tohoto nástroje je absence uživatelsky přívětivého rozhraní pro zadávání optimalizačních úloh kogeneračních provozů a zobrazení jejich výsledků.

Optimalizační úlohou je hledání maximálního nebo minimálního řešení objektivní funkce., jejíž proměnné podléhají omezujícím podmínkám. V případě optimalizace kogeneračních provozů je touto funkcí celkový zisk provozu, který se snažíme maximalizovat [2].

V současné době v rámci naší problematiky se optimalizační úlohy formulují zápisem kódu v prostředí .NET. Tento přístup má zásadní nevýhodu:

- Nutnost vyvíjet vždy novou optimalizační aplikaci pro zadání různých provozů, které se mohou jen velmi málo lišit.

Mezi výhody lze naopak počítat vyšší flexibilitu, kterou lze uplatnit zejména u nestandardních zadání.

Cílem je tedy navrhnout nástroj, který by vytvořil většinu optimalizační aplikace automaticky s tím, že pouze velice specifické části by musel uživatel dodělat ručně. Použití grafického prostředí umožní rychlé a přehledné vytvoření modelu tedy tzv. rapid prototyping.

Dílčím cílem bylo vyvinout nástroj tak, aby bylo možné jej v budoucnu rozšířit o další funkce nebo komponenty popř., aby bylo možné použít program na zpracování schémat pro možné budoucí dedikované grafické rozhraní.

2. Grafické prostředí

Jednu z hlavních priorit při volbě grafického prostředí je výstupní formát dat. Je třeba zajistit určitou kompatibilitu s dalšími nástroji a službami, která je pro další vývoj či rozšíření navrženého rozhraní esenciální. Pro tento účel je jednoznačně vyhovující standard OMG Unified Modeling Language (UML) [3]. Důležitým aspektem, který hovoří právě pro použití UML, je možnost generace XML souborů, protože přímo v .NET [6] existují knihovny pro jednoduchou práci s těmito soubory. V neposlední řadě je třeba také uvést, že grafické UML editory se těší poměrně slušné pozornosti a existuje jich velké množství. Standard se také dále aktivně vyvíjí, poslední verze 2.5 je z října 2012 [3].

Pro úplnost zde také uvedu další ze standardů – SysML (Systems Modeling Language). SysML vychází právě z dříve vzniklého UML a částečně jej rozšiřuje, ačkoliv zdaleka nepokrývá veškeré možnosti UML – například neumožňuje zápis Object diagramů. Některé z programů jsou také schopny vytvářet modely v SysML, ovšem ne všechny [4]. Z těchto důvodů jsem vybral UML jako použitý standard.

To, co mají všechny grafické UML editory společné, je zápis objektů (komponent) do tříd, které popisují jejich parametry a metody. Výstupem pak jsou tzv. UML diagramy, které znázorňují relace mezi těmito objekty. Typů UML diagramů je celá řada, nejpoužívanějším je pravděpodobně class diagram, který obsahuje seznam objektů (tříd) a definuje vztahy mezi nimi – dědičnost, asociace apod. Pro můj nástroj je ovšem stěžejní jiný typ diagramu a tím je object diagram. Do object diagramu se vkládají instance objektů definovaných v class diagramu, mezi kterými lze pak vytvářet asociace, tedy propojení.

Protože většina vhodných programů pro levné řešení spadá do kategorie open-source projektů, je třeba počítat s nekompletní implementací standardu UML. Hlavní kritéria pro výběr grafického prostředí pro tvorbu UML diagramů jsou:

- Schopnost vytvářet object diagramy
- Možnost exportu do XML
- Podpora nejnovější verze UML (UML 2.x)
- Uživatelsky intuitivní a jednoduché prostředí
- Tendence budoucího vývoje
- Licenční dostupnost

2.1 Dostupné nástroje

V následujících podkapitolách se podrobněji zbývám některými ze zvažovaných programů.

2.1.1. OpenModelica

Objektově orientovaný open-source modelovací nástroj OpenModelica [5] zde uvádím jako jediného zástupce, který nefunguje na bázi UML. Nepracuje se zde tedy s diagramy ale přímo s modely určenými svými vlastnostmi a rovnicemi, které lze přirovnat k objektům (třídám) v UML. V jednoduchém grafickém rozhraní se dobře pracuje s vytvořenými objekty. Schopnosti

tohoto nástroje pro moje potřeby jsou více než dostatečné, dokonce oproti ostatním zkoumaným programům jsou zde jednoduše řešeny propojení mezi komponentami.

Jedinou překážkou pro použití OpenModelici je formát, ve kterém OM ukládá svá data (**.mo*). Nelze totiž vytvořit kýžený XML soubor popisující vytvořený model, což značně komplikuje jeho načítání pomocí .NET. Z tohoto důvodu není OpenModelica vyhovující, což je škoda, neboť nejen vzhledem ke schopnostem svého grafického editoru by byla optimálním prostředím.

2.1.2. StarUML

Dalším open-source editorem, tentokrát již s podporou UML 2.0 je StarUML [6]. StarUML umožňuje export dat do XMI, což je standard vystavěný na standardu XML a tedy čitelný pro parse v .NET. Nevýhodou je nepravděpodobný další vývoj (poslední verze je z roku 2005) a nemožnost vytvářet object diagramy.

2.1.3. ArgoUML

ArgoUML [8] je rovněž open-source editor, vystavěný na bázi Javy. Je schopen export do XMI, ovšem podporuje pouze starší standard UML 1.4. Ačkoliv poslední zveřejněný build je z roku 2011, další vývoj (z ohledem na podporu staršího standardu) je nejistý. Také nepodporuje object diagram.

2.1.4. Papyrus

Papyrus [9] je jediným nesamostatným prostředím, které zde rozebírám. Lze jej použít výhradně jako rozšíření do rozšířeného open-source prostředí Eclipse [10]. Tento software je stále v aktivním vývoji, jehož poslední verze je ze září 2012 s podporuje UML 2.2, dá se tedy očekávat kontinuální podpora. Výstupním formátem jsou **.uml* soubory na bázi XML. Nevyhovuje pouze v jediném bodě – zatím neumí pracovat s object diagramy.

Mimo UML podporuje Papyrus také SysML a ModelicaML [11]. ModelicaML je rozšíření standardu UML, které zaručuje dopřednou kompatibilitu s OpenModelicou (tedy export do OM souborů).

Papyrus je z mého podhledu nejsilnějším nástrojem z palety open-source UML editorů, které jsou v tuto chvíli na trhu.

2.1.5. Umlet

Tento zdarma dostupný open-source program je nejjednodušším UML prostředím, které zde uvádím [12]. Naprogramován na bázi javy, podporuje „svou“ verzi UML (není uvedeno), ovšem zvládá object diagramy a umožňuje vlastní definici UML bločků. Jedinou a zásadní nevýhodou je formát generovaného XML výstupu, neboť Umlet byl primárně navržen na rychlé navržení diagramů a jejich export do obrázků. Data jsou všechna zapsána v jednom tagu, bylo by tedy zapotřebí vytvořit vlastní parser, což by mohlo způsobit zpomalení načítání nebo vnést do programu zbytečnou chybovost.

2.1.6. Visual Paradigm for UML

Zvoleným grafickým prostředím pro výsledný nástroj je Visual Paradigm for UML [13]. Nejedná se o open-source program, přesto existuje verze Community Edition, která je zdarma. Pro komerční účely je ovšem nutné zakoupit placenou verzi. Pro potřeby mého řešení stačí

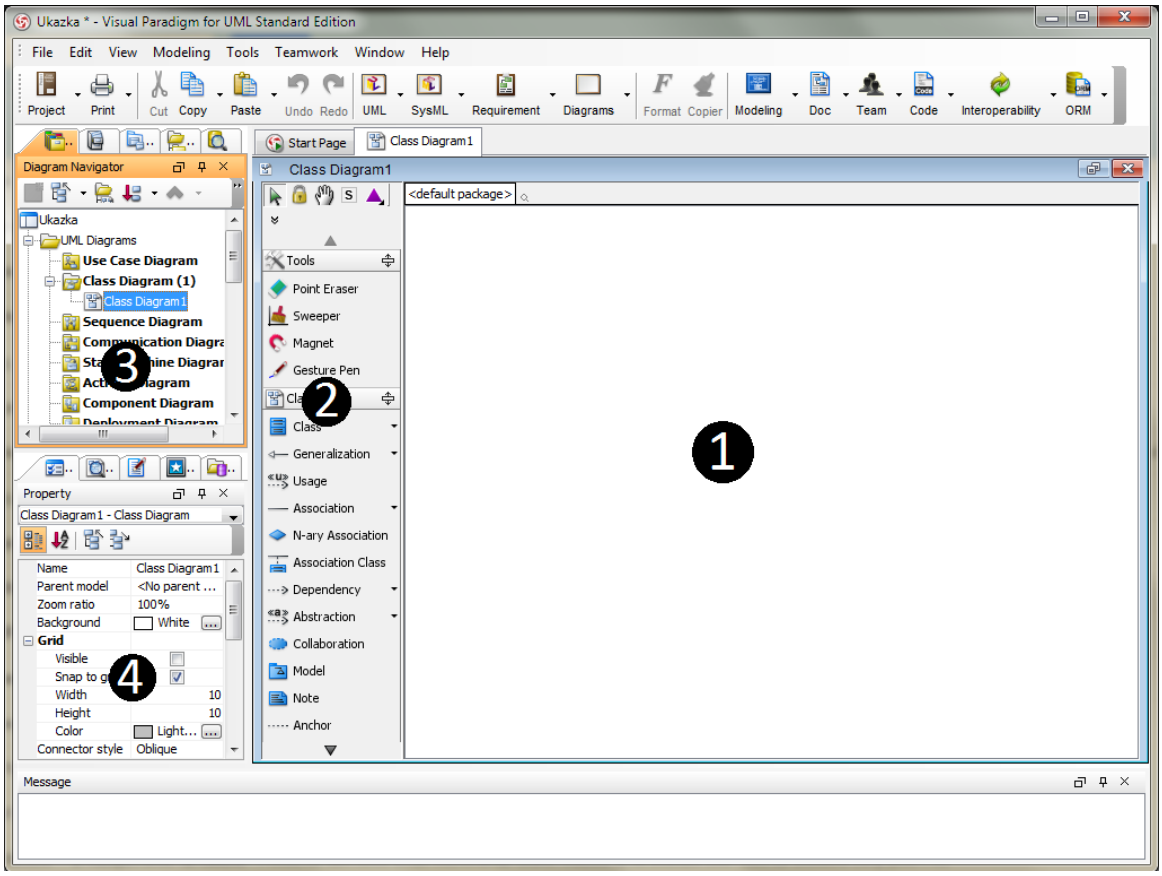
vlastnit buď verzi Community Edition nebo nejnižší komerční verzi Modeler, kterou lze pořídit za $99. Pro akademické účely vlastnilo ČVUT v době psaní této bakalářské práce akademickou licenci.

Visual Paradigm plně implementuje standard UML 2.4, tedy všechny UML diagramy, včetně object diagramu. Poslední verze 10 je z června 2012, lze tedy předpokládat další vývoj a podporu.

Výstupní formáty jsou XML, zde je možnost výběru mezi tradiční verzí a zjednodušenou, a dále také XMI.

Grafické rozhraní programu je patrné z obr. 2.1 a lze jej rozdělit na 4 hlavní části:

1) Pracovní okno pro editaci diagramů
2) Paleta nástrojů
3) Navigátor projektu (záložkou lze přepínat mezi různými zobrazeními)
4) Okno pro úpravu vlastností označeného objektu



Obr. 2.1: Visual Paradigm GUI.

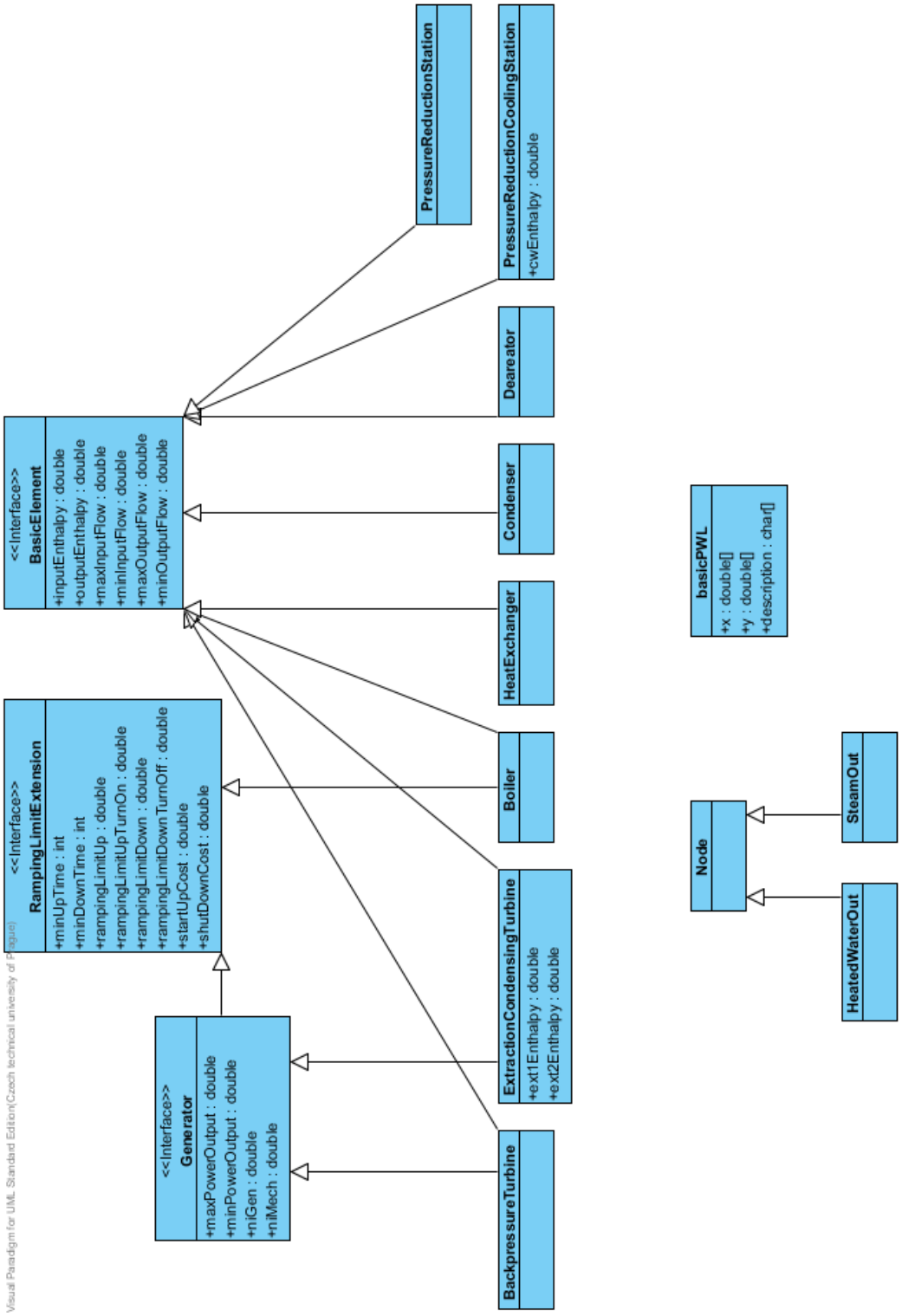
2.2 Tvorba knihovny ve Visual Paradigm

Pro potřeby výsledného nástroje je nutné vytvořit vlastní knihovnu s komponenty ve formě class diagramu, ze které lze sestavit model drtivé většiny kogeneračních provozů.

Tento class diagram obsahuje nejen veškeré výsledné komponenty, ale také jejich pomocné (mnohdy společné) předky pro snadnou orientaci a editaci. Jednotlivé komponenty

jsou zde reprezentovány jako samostatné třídy, které obsahují vlastní i zděděné parametry. Ve standardním class diagramu bývá také uváděno jaké metody jednotlivé třídy implementují, ovšem pro potřeby mého řešení nejsou metody podstatné, proto je neuvádím. Výhodou objektové struktury s dědičností je možnost jednoduchého rozšíření knihovny o další komponenty.

Knihovna ve formě class diagramu je na obr. 2.2.



Obr. 2.2: Navržená knihovna komponent

2.2.1. Komponenty knihovny

Třídy komponent lze rozdělit do dvou skupin - fyzické a pomocné. Fyzické třídy přímo odrážejí skutečné části teplárny – např. Boiler nebo BackpressureTurbine, zatímco
pomocné třídy jako např. Node slouží ke zpřehlednění schématu popř. předávají fyzickým komponentám další vlastnosti (např. PWL aproximaci parametru). Veškeré zadávané parametry mohou nabývat pouze nezáporných hodnot.

V následujících podkapitolách uvedu jednotlivé implementované třídy a rozhraní (interface) s jejich parametry (dle požadavků jednotlivých komponent, viz [2]).

2.2.1.1. BasicElement

Tento objekt není ve skutečnosti třídou, ale rozhraním (interface), což znamená, že z něj nelze vytvářet instance. Slouží tedy pouze jako předek pro své potomky.

Pouhou deklarací objektu jako interface, není zajištěna ochrana proti vytváření instancí – prakticky je ve VP možné tyto instance vytvořit, to ovšem vyústí v následné chybné načtení schématu.

BasicElement je základním zdrojem parametrů všech fyzických komponent (tříd). Obsahuje tyto parametry s datovými typy (v hranatých závorkách je uvedena jednotka – vzorkování času je po jedné hodině):

- *inputEnthalpy [kJ/kg] : double* – vstupní entalpie
- *outputEnthalpy [kJ/kg]: double* – výstupní entalpie
- *maxInputFlow [t/h] : double* – maximální vstupní hmotnostní průtok
- *minInputFlow [t/h] : double* – minimální vstupní hmotnostní průtok
- *maxOutputFlow [t/h] : double* – maximální výstupní hmotnostní průtok
- *minOutputFlow [t/h] : double* – minimální výstupní hmotnostní průtok

2.2.1.2. RampingLimitExtension

Interface, který přidává parametry pro omezení náběhu komponenty:

- *minUpTime [h] : int* – minimální počet časových úseků, po které musí komponenta běžet než je možné ji vypnout
- *minDownTime [h] : int* - minimální počet časových úseků, po které musí být komponenta vypnuta než je možné ji znovu zapnout
- *rampingLimitUp [t/h] : double* – maximální růst hmotnostního průtoku mezi dvěma přilehlými časovými úseky.
- *rampingLimitDown [t/h] : double* – maximální pokles hmotnostního průtoku mezi dvěma přilehlými časovými úseky (kladné číslo).
- *rampingLimitUpTurnOn [t/h] : double* – maximální růst hmotnostního průtoku mezi dvěma přilehlými časovými úseky při zapnutí komponenty.
- *rampingLimitDownTurnOff [t/h] : double* – maximální pokles hmotnostního průtoku mezi dvěma přilehlými časovými úseky (kladné číslo) při vypnutí komponenty
- *startUpCost [-] : double* – cena za zapnutí příslušné komponenty
- *shutDownCost [-] : double* – cena za vypnutí příslušné komponenty

2.2.1.3. Generator

Interface s parametry pro turbíny, který dále dědí z rozhraní RampingLimitExtension. Definuje tyto parametry:

- *maxPowerOutput [MW]: double* – maximální hodnota výstupní elektrické energie
- *minPowerOutput [MW]: double* – minimální hodnota výstupní elektrické energie
- *niGen [-] : double* – generátorové ztráty
- *niMech [-] : double* – mechanické ztráty

2.2.1.4. BackpressureTurbine

Třída, která dědí parametry z rozhraní BasicElement a Generator. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.5. ExtractionCondensingTurbine

Třída, která dědí parametry z rozhraní BasicElement a Generator. Ve schématu očekává jeden vstup a tři výstupy. Dále definuje následující parametry:

- *ext1Enthalpy [kJ/kg] : double* – výstupní entalpie prvního odběru
- *ext2Enthalpy [kJ/kg] : double* – výstupní entalpie druhého odběru

2.2.1.6. Boiler

Dědí parametry z rozhraní BasicElement a RampingLimitExtension. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.7. HeatExchanger

Dědí parametry z rozhraní BasicElement. Ve schématu očekává dva vstupy a dva výstupy.

2.2.1.8. Condenser

Dědí parametry z rozhraní BasicElement. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.9. Deareator

Dědí parametry z rozhraní BasicElement. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.10. PressureReductionStation

Dědí parametry z rozhraní BasicElement. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.11. PressureReductionCoolingStation

Dědí parametry z rozhraní BasicElement. Ve schématu očekává dva vstupy a jeden výstup a definuje následující parametr:

- *cwEnthalpy [kJ/kg] : double* – vstupní entalpie chladící vody

2.2.1.12. Node

Pomocná třída, která funguje jako místo stejného energetického stavu. Nemění tedy vlastnosti vstupní vody nebo páry. Může z ní i do ní vést libovolný počet spojení.

2.2.1.13. HeatedWaterOut

Dědí z třídy Node – tato třída je pouze k označení místa pro výstupní vyhřívanou vodu. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.14. SteamOut

Dědí z třídy Node – tato třída je pouze k označení místa pro výstupní vyhřívanou páru. Ve schématu očekává jeden vstup a jeden výstup.

2.2.1.15. BasicPWL

Třída, která modeluje PWL aproximaci. Definuje tyto parametry:

- *description : char[]* – popis pro určení cílových vázaných veličin (notace v kapitole 3.3.1.2)
- *x : double[]* - x-ové hodnoty PWL aproximace
- *y : double[]* - y-ové hodnoty PWL aproximace

3. Uml2Solver

Program byl vytvořen v IDE (Integrated Development Environmet) Microsoft Visual C# Studio 2012 Express [15].**Error! Reference source not found.** pomocí frameworku .NET 4.5. Mimo standardní knihovny jsou v projektu použity knihovny *System.Xml* k parsování XML a *System.Data.OleDb* pro čtení **.xls(x)* souboru. Knihovna *System.Data.OleDb* vyžaduje pro svůj běh nainstalovaný volně dostupný databázový stroj Microsoft AccessDatabaseEngine 2007 [14] (přiložen na CD). Další nutnou součástí je instalace ovladačů pro Gurobi solver [16].

Program vyžaduje pro svůj chod operační systém Windows Vista a novější. Jsou přiloženy verze pro 32-bit i 64-bit OS (viz přiložené CD).

Dokumentace byla vytvořena pomocí volně dostupného programu Doxygen [17] (viz přiložené CD).

3.1 Vstupní data

Program Uml2Solver vyžaduje pro svůj běh dva zdroje vstupních dat. Prvním je xml soubor vygenerovaný v prostředí Visual Paradigm for UML s popisem systému a parametry jeho komponent. Druhým je pak soubor v Excelu, který obsahuje vstupní časové řady predikovaných veličin (elektrických či tepelných výkonů), očekávané ceny paliv a elektřiny a dalších, kvantovaných po jednotkách času (zpravidla po t = 1h).

3.1.1. Struktura Excel souboru

Prvotní myšlenkou bylo vstupní data zachytit také v grafickém prostředí Visual Paradigm tak, aby byl výstupem pouze jediný xml soubor. Tento přístup se ovšem ve výsledku ukázal jako zbytečně složitý a nepřehledný, zejména vzhledem k variabilní délce časové řady – uživatel by musel zadávat pole čísel dlouhá i několik desítek pozic. Proto jsem nakonec zvolil původní koncept zadávání vstupních dat pomocí tabulek v Excelu. Výhodou tohoto řešení je přehlednost a snadná editace zejména dlouhých časových řad. Přístup k datům z tabulek v prostředí .NET je rovněž usnadněn zabudovanými knihovnami. Pokud by se v budoucnu přistoupilo k tvorbě dedikovaného grafického rozhraní, bylo by možné všechna data souhrnně spravovat právě tam.

Vstupní data jsou rozdělena do dvou listů. V prvním jsou uvedeny vstupní parametry s predikcí dodávek tepla, energie a odhady cen pro celý časový vektor. Zde je třeba zachovat pořadí parametrů v hlavičce (viz obr. 3.1). Tento princip by v budoucnu šel dále zdokonalit tak, aby zahrnul širší spektrum typů vstupních dat. Za povinnými položkami následuje popis dostupných energetických balíčků. Při zachování příslušného formátu balíčku jich může následovat libovolný počet (viz obr. 3.2).

Ve druhém listu jsou uvedeny možné stavy jednotlivých komponent. Hodnoty v této tabulce mohou nabývat tří hodnot:

- 0 : Komponenta je v daný časový úsek vypnuta.
- 1 : Komponenta je v daný časový úsek zapnuta.
- ostatní : solver má možnost mít komponentu zapnutou nebo vypnutou.

V tomto listě mohou být záznamy komponent v libovolném pořadí, musí však existovat ve schématu (viz obr. 3.3).

	A	B	C	D	E	F	G	H	I
1		PREDIKCE DODÁVKY TEPLA						REALIZOVANÉ PRODUK	
2		Voda		Pára		Vlastní spotřeba		Silová elektřina	
3		Dodávka	Cena	Dodávka	Cena	Voda	Pára	Výkon	P_cena
4	DATUM A ČAS	[GJ]	[Kč/GJ]	[GJ]	[Kč/GJ]	[GJ]	[GJ]	[MW]	[Kč/MWh]
5	24.3.2011 23:00	68,6	310	15,3	220	1	3	10	1000
6	25.3.2011 0:00	68,6	310	15,3	220	1	3	10	1000
7	25.3.2011 1:00	69,1	310	15,3	220	1	3	10	1000
8	25.3.2011 2:00	66,0	310	15,3	220	1	3	10	1000
9	25.3.2011 3:00	63,8	310	15,3	220	1	3	10	1000
10	25.3.2011 4:00	62,9	310	28,1	220	1	3	10	1000
11	25.3.2011 5:00	64,2	310	30,7	220	1	3	10	1000
12	25.3.2011 6:00	64,4	310	30,7	220	1	3	10	1000
13	25.3.2011 7:00	65,9	310	28,1	220	1	3	10	1000
14	25.3.2011 8:00	67,9	310	25,6	220	1	3	10	1000
15	25.3.2011 9:00	67,0	310	25,6	220	1	3	10	1000
16	25.3.2011 10:00	64,8	310	25,6	220	1	3	10	1000
17	25.3.2011 11:00	63,5	310	25,6	220	1	3	10	1000
18	25.3.2011 12:00	62,8	310	25,6	220	1	3	10	1000
19	25.3.2011 13:00	61,6	310	25,6	220	1	3	10	1000
20	25.3.2011 14:00	58,8	310	20,5	220	1	3	10	1000
21	25.3.2011 15:00	58,8	310	20,5	220	1	3	10	1000
22	25.3.2011 16:00	62,0	310	20,5	220	1	3	10	1000
23	25.3.2011 17:00	61,2	310	20,5	220	1	3	10	1000
24	25.3.2011 18:00	63,0	310	20,5	220	1	3	10	1000

	A	J	K	L	M	N	O
1		Odhadovaná cena vlastní odchylky					Cena emisních povolenek
2				Uhlí	Plyn		
3		Kladná	Záporná	Prům. cena	Prům. výhřevn.	Prům. cena	
4	DATUM A ČAS	[Kč/MWh]	[Kč/MWh]	[Kč/GJ]	[MJ/m^3]	[Kč/m^3]	[Kč/t]
5	24.3.2011 23:00	3000	300	90	0	0	370
6	25.3.2011 0:00	3000	300	90	0	0	370
7	25.3.2011 1:00	3000	300	90	0	0	370
8	25.3.2011 2:00	3000	300	90	0	0	370
9	25.3.2011 3:00	3000	300	90	0	0	370
10	25.3.2011 4:00	3000	300	90	0	0	370
11	25.3.2011 5:00	3000	300	90	0	0	370
12	25.3.2011 6:00	3000	300	90	0	0	370
13	25.3.2011 7:00	3000	300	90	0	0	370
14	25.3.2011 8:00	3000	300	90	0	0	370
15	25.3.2011 9:00	3000	300	90	0	0	370
16	25.3.2011 10:00	3000	300	90	0	0	370
17	25.3.2011 11:00	3000	300	90	0	0	370
18	25.3.2011 12:00	3000	300	90	0	0	370
19	25.3.2011 13:00	3000	300	90	0	0	370
20	25.3.2011 14:00	3000	300	90	0	0	370
21	25.3.2011 15:00	3000	300	90	0	0	370
22	25.3.2011 16:00	3000	300	90	0	0	370
23	25.3.2011 17:00	3000	300	90	0	0	370
24	25.3.2011 18:00	3000	300	90	0	0	370

Obr. 3.1: Vstupní data parametrů predikce tepla a cen v časovém vektoru – pevné pořadí.

	A	P	Q	R	S	T	U
1		Dostupné produkty silové elektřiny					
2		Prod. 1		denní	Prod. 2		denní
3		Min.Výkon	Max.Výkon	Cena	Min.Výkon	Max.Výkon	Cena
4	DATUM A ČAS	[MW]	[MW]	[Kč/MWh]	[MW]	[MW]	[Kč/MWh]
5	24.3.2011 23:00	10	15	1000	0	0	1200
6	25.3.2011 0:00	10	15	1000	0	0	1200
7	25.3.2011 1:00	10	15	1000	0	0	1200
8	25.3.2011 2:00	10	15	1000	0	0	1200
9	25.3.2011 3:00	10	15	1000	0	0	1200
10	25.3.2011 4:00	10	15	1000	0	0	1200
11	25.3.2011 5:00	10	15	1000	0	0	1200
12	25.3.2011 6:00	10	15	1000	0	0	1200
13	25.3.2011 7:00	10	15	1000	0	0	1200
14	25.3.2011 8:00	10	15	1000	2	10	1200
15	25.3.2011 9:00	10	15	1000	2	10	1200
16	25.3.2011 10:00	10	15	1000	2	10	1200
17	25.3.2011 11:00	10	15	1000	2	10	1200
18	25.3.2011 12:00	10	15	1000	2	10	1200
19	25.3.2011 13:00	10	15	1000	2	10	1200
20	25.3.2011 14:00	10	15	1000	2	10	1200
21	25.3.2011 15:00	10	15	1000	2	10	1200
22	25.3.2011 16:00	10	15	1000	2	10	1200
23	25.3.2011 17:00	10	15	1000	2	10	1200
24	25.3.2011 18:00	10	15	1000	0	0	1200

Obr. 3.2: Struktura energetických balíčků.

	A	B	C	D	E	F	G	H	I	J	K	L
1		Požadované stavy zařízení										
2		0 = nesmí být v provozu					1 = musí být v provozu					
3	Datum a čas	B1	B2	B3	TG1	TG2	PRCS1	PRCS2	PRS	HE1	HE2	HE3
4	24.3.2011 23:00	2	2	2	2	2	2	2	2	2	2	2
5	25.3.2011 0:00	2	2	2	2	2	2	2	2	2	2	2
6	25.3.2011 1:00	2	2	2	2	2	2	2	2	2	2	2
7	25.3.2011 2:00	2	2	2	2	2	2	2	2	2	2	2
8	25.3.2011 3:00	2	2	2	2	2	2	2	2	2	2	2
9	25.3.2011 4:00	2	2	2	2	2	2	2	2	2	2	2
10	25.3.2011 5:00	2	2	2	2	2	2	2	2	2	2	2
11	25.3.2011 6:00	2	2	2	2	2	2	2	2	2	2	2
12	25.3.2011 7:00	2	2	2	2	2	2	2	2	2	2	2
13	25.3.2011 8:00	2	2	2	2	2	2	2	2	2	2	2
14	25.3.2011 9:00	2	2	2	2	2	2	2	2	2	2	2
15	25.3.2011 10:00	2	2	2	2	2	2	2	2	2	2	2
16	25.3.2011 11:00	2	2	2	2	2	2	2	2	2	2	2
17	25.3.2011 12:00	2	2	2	2	2	2	2	2	2	2	2
18	25.3.2011 13:00	2	2	2	2	2	2	2	2	2	2	2
19	25.3.2011 14:00	2	2	2	2	2	2	2	2	2	2	2
20	25.3.2011 15:00	2	2	2	2	2	2	2	2	2	2	2
21	25.3.2011 16:00	2	2	2	2	2	2	2	2	2	2	2
22	25.3.2011 17:00	2	2	2	2	2	2	2	2	2	2	2
23	25.3.2011 18:00	2	2	2	2	2	2	2	2	2	2	2
24	25.3.2011 19:00	2	2	2	2	2	2	2	2	2	2	2

Obr. 3.3: Struktura listu s požadovanými stavy zařízení.

3.2 O programu

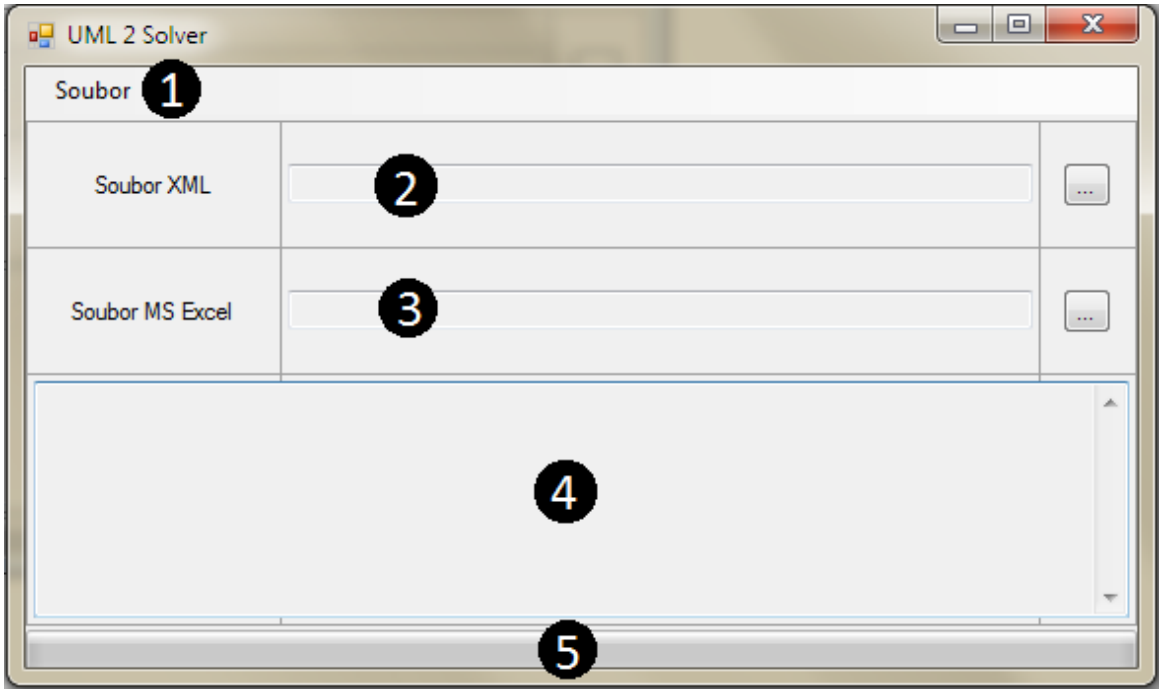
Algoritmus programu si vytvoří na základě schématu vytvořeného ve Visual Paradigm vlastní popis systému odpovídající struktury jakou má knihovna komponent (viz kapitola 2.2). Dále se vytvoří vnitřní popis propojení komponent a popis načtených vstupů včetně produktů silové elektřiny. Tento přístup zaručuje určitou modulárnost v případě budoucího rozvoje.

Po úspěšném a bezchybovém načtení všech vstupů vytvoří model pomocí frameworku JD a vloží do něj příslušná omezení. V posledním kroku zavolá skrze framework JD solver Gurobi (který se ukázal jako nejrychlejší viz [1]).

3.2.1. Grafické rozhraní (GUI)

Po spuštění aplikace se objeví hlavní okno programu, které lze rozdělit na pět částí (viz obr. 3.4):

1) Nabídka menu – Soubor :
    - Načíst data – pokusí se načíst vstupní data ze zadaných souborů **.xml* a **.xls(x)*.
    - Vyřešit – v základním stavu není aktivní, aktivuje se až po bezchybovém načtení systému. Zavolá solver.
    - Nastavení – otevře nové okno, ve kterém uživatel specifikuje názvy obou listů vstupních parametrů v Excelu.
    - Konec – ukončí program.
2) Formulář pro zadání souboru **.xml* (cestu lze zadat pouze pomocí tlačítka k procházení adresářové struktury). V otevíracím dialogu je definován souborový filtr pro formát **.xml*.
3) Formulář pro zadání Excel souboru se vstupními parametry (cestu lze zadat pouze pomocí tlačítka k procházení adresářové struktury). Podporované formáty jsou **.xlsx* (MS Excel 2007 a novější) a **.xls* (starší formát MS Excel – testováno pro formát verze MS Excel 97). Pro oba formáty jsou v otevíracím dialogu předdefinovány souborové filtry.
4) Výstupní konzole. Zobrazuje výstup z parsování obou souborů – chybové hlášky a upozornění. Při úspěšném načtení bez chyb uživatele upozorní.
5) Progress bar – je aktivní v době načítání vstupních souborů. Informuje uživatele o současném průběhu.



Obr 3.4: Uml2Solver grafické rozhraní.

3.3 Návod

V této kapitole ukáži jak s výsledným nástrojem pracovat.

3.3.1. Visual Paradigm

Pro tvorbu schémat je nutné, aby projekt obsahoval knihovnu s komponentami (class diagram, viz obr. 3.5). Schéma samotné je pak ve formě samostatného object digramu.

3.3.1.1. Vložení a propojení komponent

Pro tvorbu nového schématu je tedy třeba vytvořit nový object diagram, do něhož přetáhnutím ze zdrojového class diagramu „Components“ v levé horní části editoru (Diagram navigator – viz obr. 3.6) uživatel vloží potřebné komponenty, tedy instance předdefinovaných tříd. Po vložení komponent je třeba je propojit pomocí vztahu Association z palety nástrojů (klávesová zkratka „a“) a vyznačit u spojení kauzalitu – k tomu slouží pole „Multiplicity“, které původně určovalo vztahy mezi objekty (M:N, 0:M apod.). Do vlastností označeného spojení může uživatel vstoupit stisknutím klávesy Enter. Pro účely nástroje se tato funkce změnila a uživatel zde zadá buď hodnotu 1 pro vstup nebo 0 pro výstup. Takto je nutné vyznačit kauzalitu na obou koncích spojení. Dále je u spojení možné označit role (Role) jednotlivých komponent, což může sloužit uživateli jako informativní popisek. U MIMO (multiple input, multiple output) komponent je toto pole vyžadováno se speciální notací k označení jednotlivých vstupů/výstupů. Vlastnosti asociace jsou zobrazeny na obr. 3.7.

MIMO komponenty a notace *Role* vstupujících/vystupujících spojení (kontrola notace není case-sensitive):

- ExtractionCondesingTrubine
  - *qIn* k označení vstupu horké páry
  - *qX1* k označení výstupu první extrakce
  - *qX2* k označení výstupu druhé extrakce
  - *qOut* k označení konečného výstupu
- HeatExchanger
  - *qSteamIn* k označení vstupu horké páry
  - *qSteamOut* k označení výstupu páry, která předala svoji energii
  - *qWaterIn* k označení vstupu ohřívané vody
  - *qWaterOut* k označení výstupu ohřáté vody
- PressureReductionCoolingStation
  - *qIn* k označení vstupního toku
  - *qOut* k označení výstupního toku
  - *qCW* k označení vstupu chladící vody

Připojení PWL aproximace nevyžaduje žádné vnější speciální značení role ani označení kauzality.

3.3.1.2. Vložení parametrů

Veškeré třídy kromě těch, co jsou odvozeny od třídy Node mají předepsané parametry. K vložení hodnot je třeba nejprve definovat sloty (parametry) a záložce „Slots“ ve vlastnostech komponenty - úspěšné definování se projeví přesunutím parametru (slotu) z levé části okna do pravé (viz obr 3.8).

Ve spodní části okna s definovanými sloty lze vložit číselnou hodnotu následujícím způsobem (viz obr. 3.9):

- Označení konkrétního slotu
- „Edit values“
- „Add“ → „Text“
- Zadání čísla.

PWL aproximace obsahují pomocný parametr *description*, který je vyžadován pokud je v komponentě modelováno více PWL charakteristik. Notace označení těchto komponent je následující (kontrola notace není case-sensitive):

- Boiler
  - *Coal* označuje charakteristiku spotřeby uhlí
  - *Gas* označuje charakteristiku spotřeby zemního plynu
- ExtractionCondesingTrubine
  - *pExt1* označuje charakteristiku výkonu pro první extrakci
  - *pExt2* označuje charakteristiku výkonu pro druhou extrakci
  - *pE* označuje charakteristiku výkonu pro výstup

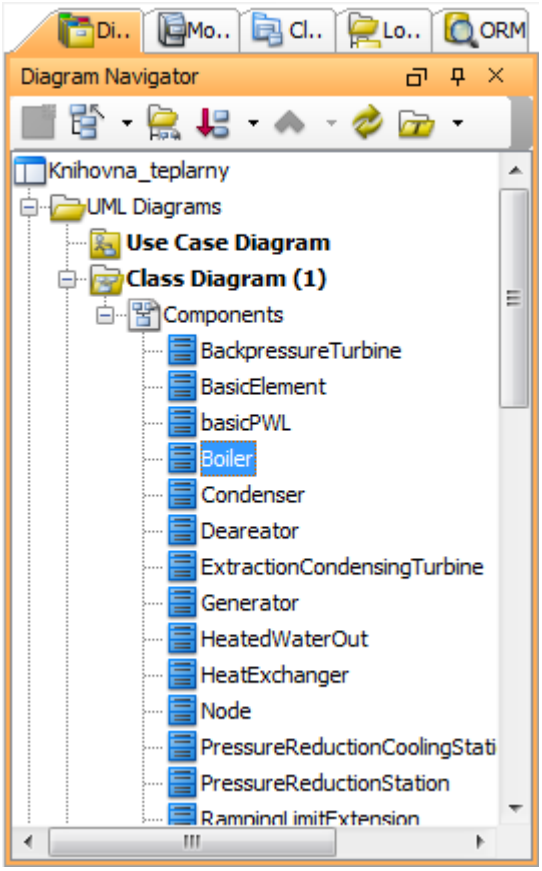
Desetinná čísla se zadávají s desetinou čárkou. Hodnoty parametru typu pole (double[]) v třídě PWL se oddělují středníkem.

Jako kontrola zadání jsou ve výsledném schématu definované sloty s hodnotami viditelné v jednotlivých bločcích instancí tříd.

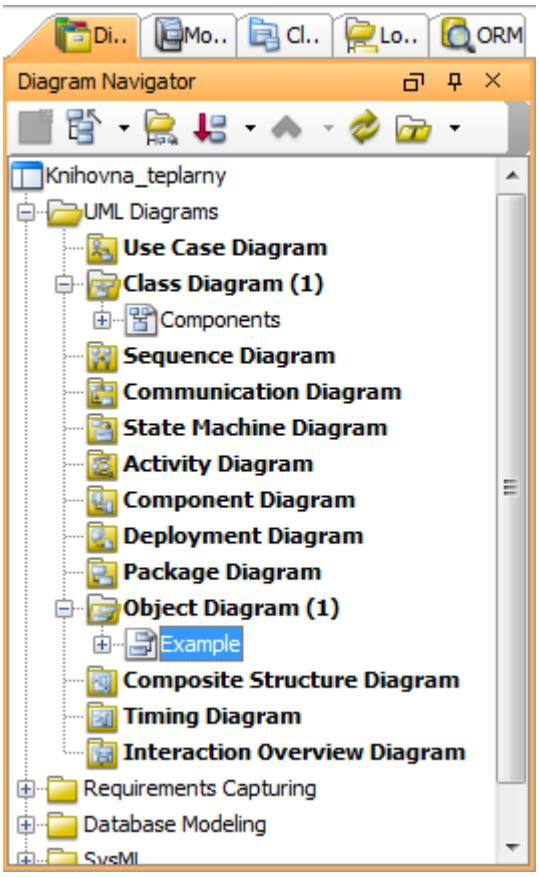
3.3.1.3. Export do Xml

K funkci exportu schématu do XML se uživatel dostane skrze menu File → Export → XML. Před samotným exportem je možné zvolit cílový adresář a přepínačem určit zda se vygeneruje XML s tradiční strukturou anebo zjednodušenou. Pro nástroj Uml2Solver je vyžadován formát zjednodušeného XML (vytváří znatelně menší soubory a šetří čas při parsování). Posledním krokem k úspěšnému exportu je zvolení zdrojového object diagramu z diagramové struktury zobrazené v dialogovém okně. Není třeba exportovat strukturu class diagramu obsahující knihovnu komponent.

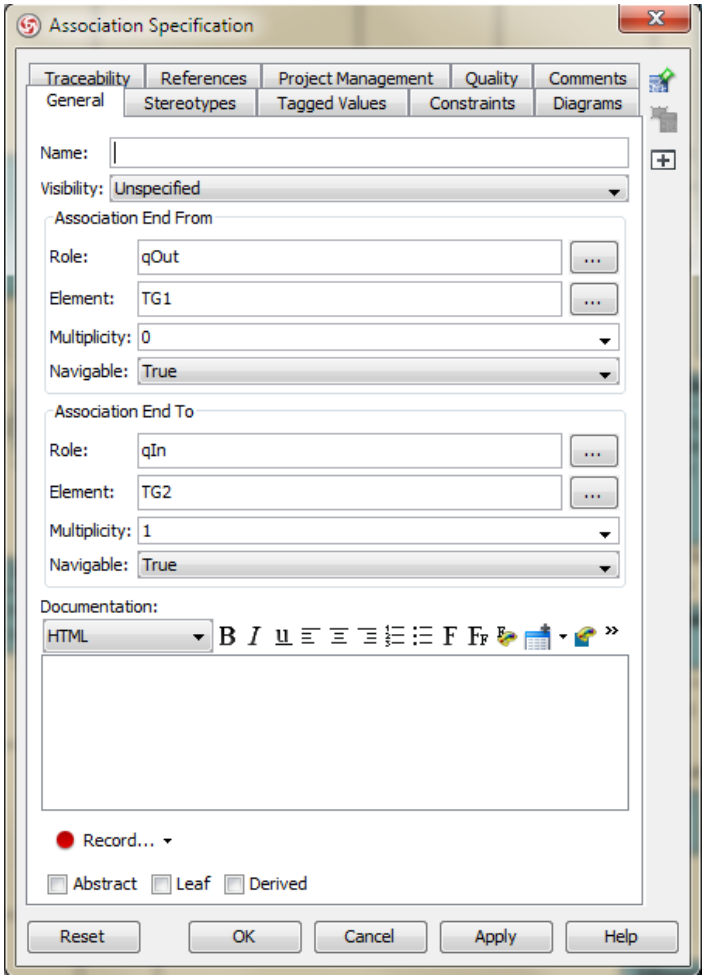
Ve výchozím stavu je v rámci exportu do XML generován také obrázek schématu ve formátu **.png*. U rozsáhlejších systému může být toto generování zabrat mnohonásobně více času než je třeba k vytvoření samostatného XML souboru. Proto jej doporučuji vypnout (není-li požadováno). Dialog exportu do XML je zachycen na obr. 3.10.



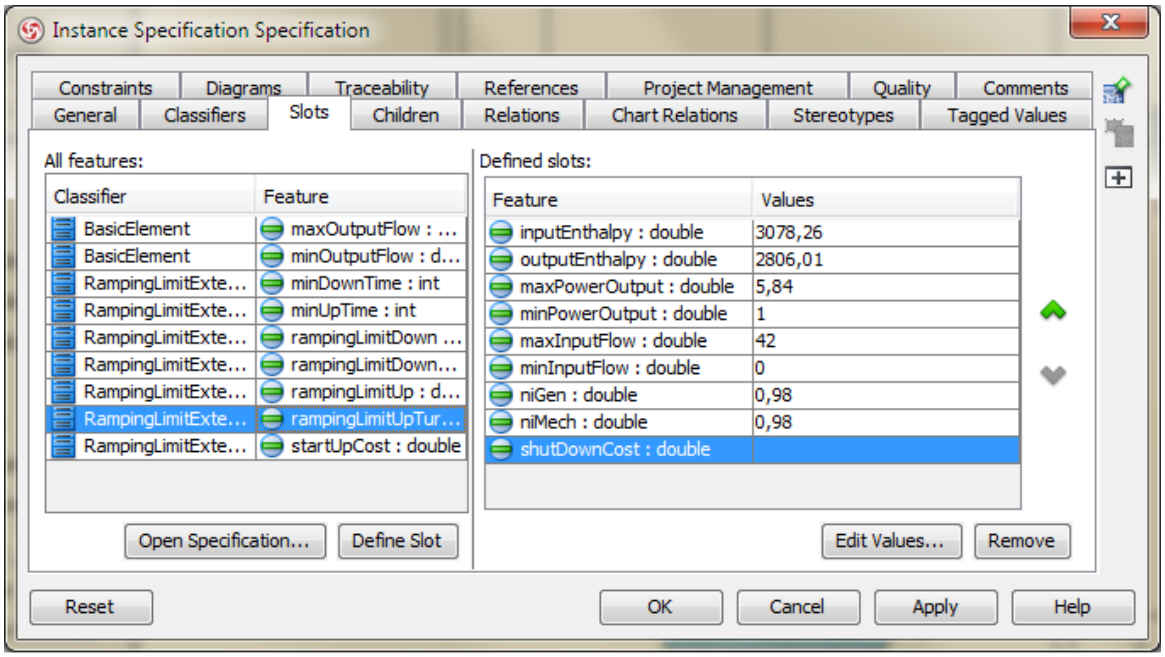
Obr. 3.5: Diagram navigator – knihovna komponent.



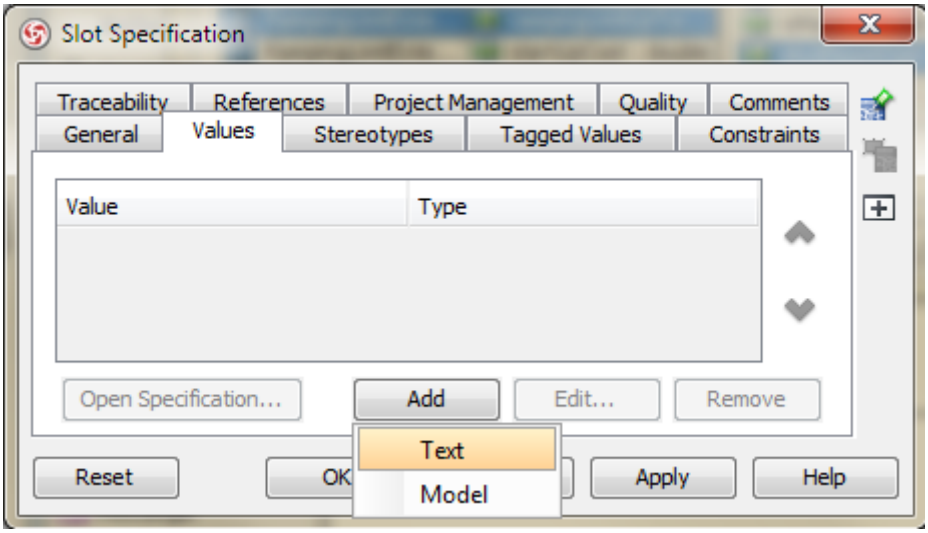
Obr. 3.6: Diagram navigator – object diagram.



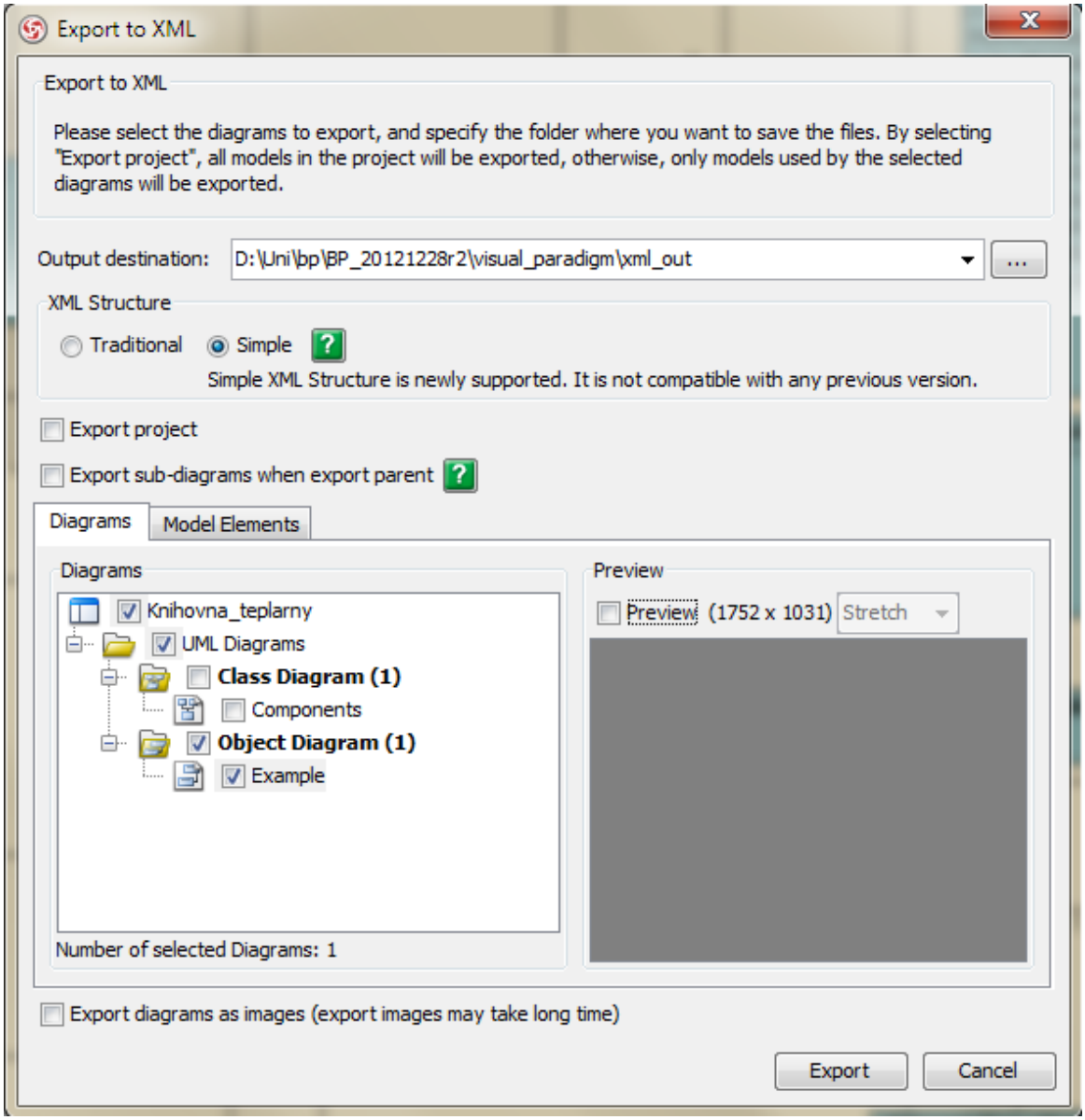
Obr. 3.7: Vlastnosti asociace (spojení komponent).



Obr. 3.8: Vlastnosti instance – definování slotu (parametru).



Obr. 3.9: Vložení hodnoty parametru.



Obr. 3.10: Dialog exportu do XML.

3.3.2. Excel

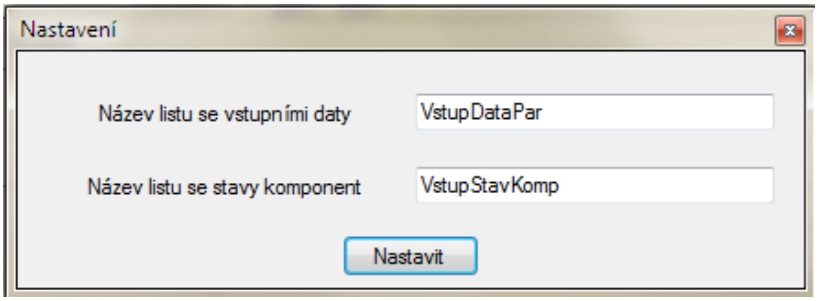
Do souboru v Excelu se do dvou listů zapíší vstupní data pro optimalizaci ve formátu popsaném v kapitole 3.1.1. Je vyžadován stejný časový vektor pro oba listy (kontroluje se délka vektoru - popis jednotlivých řádek se může lišit).

3.3.3. Program Uml2Solver

Program vyžaduje zadání obou zdrojových souborů pomocí tlačítek *(…)* k prohledávání adresářové struktury v hlavním okně aplikace. Dále je důležité zkontrolovat, aby názvy listů v Excelu odpovídaly nastavení v programu (*Soubor* → *Nastavení*) (viz obr. 3.11).

Spuštění parsování se provádí příkazem Soubor → Načíst data. Při chybovém parsování musí uživatel odstranit uvedené chyby než je možné systém optimalizovat (příklad chybového výpisu na obr. 3.12). Kromě chyb (Error) program vypisuje také varování (Warning), která jsou vypsána, pokud program nenalezne zadaní příslušných parametrů jednotlivých komponent. S varováními lze pokračovat k optimalizaci, uživatel ovšem musí mít na paměti, že všechny nezadané parametry jsou vynulovány – je pravděpodobné, že solver vrátí chybný výsledek nebo se pro něj problém stane neřešitelným.

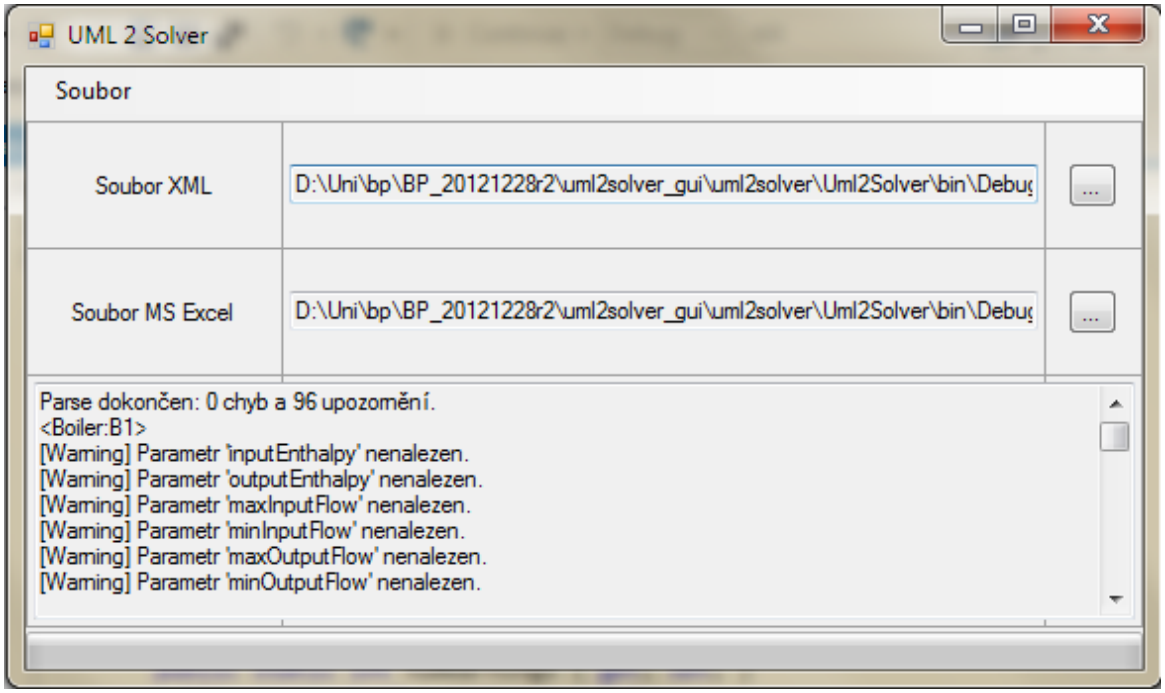
Pokud je splněna podmínka načtení dat bez chyb, uživateli se zpřístupní položka *Soubor* → *Vyřešit* v hlavním menu, která spustí solver Gurobi. Po vyřešení optimalizační úlohy se v konzoli programu zobrazí výsledek optimalizace a do **.xls(x)* souboru se do nového listu „Output“ zapíší navrhované hodnoty veličin řízených komponent a výnosy.



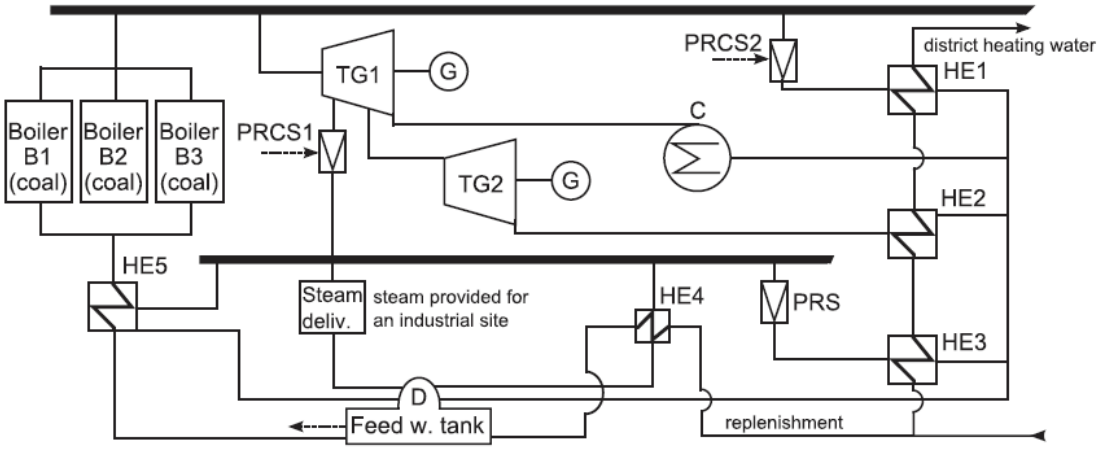
Obr. 3.11: Nastavení názvů listů *.xls(x) souboru.

3.4 Příklad

Součástí zadání je demonstrace výsledného nástroje na příkladě, jehož zadání provozu je patrné z obr. 3.13. Toto zadání bylo převedeno do object diagramu v programu Visual Paradigm – výsledné schéma včetně pomocných bločků je patrné z obr. 3.14 a v celém rozlišení přiloženo na CD, stejně tak jako projekt obsahující zdrojové diagramy. Příslušný **.xlsx* soubor se vstupními parametry je přiložen na CD (z důvody velikosti by obrázek nebyl čitelný, proto jej zde neuvádím). Jedná se o optimalizační úlohu s 26-hodinovým časovým vektorem se vzorkováním po 1h.



Obr. 3.12: Ukázka chybového výpisu.



Obr. 3.13: Schéma zadání. Převzato z [2], Fig. 1.

Obr. 3.14: Schéma provozu v příkladu.

4. Závěr

V rámci této práce byl navržen ucelený nástroj pro modelování a optimalizaci provozu kogeneračních provozů. Nástroj využívá grafické prostředí programu Visual Paradigm k tvorbě schémat provozů ve formě object diagramů za použití navržené knihovny komponent (class diagram).Takto sestavená schémata jsou pak exportována do formátu XML. Visual Paradigm for UML byl vybrán na základě stanovených kritérií v kapitole 2.

Systém zadávání vstupních dat zůstal zachován vzhledem k přehlednosti a dlouhodobého užívání ve formě souboru **.xls(x)* .

Posledním článkem nástroje je program Uml2Solver, který načte oba zdrojové soubory (**.xml* a **.xls(x)*), dle nichž si vytvoří vlastní popis systému. Program zkontroluje navržené schéma, vytvoří optimalizační úlohu pomocí frameworku JD a pošle ji do solveru. Výsledky optimalizace jsou při úspěšném řešení zapsány do nového listu **.xls(x)* souboru se vstupy.

Vzhledem k navrženému popisu komponent lze nástroj snadno rozšířit o další komponenty dle budoucí potřeby. Dalším stupněm vývoje nástroje by mohlo být vytvoření vlastního dedikovaného grafického nástroje, který by byl definován dle potřeb zpracovávajícího programu, popř. v něm přímo integrován.

V době odevzdání této práce nebylo z časových důvodů zcela odladěno správné automatické sestavení optimalizační úlohy v programu Uml2Solver.

Použitá literatura

[1] HÁK, Josef. *Návrh SW platformy pro formulaci a řešení optimalizačních úloh v energetice*. Praha, 2012. Diplomová práce. ČVUT.

[2] DVOŘÁK, Michal a Petr HAVEL. Combined heat and power production planning under liberalized market conditions. *Applied Thermal Engineering*. 2012, s. 163-173.

[3] *Object management group - UML* [online]. Copyright © 1997-2012 [cit. 2012-12-29]. Dostupné z: http://www.uml.org/

[4] *OMG SysML* [online]. Copyright © 1997-2012 [cit. 2012-12-29]. Dostupné z: http://www.omgsysml.org/

[5] *OpenModelica* [online]. Copyright © 2012 [cit. 2012-12-29]. Dostupné z: https://www.openmodelica.org/

[6] MICROSOFT. *Microsoft .NET Framework: .NET Downloads, Developer Resources & Case Studies* [online]. ©2012 [cit. 2012-12-29]. Dostupné z: http://www.microsoft.com/net

[7] *StarUML - The Open Source UML/MDA Platform* [online]. 2008 [cit. 2012-12-29]. Dostupné z: http://staruml.sourceforge.net/en/

[8] *ArgoUML* [online]. © 2001 - 2009 [cit. 2012-12-29]. Dostupné z: http://argouml.tigris.org/

[9] *Papyrus UML web site* [online]. © 2003-2012 [cit. 2012-12-29]. Dostupné z: http://www.papyrusuml.org/

[10] *Eclipse - The Eclipse Foundation open source community website* [online]. © 2012 [cit. 2012-12-29]. Dostupné z: http://www.eclipse.org/

[11] SCHAMAI, Wladimir. *Modelica Modeling Language (ModelicaML): A UML Profile for Modelica*. Linköping, 2009. Dostupné z: http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-20553. Technical report. Linköping University Electronic Press.

[12] *UML Tool for Fast UML Diagrams* [online]. [cit. 2012-12-29]. Dostupné z: http://www.umlet.com/

[13] *UML CASE tool for software development: Visual Paradigm for UML* [online]. 2012 [cit. 2012-12-29]. Dostupné z: http://www.visual-paradigm.com/product/vpuml/

[14] Visual Studio 2012 Express for Windows Desktop. MICROSOFT. *Visual Studio 2012* [online]. ©2012 [cit. 2012-12-29]. Dostupné z: http://www.microsoft.com/visualstudio/eng/downloads#d-express-windows-desktop

[15] MICROSOFT. *Microsoft Download Center: Download 2007 Office System Driver: Data Connectivity Components from Official Microsoft Download Center* [online]. ©2012 Copyright [cit. 2012-12-29]. Dostupné z: http://www.microsoft.com/en-us/download/details.aspx?id=23734

[16] GUROBI OPTIMIZATION. *Gurobi Optimizer* [online]. © 2012 [cit. 2012-12-29]. Dostupné z: http://www.gurobi.com

[17] VAN HEESCH, Dimitri. *Doxygen* [online]. Copyright © 1997-2012 [cit. 2012-12-29]. Dostupné z: http://www.stack.nl/~dimitri/doxygen/

[18] MICROSOFT. *MSDN – the Microsoft Developer Network* [online]. © 2012 [cit. 2012-12-29]. Dostupné z: http://msdn.microsoft.com/en-us

Přiložené CD

Na přiloženém CD je uložena elektronická verze této práce, knihovna komponent v samostatném VP projektu, příklad a program Uml2Solver se zdrojovými kódy.

Struktura adresářů:

- /bp_prasekjan.pdf – elektronická verze tohoto dokumentu
- /driver/ - obsahuje instalační soubor AccessDatabaseEngine 2007
- /knihovna_vp/ - obsahuje prázdný projekt ve Visual Paradigm pouze s knihovnou
- /obr/ - složka s obrázky použitými v BP v plném rozlišení
- /priklad/ - obsahuje zpracovaný příklad
- /uml2solver_zdroj/ - složka se zdrojovým projektem pro Visual Studio 2012 a dokumentací
- /uml2solver/ - složka obsahující zkompilovaný program s přiloženými knihovnami
    - /x64/ - 64-bit verze
    - /x86/ - 32-bit verze