# POZNÁMKY K SOFTWAROVÉMU PROJEKTU
# PROJEKT FLOWOPT

## 1 Úvod

Projekt FlowOpt si klade za cíl vytvoření sady spolupracujících aplikací pro vytváření rozvrhů z pracovních postupů. V této sadě budou následující aplikace:

1) Editor pracovních postupů (Workflow editor)
2) Rozvrhovací aplikace (Scheduling engine)
3) Zobrazovač rozvrhů s (vybranými) možnostmi jejich úprav a přerozvrhování (iGantt)
4) Řídící aplikace pro správu projektů a uživatelsky příjemnější propojení jednotlivých aplikací (Project orchestration)

Toto dělení projektu umožňuje zcela oddělené použití jednotlivých aplikací, ale i jejich spolupráci, ať již „ručním“ způsobem (použitím výstupního souboru jedné aplikace jako vstupního souboru druhé aplikace) nebo „automatickým“ způsobem pomocí řídící aplikace, která navíc umožní tvorbu projektů obsahujících více pracovních postupů nebo rozvrhů.

Celý projekt bude vytvořen pro platformu jazyka Java, čímž bude zajištěna maximální možná přenositelnost mezi operačními systémy.

## 2 Řídící aplikace

Řídící aplikace bude umožňovat:

1) Vytváření, ukládání a načítání projektů FlowOpt do/ze souboru, které v sobě soustřeďují související pracovní postupy a rozvrhy
2) Zobrazování seznamů pracovních postupů a rozvrhů v projektu
3) Spouštění editoru pracovních postupů pro vytvoření nového pracovního postupu nebo úpravu stávajícího
4) Spouštění rozvrhovače pro generování nových rozvrhů pro vybraný pracovní postup
5) Spouštění zobrazovače/editoru rozvrhů pro vybraný rozvrh

K řízení ostatních aplikací (k jejich vzájemné komunikaci) bude výhradně používat

1) Parametry z příkazové řádky
2) Vstupní a výstupní soubory
3) Informaci o tom, zda aplikace byla ukončena

Formáty souborů, které budou jednotlivé aplikace používat, budou založeny na formátu XML. V aplikaci budou nejméně tyto typy souborů v XML formátu:

1) Projekt FlowOpt
2) Pracovní postup (Workflow)

3) Zadání rozvrhovacího problému (výstupní soubor editoru pracovních postupů a vstupní soubor rozvrhovací aplikace)
4) Rozvrh (výstupní soubor rozvrhovací aplikace a vstupní soubor pro zobrazovač/editor rozvrhů)

Některé tyto soubory budou kromě svých „klíčových dat“ (např. samotného pracovního postupu) obsahovat i „meta informace“ pro potřeby vytváření vazeb mezi projekty, pracovními postupy a rozvrhy, pro správu verzí souborů (pokud bude vazba mezi pracovním postupem a rozvrhem, tak je dobré vědět, že pracovní postup byl změněn od doby, co z něj byl rozvrh vygenerován) a pro potřeby předávání informací řídící aplikaci (např. pokud dojde k ukončení práce s editorem pracovního postupu, může soubor obsahovat příznak, zda má řídící aplikace spustit rozvrhování nebo ne)[1].

# 3 Moduly „mezi aplikacemi“

Aby nedošlo k redundantnímu vytváření kódů, budou některé moduly sdíleny mezi aplikacemi. Tyto moduly budou mít podobu knihovny „jar“. Jde o moduly (zde s odkazem na Tomášův diagram):

1) „Data for scheduling engine“
2) „Schedules“

Tyto moduly budou zastřešeny hlavní třídou, která bude spolu s pomocnými třídami poskytovat rozhraní nejméně pro následující operace:

1) Načtení dat ze souboru
2) Uložení dat do souboru
3) Čtení a úpravu (paměťové) datové struktury (tedy například: úpravu zadání rozvrhovacího problému přidáním aktivity, zdroje, apod., získání seznamu aktivit, atd.)

Vytváření těchto modulů bude vždy silně závislé přinejmenším na obou aplikacích, které budou modul sdílet. Proto musí být dobře definovány potřeby obou stran a nalezena vhodná struktura, která bude schopná všechna potřebná data uchovávat (jak v paměti, tak v souboru).

# 4 Zadání rozvrhovacího problému

Rozvrhové problémy, které bude rozvrhovač řešit, budou vymezeny možnostmi, které nabízí následující popis rozvrhovacího problému.

| Predikát | Význam |
|---|---|

[1] Například již při spouštění „Workflow editoru“ by bylo možné parametrem v příkazové řádce říci, zda je spouštěn jako samostatná aplikace nebo z řídící aplikace, a tento editor poté může přizpůsobit své možnosti nabízené uživateli – například přidat do nabídky příkaz „Vytvořit rozvrh“, pokud je editor spuštěn z řídící aplikace. Po stisku na toto tlačítko by pak došlo k uložení práce do souboru s přídavným příznakem (meta informací) pro řídící aplikaci, která by okamžitě po ukončení editoru spustila rozvrhovací aplikaci a nakonec i zobrazila výsledný rozvrh.

| | |
|---|---|
| activity(A,$D_U$,C,$D_D$,$C_E$,$C_L$) | Existuje aktivita A s trváním $D_U$ a cenou C, která má být vykonána do doby (její konec musí být dříve) $D_D$. Multiplikativní koeficient pro změnu ceny v případě předčasného dokončení je $C_E$, pozdního dokončení $C_L$. |
| decomp(A,{$A_1$,$A_2$,…,$A_N$},T) | Aktivita A je složená z (pod)aktivit $A_1$, $A_2$, …, $A_N$. T je označením typu složení – zdali se jedná o paralelní (pod)aktivity - má-li se vykonat aktivita A, pak se mají vykonat všechny její (pod)aktivity - nebo o alternativní (pod)aktivity, kdy k vykonání aktivity A stačí, vykoná-li se jedna z aktivit $A_1$, $A_2$, …, $A_N$. Žádná z aktivit $A_1$, $A_2$, …, $A_N$ se nemůže vyskytovat jako (pod)aktivita více složených aktivit. |
| res(R) | Existuje unární zdroj R.[2] |
| resdep(A,R) | Aktivita A potřebuje ke svému vykonání zdroj R. Aktivita může spotřebovávat více zdrojů. |
| log($A_1$,$A_2$,O) | Binární logická podmínka na aktivity $A_1$ a $A_2$. O označuje logický operátor (AND, OR, XOR). |
| temp(A1,A2,T) | Temporální vazba mezi aktivitami $A_1$ a $A_2$. T označuje typ temporální vazby z množiny { SS, SE, ES, EE }, kde například ES znamená, že druhá aktivita nesmí začít dříve, než byla první skončena. |

[2] Zde bychom se měli dohodnout na tom, umožníme-li použití pouze unárních zdrojů nebo také kumulativní zdroje (reservoáry).