Souhrnná výzkumná zpráva k projektu

Nástroj pro tvorbu schémat technologických procesů v oblasti čistíren odpadních vod (č. smlouvy 006770/2018/00)

za rok 2018

Ing. Radek Kočí, Ph.D.

1 Cíle projektu a postup řešení

Projekt se věnuje výzkumu v oblasti návrhu technologických procesů čistíren odpadních vod, jehož cílem je vytvoření simulačního jádra pro výpočty hmotnostních a energetických bilancí kombinovaných technologických linek. Simulační jádro je doplněno nástrojovou podporou pro tvorbu a bilanční výpočty navržených technologických procesů.

Cíle projektu stanovené pro rok 2018 jsou následující

1. Návrh architektury a implementace prototypu. Zahrnuje základní koncept tvorby modelů a jejich simulaci, základní uživatelské prvky ovládání nástroje (ukládání a načítání modelů, editace modelů, tvorba základních sestav atributů a výstupních reportů). Simulace se zaměří pouze na přímé technologické linky bez zpětných vazeb. Důležitým požadavkem byla jednoduchost přidávání nových typů výpočetních elementů (jednotek a proudů) do nástroje.

2. Základní implementace nástroje zahrnující prvky uvedené v bodu 1.

Během roku 2018 byla vytvořena základní koncepce nástroje pro návrh, modelování a simulaci technologických procesů čistíren odpadních vod. Nástroj je rozdělen do dvou základních částí. Grafické uživatelské rozhraní se základními prvky ovládání a výpočetní modul uchovávající informace o modelu a zajišťující jeho simulaci.

2 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní realizuje základní prvky ovládání, zejména možnost vkládat a rušit elementy modelu (bloky a propoje), posunovat a otáčet elementy modelu, vytvářet a zobrazovat sestavy vypočítaných atributů, spouštět simulační výpočty či ukládat a načítat modely. Na obrázku 1 je zachycen snímek obrazovky hlavního okna aplikace, kde je možné vidět rozložení prvků rozhraní, jednoduchý model po provedení výpočtu s jednou sestavou atributů pro proudy OV3 a OV4.

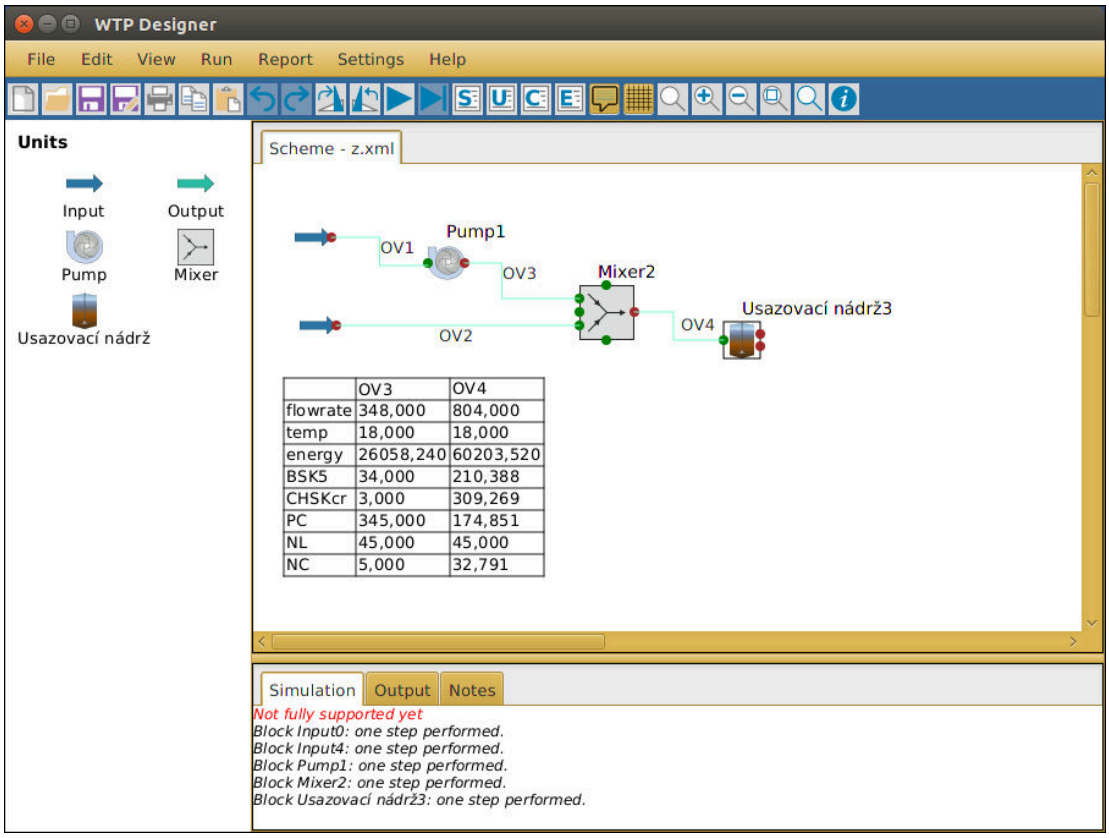
3 Výpočetní modul

Model technologických procesů je tvořen předdefinovanými výpočetními jednotkami (bloky), které pracují se vstupními a výstupními proudy. Proud reprezentuje technologické médium, např. voda, vzduch nebo kal, a je definován množinou atributů (teplota, průtok, složení apod.) Výpočetní jednotka provede bilanční výpočty výstupních proudů na základě vstupních proudů a interních atributů jednotky.

Bloky jsou navzájem propojeny spojeními (*connection*), která reprezentují komunikační kanál pro jednosměrný přenos proudů (dat) z výstupu jednoho bloku na vstup dalšího bloku. Spojení vždy nabývá hodnot asociovaného typu proudu. Kromě toho má každý model vstupní proudy modelující média vstupující do technologického procesu a výstupní proudy modelující média, která vystupují z procesu a dále nejsou zpracovávány.

3.1 Základní architektura

Na obrázku 2 je zachycen koncept základní architektury modelu a jeho prvků. Model je reprezentován třídou Scheme, která uchovává informace o výpočetních jednotkách v modelu (třída $\mathrm{UnitBlock}$) a jejich spojeních ($\mathrm{Connection}$). S každou jednotkou jsou asociovány proudy (třída Stream) ve vstupních a výstupních portech bloku. S každým proudem a každým blokem jsou asociovány jejich atributy (třída $\mathrm{Attribute}$).



	OV3	OV4
flowrate	348,000	804,000
temp	18,000	18,000
energy	26058,240	60203,520
BSK5	34,000	210,388
CHSKcr	3,000	309,269
PC	345,000	174,851
NL	45,000	45,000
NC	5,000	32,791

Obrázek 1: Náhled obrazovky nástroje s jednoduchým modelem a průběhem výpočtu.

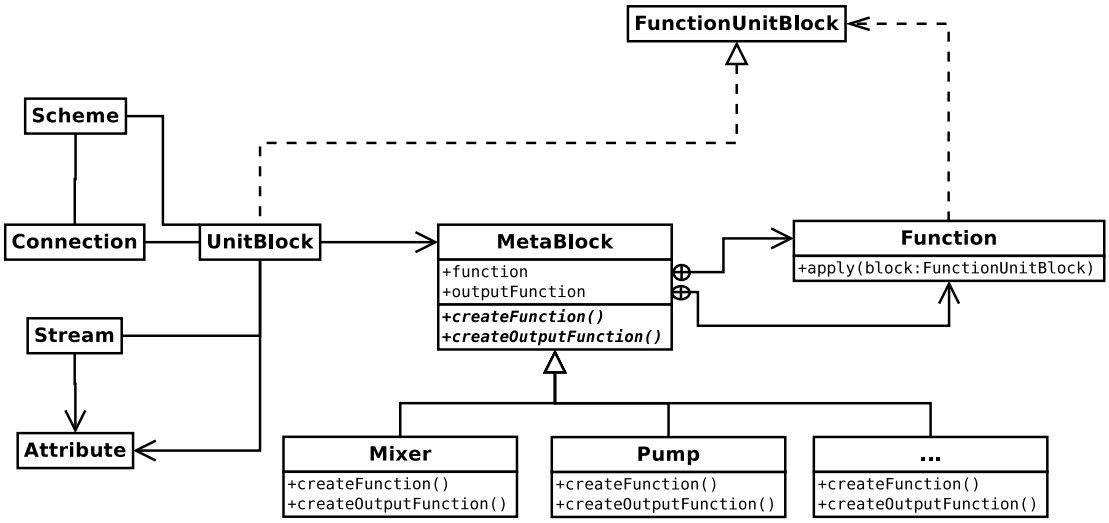
Třída UnitBlock (resp. její instance) reprezentuje konkrétní blok v schématu. Definice bloku, tj. reprezentace typu příslušného bloku, je modelována třídou MetaBlock, resp. třídami odvozenými z MetaBlock (tyto instance budeme nazývat meta-bloky). Pro každý nově vložený typ bloku se vytvoří třída odvozená od třídy MetaBlock. Tato nová třída definuje metody pro vytvoření interní výpočetní funkce a výstupní funkce (odvozeno od třídy Function). Instance tříd reprezentujících funkce jsou pak uloženy jako atributy function a outputFunction meta-bloků. Tyto funkce pracují s typem FunctionUnitBlock, který definuje rozhraní na konkrétní blok (instance třídy UnitBlock), nad kterým se funkce vykonávají. Význam těchto funkcí bude popsán v kapitole 3.2.

3.2 Výpočetní funkce

Důležitou vlastností nástroje je možnost nahlížet na teoreticky vypočítaná data a nahradit je empiricky získanými daty pro další části výpočtu. Tímto způsobem je možné porovnávat chování modelu s teoretickými hodnotami získanými výpočtem se skutečnými hodnotami získanými měřením na reálných zařízení a analyzovat tak navržený model.

Interní funkce slouží pro bilanční výpočet těch atributů, které je možné uživatelským způsobem modifikovat. Vypočítané hodnoty jsou uloženy do atributů bloku, které si pak může uživatel prohlížet a vybrat, zda se pro další výpočet použijí tyto vypočítané hodnoty nebo hodnoty naměřené a zadané uživatelem. Ukázka takového náhledu je zachycena na obrázku 3. Šedě podkreslená pole reprezentují vypočítané hodnoty, výběrovými boxy lze určit, které hodnoty se dále použijí.

Výstupní funkce slouží pro bilanční výpočet bloku a generování výstupních proudů. V rámci



Obrázek 2: Základní koncept architektury.

výstupní funkce se počítají atributy, které nelze uživatelským způsobem modifikovat, a pro výpočet se používají atributy bloku. Podle zadané volby se použijí buď hodnoty vypočítané interní funkcí nebo hodnoty zadané uživatelem.

Ukázka definice obou funkcí pro meta-blok Pumpa včetně komentáře je na obrázku 4.

3.3 Simulace bilančních výpočtů

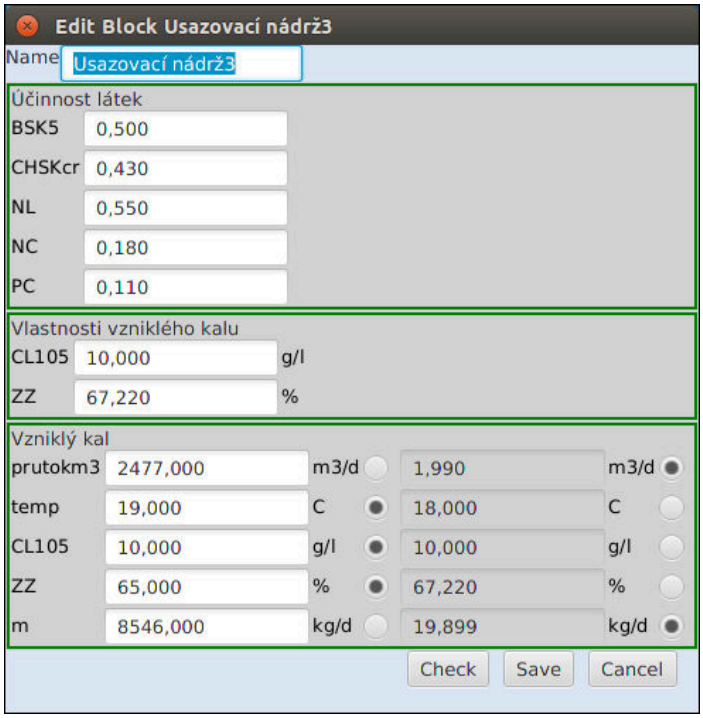
Algoritmus simulace je rozdělen do několika částí. Při řešení v roce 2018 jsme se zaměřili na modely s přímým přenosem technologických dat, tj. bez zpětných vazeb. Z toho plyne i konstrukce algoritmů, která bude v dalším roce upravena na detekci zpětných vazeb.

Nejprve je nutné získat seznam výpočetních jednotek (bloků) seřazených podle pořadí, v jakém se budou jednotlivé bloky vyhodnocovat, tj. vyvolávat jejich funkce. V rámci algoritmu lze také provést vyhodnocení korektnosti modelu, tj, zda jsou zapojeny všechny bloky modelu, zda je každý blok napojen na jiný blok nebo vstupní proud apod. Algoritmus výpočtu simulačního seznamu bloků ($\mathrm{simList}$) je uveden v algoritmu 2 v příloze A.1.

Dalším krokem je provedení simulačních výpočtů. Algoritmus projde simulační seznam od začátku do konce a nad všemi bloky volá postupně interní a výstupní funkci. Funkce jsou volány pouze v případě, že některý se vstupů bloku zaznamenal změnu hodnot alespoň jednoho atributu. Protože uživatel může vnutit jinou hodnotu než vypočítanou i po provedení simulačního výpočtu, je po takové změně nutné provést simulaci znovu. V takovém případě ale nedává smysl počítat bloky, u kterých k žádné změně nedošlo. Algoritmus simulace je naznačen v algoritmu 1.

4 Vyhodnocení a plán prací pro rok 2019

Základní funkce nástroje podle specifikovaných cílů pro rok 2018 jsou realizovány v souladu se smlouvou objednatele. Výpočetní jednotky, dosud dodané zadavatelem, jsou implementovány a dodaný základní model je vytvořen a odsimulován. Byly implementovány 3 výpočetní jednotky (pumpa, směšovač a usazovací nádrž), dvě speciální jednotky reprezentující vstupní a výstupní proudy a 3 typy proudů (vzduch, voda a kal). Aplikace je připravena na generování reportů (aktuálně generuje



Obrázek 3: Náhled editačního okna bloku s vypočítanými a nastavitelnými atributy.

souhrnné reporty pro výpočetní jednotky a proudy), dopracování bude záviset na dodaných datech zadavatelem (tj. co a v jakém stylu mají reporty obsahovat).

Jak již bylo řečeno, při řešení v roce 2018 jsme se zaměřili na modely s přímým přenosem technologických dat, tj. bez zpětných vazeb. Modely se zpětnou vazbou budou řešeny v roce 2019. V roce 2019 také dojde k dokončení nástroje a realizaci zbývajících výpočetních jednotek a proudů. V průběhu března bude nástroj demonstrován a podle připomínek bude doladěna podoba uživatelského rozhraní, způsob práce s modely, simulační a výpočetní algoritmy apod. Během března a dubna proběhne testování a úpravy aplikace a v květnu pak příprava dokumentů a předání.

Data:
simList : seznam bloků s definovaným pořadím pro vyhodnocování
forall the $b \in simList$ **do**
    inicializuj simulaci nad b (vstupy se označí jako změněné)
end
forall the $b \in simList$ **do**
    if *některý vstup bloku b se změnil* **then**
        call interní funkci nad b call výstupní funkci nad b
    end
end

Algoritmus 1: Algoritmus vytvoření seznamu bloků pro vyhodnocování.

```
public class PumpBlock extends MetaBlock {
    @Override
    protected Consumer<FunctionUnitBlock> createFunction() {
        return (b) -> {
            // Výpočet atributů (spotřebovaná energie pumpy).
            double p = b.getAttributeDouble("P");
            double time = b.getAttributeDouble("time");
            b.setAttribute("Psum", p * time);
        };
    }

    @Override
    protected Consumer<FunctionUnitBlock> createOutputFunction() {
        return (b) -> {
            // Generování výstupních hodnot. V případě pumpy pouze
            // kopírování vstupního proudu na výstupní.
            b.forEachPortAttribute("in", a ->
                b.setPortAttribute("out", a.id(), a.getDouble()));
        };
    }
}
```

Obrázek 4: Ukázka definice funkcí v meta-bloku.

A Přílohy

V této části jsou uvedeny algoritmy a ukázky demonstrující vlastnosti nástroje a simulace popisované v textu.

A.1 Algoritmus výpočtu simulačního seznamu

Data:
blockList : seznam všech bloků modelu
ancDict : slovník bloků se seznamy jejich přímých předchůdců
procList : seznam bloků pro aktuální zpracování
waitList : seznam bloků, které nemají zpracovány všechny předchůdce
Result:
simList : výstupní seznam bloků s definovaným pořadím pro vyhodnocování

```
1  forall the b ∈ blockList : b není vstupní blok do
2  |  ancDict(b) ← seznam přímých předchůdců b
3  end
4  forall the b ∈ blockList : b je vstupní blok do
5  |  přidej b do procList
6  |  přidej b do simList
7  end
8  while procList ≠ ∅ do
9  |  b ← procList.removeFirst // odstraní první blok ze seznamu
10 |  forall the d ∈ následníci bloku b do
11 |  |  if ! simList.contains(d) then
12 |  |  |  forall the a ∈ ancDict(d) do
13 |  |  |  |  if simList.contains(a) then
14 |  |  |  |  |  simList.remove(a)
15 |  |  |  |  end
16 |  |  |  end
17 |  |  |  if ancDict(a) = ∅ then
18 |  |  |  |  simList.add(d)
19 |  |  |  |  ancDict.remove(d)
20 |  |  |  |  waitList.remove(d)
21 |  |  |  |  procList.add(d)
22 |  |  |  else
23 |  |  |  |  if ! waitList.contains(d) then
24 |  |  |  |  |  waitList.add(d)
25 |  |  |  |  end
26 |  |  |  end
27 |  |  end
28 |  end
29 end
```

Algoritmus 2: Algoritmus vytvoření seznamu bloků pro vyhodnocování.