Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Interaktivní generátor rozvrhu

Plzeň, 2013 Lukáš Vlček

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 21. dubna 2013

Lukáš Vlček

Poděkování

Na tomto místě bych rád poděkoval panu Doc. Ing. Pavlovi Heroutovi, Ph.D. a panu Ing. Tomášovi Riegerovi za čas, který věnovali konzultacím a řešení problémů vyskytujících se v této práci.

Abstrakt

Tato práce se zabývá návrhem a implementací interaktivního generátoru rozvrhu pro základní a střední školy. Tento bude následně využit jako jedna ze služeb systému Škola OnLine. Navržený systém, který je široce konfigurovatelný, používá heuristické algoritmy s následným výběrem nejlepšího řešení. Systém je vhodný především pro malé až středně velké školy, to znamená s počtem tříd cca do dvaceti. Předností řešení oproti podobným řešením je rychlost nalezení vyhovujícího rozvrhu.

Abstract

Interactive generator of schedule

This thesis deals with design and implementation of an interactive schedule generator for elementary and high schools. Afterwards, this one will be used as one of services of Škola OnLine system. The designed system is widely configurable and uses heuristic algorithms followed by the choice of the best solution. The system is suitable mainly for small to middle–sized schools, meaning schools with up to 20 classes. The advantage of the solution, unlike similar ones, is the speed with which the suitable schedule is found.

.

Obsah

1 Úvod

Na internetu je provozován systém Škola OnLine, který nabízí školám zpracování jejich kompletní agendy na jednom místě. Ke školám patří neodmyslitelně i rozvrhy jednotlivých tříd, učitelů nebo místností. Jejich vytváření není triviální záležitostí a při ručním vytváření (připínání lístečků na tabuli) může trvat i několik dní. Pro účely pohodlné tvorby těchto rozvrhů existuje standalone aplikace aSc Rozvrhy, jež není přímo integrovaná do systému Škola OnLine, ale je možné ji připojit přes rozhraní. Tato aplikace je schopná připravit rozvrhy i pro velmi velké školy obsahující desítky až stovky tříd a v současné době je považována za jeden z nejlepších dostupných rozvrhovacích systémů. Její nevýhodou je poměrně velká cena a značná komplexnost, která je pro menší školy téměř zbytečná. Proto tvůrci systému Škola OnLine začali uvažovat o vytvoření jednodušší a levnější varianty rozvrhovacího systému. Protože malých a středně velkých škol je v systému Škola OnLine evidováno značné množství, bylo by výhodné mít rozvrhovací aplikaci v systému integrovánu a těmto školám ji poskytovat jako součást již placené služby. Výhodou tohoto způsobu by bylo, že tyto školy by si mohly vyzkoušet přípravu rozvrhu „zdarma“ pomocí integrované aplikace a teprve v případě, že by jim navržený rozvrh nevyhovoval, by požádaly o další placenou službu formou využití aplikace aSc Rozvrhy.

Hlavním úkolem této práce je analyzovat, navrhnout a implementovat aplikaci, která by splňovala výše uvedené požadavky a bylo by možno ji přímo integrovat do systému Škola OnLine. Na aplikaci jsou mimo jiné kladeny požadavky na co možná nejjednodušší používání a rychlost generování nových rozvrhů. Vytvořená aplikace by přitom neměla být přímou konkurencí programu aSc Rozvrhy, nýbrž by měla z pohledu uživatele sloužit jako levnější a jednodušší alternativa, vhodná hlavně pro kapacitou malé a střední školy.

Protože z pohledu řešení rozvrhovacího problému je známo a popsáno několik vyhovujících způsobů řešení, bude nedílnou částí této práce analýza těchto způsobů a výběr vhodného řešení pro zamýšlenou aplikaci. Ta bude programována tak, aby plánovaná integrace do Školy OnLine mohla proběhnout co nejjednodušším způsobem.

2 Rozvrhování v praxi

2.1 Ruční

Jelikož se práce zabývá generováním rozvrhu pro školy, budou v této kapitole popsány poznatky, které byly získány přímo z praxe. Jako zdroj posloužil rozhovor s paní Mgr. Jiřinou Reitmayerovou z 31. Základní školy v Plzni, která pro tuto školu několik desítek let vytvářela rozvrhy ručně. Konkrétně se jednalo o rozvrhy pro druhý stupeň této školy.

2.1.1 Způsob rozvrhování

Na začátku vytváření rozvrhu bývá soubor úvazků, jež určují který učitel bude v které třídě vyučovat který předmět. Tyto úvazky se většinou přebírají z minulých let. Samotný způsob vytváření rozvrhu je takový, že se na magnetickou tabuly umístily jednotlivé prázdné rozvrhy, přičemž každý reprezentoval jednu třídu. Tříd bylo pro danou školu od 15 do 20. Poté byly připraveny různě barevné kartičky, z nichž každá barva znamenala jiného učitele a jedna kartička reprezentovala jeden předmět. Následně začalo samotné umisťování.

Začínalo se s předměty, které byly stejné pro první i druhý stupeň této školy, aby se poté mohlo na každém stupni rozvrhovat nezávisle. Následovalo rozvrhování těch předmětů, u kterých se očekávalo, že je bude těžké umístit. Jednalo se hlavně o tělocviky, u kterých se muselo počítat s možnou výukou od externích pracovníků (trenérů, ...), dále o plavání, kde bylo nutné povolení a pronajmutí bazénu (možno jen na určitou hodinu). Další hodiny, které se vybíraly, byly ty, jež byly půlené (rozdělení třídy na angličtináře, němčináře, ...), a také ty, co se vyučují v lichých a nebo sudých týdnech. U těchto hodin byla snaha je umisťovat na stejné časové úseky. Co se týče učitelů, byly jako první rozvrhovány ty hodiny, ve kterých měl učit ředitel. To bylo z toho důvodu, že měl jisté úřední hodiny, které musel dodržovat a výuka se s nimi nesměla krýt. Jako poslední se umisťovaly krátké hodiny jako je Český jazyk, Matematika atd. Pokud se nenašlo volné místo, kam by mohl být předmět umístěn, musel se jiný lísteček odebrat a zkusit umisťování znovu. Pro odebírání platilo to pravidlo, že se odebírá položka, která je umístěna nejhůře, nebo jí je možné umístit snadno (jednohodinová s jedním

učitelem).

2.1.2 Podmínky v rozvrhování

Místnosti

Mezi důležité podmínky pro rozvrh patří ta, že většina místností musí být využita na 100 %. U tělocvičen bylo nesplnění této podmínky považováno za neúspěch, u ostatních místností to nebylo tak kritické, ovšem znamenalo to, že musejí být hledány další možné optimalizace. Zároveň bylo vyžadováno, aby se většina výuky vyučovala v kmenové učebně třídy a pouze, pokud to nebylo možné, vybrala se jiná třída. Specializované předměty se samozřejmě vyučovaly v místnostech pro to určených (laboratoře, PC učebny, ...), ovšem bylo preferováno žáky příliš nestěhovat.

Třídy

Co se týče tříd, bylo stanoveno to, že třídy by měly mít v řadě čtyři hodiny a poté by měla následovat volná hodina na oběd. Optimalizovat se mělo i vůči jídelně, aby nedocházelo k jejímu přeplnění. Delší volné bloky, či více volných bloků byl v rozvrhu tříd nežádoucí jev. Nulté hodiny u rozvrhu třídy se měly vyskytovat jen výjimečně. Co se týče rozložení výuky v týdnu, byla snaha, aby byla výuka rovnoměrně rozprostřena v rámci všech dnů.

Učitelé

Pokud uvažujeme rozvrhy učitelů, byla snaha jim rozložit rozvrh optimálně, aby například neučili jeden den pět hodin v kuse a další neměli nic. Volné hodiny u učitelů nebyly tak kritickou záležitostí jako u tříd, ovšem bylo cílem je minimalizovat. Další požadavek byl ten, aby učitel měl pouze 7 hodin denně, počet hodin celkem byl dán úvazkem.

Předměty

Výuka jednotlivých předmětů měla také svá pravidla. U předmětů jako přírodopis či pracovní výchova bylo preferováno, aby se vyučovaly odpoledne. Bylo to z toho důvodu, že tyto předměty měly předepsány nějaké laboratorní práce, respektive práce na pozemku, a proto bylo vhodné, aby měli žáci po skončení výuky volno a pokud by se vše nestihlo, mohlo se ještě chvíli pokračovat. Další požadavek byl ten, že dvě hodiny od stejného předmětu se neměly vůbec vyučovat ve stejný den a jen výjimečně se mohly vyučovat dva dny po sobě.

2.1.3 Zhodnocení

Rychlost

Ruční vytváření rozvrhu bylo velice náročné. Rozvrhování se skládalo ze dvou fází. V té první se naskládaly lístečky na tabuli tak, aby nevznikaly kolize, ve druhé fázi poté následovalo přesunování lístečků podle různých preferencí a podmínek. Vytvoření rozvrhu pro 15–20 tříd trvalo od dvou do tří dnů.

Kvalita

Ze zkušeností mi bylo sděleno, že ručně vytvořený rozvrh podle předchozích bodů obsahoval 95 % rozvrhů, které byly velice pěkné a splňovaly všechny důležité podmínky. Zbylých 5% bohužel byly takové rozvrhy, kdy například učitel měl dlouhou pauzu mezi hodinami nebo učil mnoho odpoledních hodin. Co se týče rozvrhování počítačem, byla vyslovena obava, zda je tento způsob citlivý vůči rozvrhům učitelů. Pokud při ručním rozvrhování bylo nutné porušit některou z výše uvedených podmínek, bylo zapotřebí si toto porušení obhájit a zdůvodnit ho.

2.2 Elektronické

Pokud nechceme generovat rozvrh ručně, můžeme využít možnosti některého z programů, které tuto práci udělají za nás. V této části bude popsán pro-

gram aSc Rozvrhy, který se vyvíjí již několik let a je opravdu velice kvalitní. V této práci budou rozvrhy, generované tímto programem, považovány za referenční a budeme se k nim snažit co nejvíce přiblížit. Chtěl bych zdůraznit, že předkládaná práce nemá za cíl konkurovat, či nahradit tento produkt, ale pouze poskytnout jednoduchou, bezplatnou alternativu vhodnou pro malé školy, pro které je zbytečné kupovat si, pro ně relativně drahou, licenci.

2.2.1 aSc Rozvrhy

Program aSc Rozvrhy je propracovaný program umožňující generování rozvrhů pro různé typy škol. Jeho hlavní funkce jsou:

- Jednoduché zadávání údajů – rychlé a pohodlné zadávání předmětů, učitelů, úvazků, ...
- Automatické generování – na základě omezujících podmínek vygeneruje do několika minut požadovaný rozvrh
- Kontrola rozvrhu – pomáhá odstranit kolize vzniklé při generování
- Kompletní tisk rozvrhu
- Rozsáhlé možnosti exportu – Excel, PDF, ...
- Podpora pro suplování

Více informací o programu je k dispozici na domovských stránkách projektu viz [2], ze kterých bylo čerpáno i pro výše uvedený popis. Z těchto stránek lze stáhnout i demoverzi za účelem vyzkoušení základních možností programu. Plná verze programu je placená.

Na stránkách se pochopitelně nenachází žádné implementační detaily, ani dokumentace ke kódu.

3 Programování postavené na omezujících podmínkách

3.1 Úvod do rozvrhování

Tvorba rozvrhu je (jak je dokázáno), NP–úplný problém, což jsou takové problémy, které patří do skupiny NP problémů (jsou vypočítatelné v nedeterministickém polynomiálním čase) a libovolný problém z NP je na ně převeditelný. Dále se budeme zabývat pouze tvorbou rozvrhů pro školy.

Rozvrhem se rozumí přiřazení zdrojů a nějaké úlohy na konkrétní čas. Tvorba rozvrhu u škol začíná u definice úvazků učitelů, kde se určí, že učitel učí v dané třídě nějaký předmět. Pokud máme definované úvazky, snažíme se je umístit do rozvrhu. To, jak dobře jsme položku umístili, nebo kam máme položku umístit nám určí omezující podmínky. Tento styl programování se nazývá programování s omezujícími podmínkami. Popis omezujících podmínek a jejich rozdělení najdete v části 3.2. Algoritmy, které jsou kandidáty na řešení takto zadaných problémů, naleznete v části 3.3.

3.2 Omezující podmínka

Omezující podmínka je taková podmínka, která nám redukuje možnou oblast řešení problému. V souvislosti s omezujícími podmínkami mluvíme o silných a slabých podmíkách. Silné jsou ty podmínky, které musí být vždy splněny – pokud splněny nejsou, nemůžeme naše řešení považovat za správné. Slabé podmínky naopak splněny být nemusí, ovšem algoritmus by se měl snažit splnit co nejvíce slabých podmínek pro vytvoření optimálního řešení. Omezující podmínky, které jsou uvedeny dále, byly přejaty z rozvrhu školy. Při srovnání s požadavky uvedenými v části 2.1, zjistíme, že vyhovují požadavkům praxe. Z hlediska rozvrhu by tedy mohly tyto podmínky vypadat následovně:

1. **Silné podmínky**
   - Pro třídy
     - třída nesmí mít dvě vyučovací hodiny zároveň (pokud se nedělí na skupiny)
   - Pro místnosti
     - v jedné místnosti nesmí být více než jedna výuka zároveň
     - v místnosti se smí vyučovat jen určité předměty (tělocvična, laboratoř,...)
   - Pro učitele
     - učitel nesmí učit ve dvou třídách zároveň
     - učitel smí učit jen určitý počet hodin (podle úvazku)
2. **Slabé podmínky**
   - Pro třídy
     - třída musí mít pauzu na oběd
     - preferovaný začátek vyučování by měl být v 8 hodin (závisí na typu školy)
     - pauza mezi hodinami by měla být maximálně jednohodinová
     - hodiny stejného předmětu třídy by měly být ideálně rozprostřené v průběhu týdne (ne dvě stejné hodiny ve stejný den)
     - počet hodin v jednotlivých dnech by měl být podobný
     - pokud většina žáků dojíždí, je vhodné mít kratší páteční vyučování, aby stihli odjezd
     - pokud je třída rozdělena na skupiny, předměty těchto skupin by měly být vyučovány zároveň
     - předměty, které se nevyučují každý týden, by měly být rozvrhovány „přes“ sebe (tj. v tentýž časový úsek)
   - Pro učitele
     - vyžaduje pauzu na oběd
     - učí pouze v předem definovaném čase, nebo ve specifické dny
   - Pro místnosti
     - většina předmětů by se měla učit v kmenové učebně třídy

Kromě těchto podmínek je přidána ještě další obecná, a to že rozvrhovací akce musí mít přiřazenu třídu, učitele a ve většině případů i místnost (např. praxe nemají místnost), aby se dala zařadit do rozvrhu.

3.3 Přehled algoritmů

Algoritmus se dá definovat jako popis určitého postupu. V této sekci bude popsáno několik algoritmů, které jsou schopné řešit problémy zadané omezujícími podmínkami. Zdroji pro tento přehled je literatura [12], [13], [17], [18] a [19].

V popisu jsou uvedeny zástupci Systematického prohledávání (prochází všechna možná ohodnocení, tudíž pokud existuje řešení, vždy ho najde, může to ovšem trvat velmi dlouho) a evoluční algoritmy (pro řešení úlohy používají techniky napodobující evoluční procesy známé z biologie, jako jsou dědičnost, mutace, přirozený výběr a křížení).

3.3.1 Generuj a testuj

Jedná se o jednoduchý algoritmus, který, jak je patrné z názvu, vygeneruje řešení a poté zkoumá, jestli toto řešení splňuje zadané podmínky. Pokud ne, generování se opakuje tak dlouho, dokud není řešení nalezeno, nebo dokud se nepřekročí předem stanovený počet generování. Generování řešení může probíhat dvojím způsobem, buď se generuje systematicky, nebo náhodně.[8]

Příklad

Následující příklad znázorňuje pseudokód postupu při implementaci algoritmu Generuj a testuj.

```
GenerujATestuj() {
      int pocetOpakovani = 0;
      while(pocetOpakovani < pozadovanyPocet) {
          VygenerujReseni();
          if(jeReseniPlatne)
               return reseni;
          pocetOpakovani++;
    }
}
```

Zhodnocení

Tento algoritmus je velice rychlý a může dávat i dobré výsledky, ovšem může se stát, že nebude nalezeno vůbec žádné (tedy ani částečné) řešení, které ale ve skutečnosti existuje. Pro náš případ by tento algoritmus byl použitelný, avšak kvůli tomu, že je velice malá pravděpodobnost, že by generovaný rozvrh splňoval všechny silné a alespoň některé slabé podmínky, nebude pro další implementaci uvažován.

3.3.2 Hrubá síla

Řešení hrubou silou se rozumí takový algoritmus, který systematicky prochází celý prostor možných řešení problému. V případě rozvrhu by se tedy jednalo o takový algoritmus, jenž otestuje všechny možné pozice každé položky a nakonec vybere tu nejvhodnější. [4]

Příklad

Následující příkaz znázorňuje pseudokód hledání podřetězce v řetězci.

```
HrubaSilaHledaniPodretezce(string podretezec, string retezec) {
     int n = delkaRetezce;
     int m = delkaPodRetezce;
     for(i=0;i<n-m;i++) {
          j=0;
          while(j < m and retezec[i+j] == podretezec[j]) {
               j++;
               if(j == m)
                    return i;
               return -1;
          }
     }
}
```

Zhodnocení

Algoritmus hrubé síly by měl nalézt nejlepší možné řešení, nevýhodou je však délka zpracování požadavku (musí se vyzkoušet všechny, nebo alespoň většina možných řešení) a také případná paměťová náročnost. Proto se v praktických úlohách příliš nepoužívá, protože časová náročnost roste exponenciálně či dokonce faktoriálem. Pro náš případ by byl vhodný, ovšem z hlediska časové náročnosti není jedním z kandidátů na vybraný algoritmus.

3.3.3 Backtracking

Backtracking, jinak řečeno metoda pokusů a oprav, je způsob řešení založený na prohledávání stavového stromu problému [18]. Backtracking je vlastně vylepšením algoritmu hrubé síly tak, že vylučuje několik možných řešení bez přímého vyzkoušení a tím snižuje nutný počet kroků. V případě rozvrhu by algoritmus pracoval tak, že by postupně přidával položky do rozvrhu. Po každém přidání by zkontroloval, zda rozvrh splňuje všechny zadané požadavky. Pokud ne, vrátí se o krok zpět (vyhodí poslední položku a zkusí ji nahradit jinou). Tímto postupem dojde k nějakému řešení, popřípadě se poté vrátí do řešení, ve kterém bylo nejvíce umístěných položek[4].

Příklad

Příklad znázorňuje pseudokód pro nalezení uzlu ve stromu, tento uzel musí mít označení „good“.

```
boolean BacktrackingHledejUzel(Node n, Node hledany) {
     if(n == leafNode) {
          if(n == hledany)
               return true;
          return false;
     }
     else {
          foreach (child c in n) {
              return BacktrackingHledejUzel(c, hledany);
          }
     }
}
```

Zhodnocení

Algoritmus je vhodný na mnoho problémů, ovšem má exponenciální složitost, a proto se používá pouze tam, kde neexistuje optimálnější algoritmus, nebo existuje vhodná heuristika. V našem případě by se hodil pouze pro optimalizaci pokud nebylo nalezeno řešení.

3.3.4 Heuristické algoritmy

Heuristické algoritmy, nazývány též stochastické, neboli náhodné. Jedná se o takové algoritmy, které poskytují dostatečně přesné řešení v krátké době. Podaný výsledek je poté jedním ze dvou stavů, buď jako kladný výsledek (což je odpověď), nebo jako výrok neurčitosti (nevím, zda existuje řešení)[8].

Příklad

Následující postup znázorňuje postup při hledání Hamiltonovské kružnice v grafu.

```
HeuristikaKruznice(){
     SeradHranyPodleOhodnoceni();
     foreach(hrana h in hrany) {
          PridejHranuDoKostry();
          if(VzniklaKruznice())
               OdeberHranuZKostry();
     }
     return kostra;
}
```

Zhodnocení

Heuristické algoritmy se používají pro řešení složitých problémů s mnoha parametry. Dokáže dát řešení ve velice krátkém čase, ovšem toto řešení nemusí být optimální. Díky své vysoké rychlosti a možnostem dalších úprav je to jeden z kandidátů na vybraný algoritmus.

3.3.5 Genetické algoritmy

Genetický algoritmus patří mezi evoluční algoritmy, jež zahrnují také evoluční programování, evoluční strategii a genetické programování. Jsou to vyhledávací algoritmy založené na principu genetiky a přirozeného výběru (vycházejí z Darwinovy teorie o vývoji druhů). Tyto algoritmy mají vzor v přírodě, kde probíhají procesy křížení a mutace, jsou proto velice jednoduché a přitom i účinné. Proces výběru probíhá tak, že se vždy zvolí a postupně zdokonalují jednotlivé vlastnosti daného druhu, v případě programování některého hledaného řešení.

Důležité vlastnosti genetických algoritmů jsou:

- Chromozóm – řetězec informací nesoucí vlastnosti a chování jedince
- Populace – skupina jedinců, z nichž je každý charakterizován svým chromozómem
- Fitness – hodnota vyjadřující kvalitu jedince z populace, to znamená, jak dobrý má chromozóm

Algoritmus pracuje tím způsobem, že na začátku se náhodně vygenerují chromozómy pro první generaci a spočítá se fitness. Dále se aplikují tři operace, a to

- Selekce – výběr nejvhodnějších rodičů z populace (s nejlepší možnou hodnotou fitness)
- Křížení – vyměnění části chromozómů mezi rodiči
- Mutace – výměna některých částí chromozómu potomka, tzn. nebude stejný jako má rodič.

Tento postup se opakuje, dokud není naplněn požadovaný počet generací [6].

Příklad

Příklad znázorňuje pseudokód genetického algoritmu.

```
Evoluce() {
     InicializujPopulaci();
     foreach(jedinec j in populace) {
          SpocitejFitness(j);
     }
     for(i = 0; i < velikostPopulace;i += 2) {
          VyberDvaJedincePodleFitness();
          if (pravdepodobneKrizeni)
               Krizeni();
          if (pravdepodobnaMutace)
               Mutace();
          if (pravdepodobnaEvoluce)
               Evoluce();
     }
     ZachovejNejlepsiJedince();
     OpakujPodlePoctuGeneraci();
}
```

Zhodnocení

Velkou výhodou algoritmu je to, že je poměrně jednoduchý, současně dává vyhovující výsledky (většinou ne ty nejlepší možné, ale k nejlepšímu řešení se velice blíží) a nějaký výsledek je k dispozici v každém kroku. Nevýhodou je naopak skutečnost, že při vyšším počtu generací, nebo populací, je algoritmus poměrně pomalý. I přes tuto nevýhodu se jedná o jednoho z kandidátů na algoritmus řešící zkoumaný rozvrhovací problém[4].

3.3.6 Dynamické algoritmy

Tyto algoritmy pracují tak, že se snaží složitý problém rozložit na podproblémy, které vyřeší a skládáním jejich řešení dostanou řešení výsledné. Způsob rozložení na podproblémy je možné zvolit (diskrétní – spojité, deterministické - stochastické, ...).

Příklad

Příklad znázorňuje vypočtení Fibonacciho posloupnosti ze znalosti již vypočítaných čísel

```
int DynamickeFibonacci(int n) {
     if (n == 0)
          return 1;
     if (n == 1)
          return 1;
     return DynamickeFibonacci(n-1) + DynamickeFibonacci(n-2);
}
```

Zhodnocení

Dynamické algoritmy jsou poměrně rychlé a zaručují i dobré výsledky, bohužel se příliš nehodí pro tento problém generování rozvrhu. Důvodem nevhodnosti je, že rozvrhované položky se ve většině případů prolínají a navzájem ovlivňují, a proto není možné rozdělit problém na podproblémy.

4 Požadavky na generovaný rozvrh

Tato kapitola navazuje na část 3.2, ovšem budou zde již konkrétně vybírány jednotlivé podmínky a zdůrazňován jejich význam ve výsledném rozvrhu. Také zde bude popsán systém Škola OnLine, pro niž bude v budoucnu generátor použit. Vypsané požadavky jsou ty, které se budou vyskytovat v navrženém algoritmu.

4.1 Hlavní požadavky

Hlavní požadavky jsou stejné jako silné podmínky v kapitole 3.2, jedná se tedy o to, že jeden subjekt (třída, učitel, místnost), nesmí být rozvrhována na dvě události v jeden čas, pokud to není přímo vyžadováno (možno u místností – spojení dvou tříd).

4.2 Ostatní požadavky

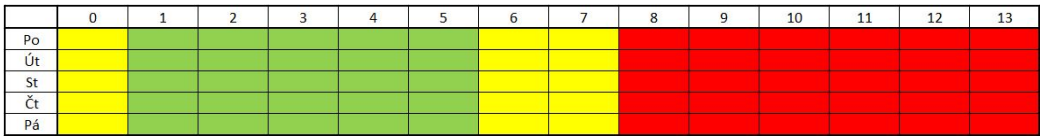
Mezi ostatní požadavky patří požadavky vyplývající ze slabých podmínek uvedených v kapitole 3.2.

4.2.1 Volné hodiny

Volné hodiny jsou jedna z nejdůležitějších slabých podmínek v generátoru. Tato podmínka se uplatňuje jak u rozvrhu učitelů, tak u rozvrhu tříd. Hlavní význam této podmínky je ten, aby učitel, nebo třída, neměli příliš mnoho volných hodin mezi vyučováním. Omezena by měla být i délka bloku volných hodin, a to ideálně na jednu hodinu – za každou další by měla být nastavena určitá penalizace.

4.2.2 Umístění hodin v rámci dne

Pokud řešíme samotné umístění vyučovací hodiny do určitého časového úseku, měli bychom se kromě již zmíněných kritérií zabývat i tím, aby hlavní část výuky byla umístěna do dopoledních hodin a teprve poté by se mělo rozvrhovat na hodiny odpolední. Zhruba znázorněno je to na obrázku 4.1, kde je vidět, že nejlepší je umístit vyučovací hodiny mezi první a pátou, další možností je umístit do nulté, nebo do šesté a sedmé. Jako poslední možnost by mělo být umisťování do pozdějších vyučovacích hodin. Toto rozdělení je samozřejmě možné upravit tak, aby se například neumisťovalo do nultých vyučovacích hodin a naopak se vyžadovaly odpolední hodiny.



Obrázek 4.1: Priority obsazovaní jednotlivých vyučovacích hodin.

4.2.3 Obsazenost dnů

Další důležitý požadavek je takový, že by předměty měly být rovnoměrně rozděleny do jednotlivých dnů. To znamená, že počet hodin v jednotlivých dnech by měl být podobný a ne takový, že například v pondělí bude podstatně méně hodin než v pátek.

4.2.4 Další obecné požadavky

Mezi další obecné požadavky zařadíme ty, které jsou dány způsobem rozvrhování v aplikaci Škola OnLine.

Opakování

Rozvrh v aplikaci Škola OnLine umožňuje zadání rozvrhu s opakováním, což znamená, že se předmět nevyučuje každý týden, ale například jednou za 14 dní. Náš požadavek je proto takový, že když se dva předměty vyučují jeden

kupříkladu v lichém a druhý v sudém týdnu, byly tyto předměty rozvrhovány proti sobě.

Skupiny

V aplikaci existují i takzvané skupiny, což jsou rozdělené třídy například na němčináře, angličtináře atd. Požadavkem tedy je, aby skupiny, které mohou být rozvrhovány na stejný čas (nesdílejí navzájem žádné žáky), byly na stejný čas rozvrhovány.

Rychlost generování

Náš program má sloužit jako jednodušší alternativa k programu aSc Rozvrhy, která by se měla používat na méně komplexní rozvrhy, proto by měla mít rychlost jako jedno z hlavních kritérií. Použitý algoritmus musí vygenerovat použitelný výsledek do jedné minuty pro rozvrhy s tisícem rozvrhovaných událostí.

4.2.5 Škola OnLine

Celý program by měl v budoucnu sloužit jako součást aplikace Škola OnLine, proto bude tato aplikace taktéž krátce představena.

Představení společnosti

Představení ze stránek [9] : Společnost ŠKOLA ONLINE a. s. působí na trhu informačních systémů a technologií (IS/IT) samostatně od roku 2008, kdy vznikla odštěpením od CCA Group a.s. V rámci této firmy fungovala coby samostatná divize Škola OnLine od roku 2001. Společnost tedy má za sebou bohaté zkušenosti z oblasti vývoje, provozování i správy informačních a vzdělávacích systémů a disponuje výkonnou vývojovou a konzultační základnou, přičemž výhradní postavení mezi jí aplikovanými technologiemi zaujímá společnost Microsoft.

Od samého počátku ŠKOLA ONLINE a. s. zajišťovala vytvoření a provoz

informačního systému Škola OnLine, který je dnes komplexním informačním systémem pro školy, pokrývající jak administrativní, tak vzdělávací potřeby každé školy.

ŠKOLA ONLINE a. s. je 100% vlastněna českým kapitálem se zcela průhlednými vlastnickými vztahy. Z pohledu lidských zdrojů pracuje více než 85 % zaměstnanců oblast vývoje a odborného poradenství.

V uplynulých letech realizovala ŠKOLA ONLINE také úspěšně více než dvacet zakázek financovaných z Evropského sociálního fondu pro různé školy v České republice. Hlavní náplní těchto projektů bylo vytvoření a provozování výukových systémů pro podporu e–learningových aktivit na školách a poskytování školení pedagogů i žáků škol.

Rozvrhování dnes

Pro generování rozvrhů se nyní se v aplikaci používá program aSc Rozvrhy. Navrhovaný program nemá za cíl konkurovat tomuto produktu, pouze nabídnout jednodušší alternativu, jelikož pro některé případy (malé školy) je tento program příliš komplexní a složitý. Pro tyto školy může být také nezanedbatelnou položkou v rozpočtu zakoupení licence na tento produkt, kterou by uvažovaný generátor mohl ušetřit.

5 Návrh algoritmu

V této kapitole bude popsán princip algoritmu, který byl navržen na základě poznatků z průzkumu v části 3.3.

5.1 Navržený algoritmus

V kapitole 3.3 je uveden přehled několika základních algoritmů, které přicházely v úvahu při návrhu algoritmu pro generování rozvrhu. Tento algoritmus vychází z heuristických algoritmů, což znamená, že vytvoří velice rychle odpověď, která ovšem nemusí být optimální. Algoritmus je ovšem mírně upraven tak, aby jeho řešení byla co možná nejlepší.

Heuristický algoritmus byl vybrán hlavně z důvodu jeho rychlosti a možnosti generování pseudonáhodných rozvrhů. To znamená, že pokud spustíme generování dvakrát po sobě se stejnými daty, ale jiným počátečním nastavením náhodného generátoru, neměli bychom dostat stejné výsledky. Další důvod zvolení byl ten, že algoritmus lze lehce upravovat a optimalizovat podle různých požadavků.

V dalším textu bude používán termín položka, který představuje jednu rozvrhovou akci definovanou předmětem, třídou, učitelem a místností, v níž tato akce probíhá.

5.2 Princip algoritmu

Samotný algoritmus je velice jednoduchý a funguje v několika krocích.

1. seřazení načtených položek,
2. náhodné vybrání položky,
3. načtení rozvrhů,
4. ohodnocení volných časů,
5. umístění položky.

Kroky 1–4 jsou znázorněny ve vývojovém diagramu na obrázku 5.2, bod 5 je na obrázku 5.3. Podrobnější popis celého algoritmu je v následujícím textu.

Seřazení načtených položek

První optimalizací, kterou je nutné provést, je seřazení načtených položek podle jistých kritérií. Zvolili jsme takový přístup, že nejtěžší položky (ty, které mají nejvíce hodin, nejvíce učitelů a tříd) by se měly umisťovat jako první a až poté by měly následovat položky lehčí, u kterých se očekává větší počet možností k umístění. Seřazení tedy spočívá v tom, že se vypočte hodnota položky založená na váze délky hodiny, počtu učitelů a tříd a zda je položka opakována či ne. Na začátku by tedy měla ideálně být nejtěžší položka, která má nejvíce učitelů a tříd a je opakována. Jako poslední by poté měla být umístěna jednohodinová položka s jedním učitelem a jednou třídou. Důvod zahrnutí opakování do řazení je ten, že se na začátku snažíme ideálně umisťovat opakované položky tak, aby byly proti sobě, to znamená, že pokud je jedna liché týdny, aby proti ní byla hodina, která probíhá v sudých týdnech a podobně.

Náhodné vybrání položky

Pokud bychom chtěli deterministické generování, vybrali bychom po seřazení z kolekce první položku. V tomto případě ovšem požadujeme, abychom

dostali pro stejné vstupy pokaždé trochu jiné výsledky. Proto je nutné položku, která se bude dále zpracovávat, vybírat náhodně. Náhodný výběr je zde implementován jako Poissonovo rozdělení, ve kterém je pravděpodobnost výběru první položky 60 % a další položky jsou poté odstupňovány poměrově k počtu položek v kolekci. Může se tedy stát, že jako první bude vybrána i jedna z položek z konce kolekce, pravděpodobnost takovéto situace je však malá.

Načtení rozvrhů

Po výběru položky je nutné zjistit, jaké časové úseky jsou pro tuto položku k dispozici. To zjistíme tak, že načteme rozvrhy všech entit, které se účastní této akce (jedná se o třídy, učitele a místnosti). Tyto rozvrhy poté spojíme do jednoho (naskládáme je na sebe) a dále uvažujeme jenom ta místa, kde po tomto spojení vzniknou volné hodiny. Musíme však uvažovat i možná opakování. Pokud se tedy umisťovaná položka opakuje a načtená také, je místo, kde je načtená položka umístěna, k dispozici pro umisťování.

Ohodnocení volných časů

Nyní již víme, které hodiny jsou vhodné pro umístění (tzn. není na ně naplánována jiná výuka). Z těchto hodin nyní musíme vybrat tu nejlepší možnou a na ni naši položku naplánovat. Rozhodnutí o tom, který čas je nejvhodnější, se děje na základě ohodnocení. Postup při ohodnocování je znázorněn na obrázku 5.2.

Dále jsou uvedeny typy ohodnocení a konkrétní hodnoty, které se přiřazují položkám. Tyto hodnoty lze samozřejmě měnit v konfiguraci, a výrazně tím ovlivnit podobu výsledného rozvrhu.

čas hodiny
: jedná se o ohodnocení podle obrázku 4.1. Konkrétní ohodnocení, které bylo dosaženo testováním, je znázorněno na obrázku 5.1. Každé volné hodině je na začátku nastavena jedna z příslušných hodnot.

volné hodiny před příslušnou hodinou
: toto ohodnocení znázorňuje to, že se objevuje snaha, aby se mezi hodinami nevyskytovaly žádné, nebo pouze jednohodinové mezery. Toho je

dosaženo tím, že za každou volnou hodinu (před tou ohodnocovanou) dáváme pokutu deset bodů. Pokud je navíc volná hodina oddělena již obsazenou (to znamená, že by byla výuka, následovala volná hodina, poté opět výuka, následně volná hodina a zase výuka), pokuta se násobí, abychom se těmto situacím vyhnuli.

vhodnost opakování
jak již bylo zmíněno, některé položky se nemusí vyučovat každý týden, ale mohou být tzv. opakované, tedy že se učí například jen jednou za 14 dní. Při jejich umist'ování se snažíme, aby se tyto položky překrývaly. Toho se snažíme docílit tím, že za každé vyučování položky (algoritmus se dívá dopředu na 30 týdnů z důvodu společných násobků – maximální délka opakování je jednou za 6 týdnů) se počítá bonus 10 bodů za každé opakování. Pokud tedy spojíme dvě položky, z nichž jedna bude v lichém a druhá v sudém týdnu, dostane hodina, ve které se toto spojení uskuteční, bonus 300 bodů.

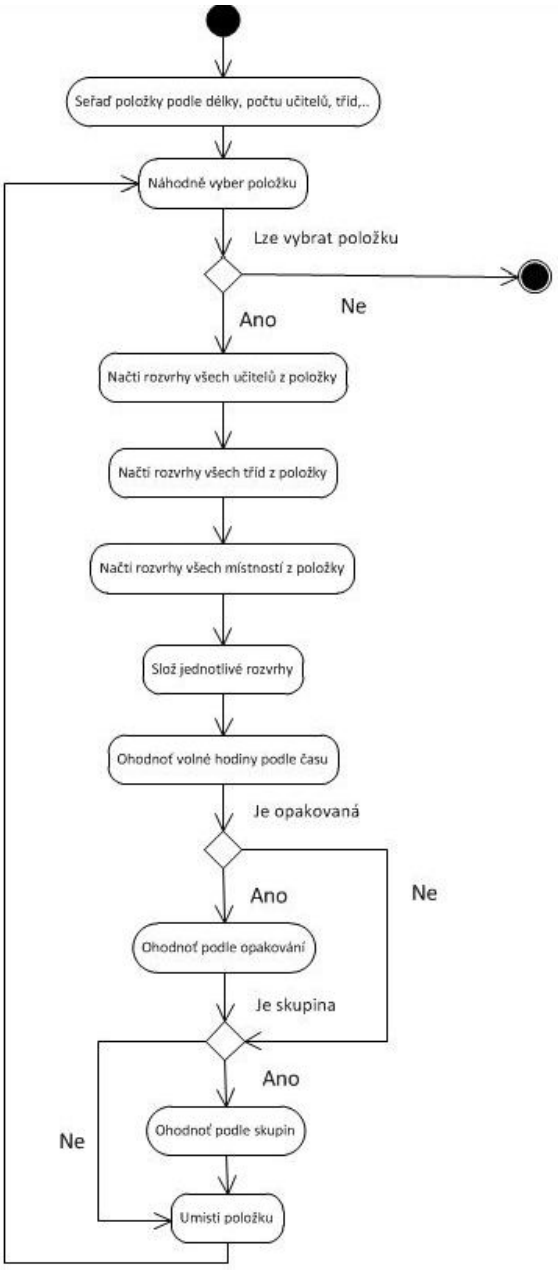
vhodnost skupiny
položky mohou obsahovat i skupiny, což znamená, že třída se této akce neúčastní celá, ale pouze její část. Tyto skupiny se také snažíme umist'ovat proti sobě, tedy pokud je třída rozdělena například na předmět německý jazyk a anglický jazyk, měly by se tyto dva předměty v ideálním případě vyučovat ve stejný čas. Toho se snažíme docílit přičtením 40 bodů k ohodnocením hodiny. Tento bonus se uděluje v případě, že je na stejný čas, na který chceme umístit příslušnou skupinu, umístěna i jiná skupina ze stejné třídy.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
Po	-50	125	100	75	50	25	-18	-21	-80	-90	-100	-110	-120	-130
Út	-50	125	100	75	50	25	-18	-21	-80	-90	-100	-110	-120	-130
St	-50	125	100	75	50	25	-18	-21	-80	-90	-100	-110	-120	-130
Čt	-50	125	100	75	50	25	-18	-21	-80	-90	-100	-110	-120	-130
Pá	-50	125	100	75	50	25	-18	-21	-80	-90	-100	-110	-120	-130

Obrázek 5.1: Rozvrh znázorňující konkrétní ohodnocení jednotlivých hodin podle času.

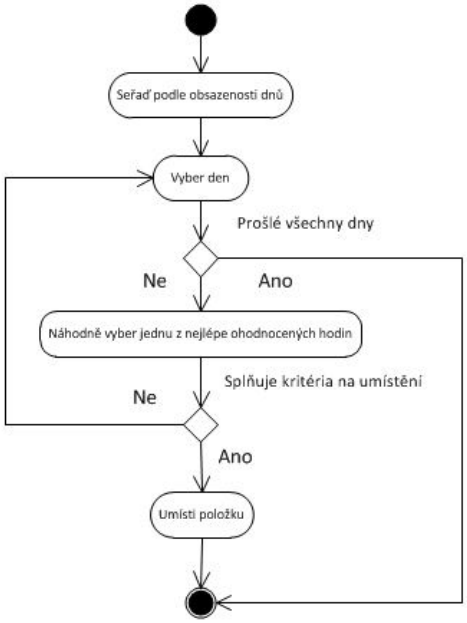
Umístění položky

Pokud již máme položky ohodnocené, snažíme se vybrat nejlepší možné umístění pro danou hodinu. Umístění hledáme tak, že si seřadíme jednotlivé dny



Obrázek 5.2: Diagram znázorňující postup při ohodnocování položek.

podle počtu hodin, které jsou v tomto dnu umístěny a začneme obsazovat od nejméně obsazeného dne. V tomto dni procházíme hodiny a podle ohodnocení vybíráme ty nejlepší. Opět zde chceme mít jistou míru náhodnosti a tak se může stát, že nebude vybrána nejlépe ohodnocená hodina, nýbrž některá jiná s mírně horším ohodnocením. To nám dává spolu s dalšími náhodně generovanými postupy z předchozích částí velkou škálu možných řešení, jež se ovšem pořád pohybují v určitých zadaných podmínkách. Při umisťování se dále uvažuje kolikátý pokus o umístění se provádí, jelikož během prvních pokusů se snažíme umisťovat jen do dopoledních hodin, až pokud na nějaký další pokus není možné umístění do dopoledne, umístí se hodina do odpoledních, či podvečerních hodin. Tento postup je znázorněn na obrázku 5.3



Obrázek 5.3: Diagram znázorňující postup při umístění položky.

Optimalizace

Jelikož nám generátor nyní vytváří několik náhodných řešení, která ovšem nemusejí být vždy optimální, bylo nakonec do programu přidáno generování několika variant a poté vybrání nejlepší z nich. Toto generování se děje paralelně a výsledná řešení se pak ohodnotí podle několika kritérií, kterými jsou

obsazenost dopoledních hodin, překrytí opakování a celkový počet umístěných hodin. Podle nich se vybere optimální řešení a to se zobrazí uživateli. Počet generovaných variant je omezen pouze paměťovými možnostmi serveru či PC a může se pohybovat až ve statisících variantách v závislosti na počtu rozvrhovaných položek.

Další zvolená optimalizace, po proběhnuvším heuristickém algoritmu, je založena na principu druhé šance. Jedná se o to, že při umístění položky si pamatujeme ohodnocení, které její aktuální umístění má. Poté, pokud nám zbudou nějaké neumístěné položky, odstraníme z rozvrhu několik z těch, jež byly umístěny s nejhorším ohodnocením, a jejich odstranění by nám mohlo pomoci při umisťování položek (tzn. mají stejné učitele, třídy, nebo místnosti) a zkusíme neumístěné položky umístit znovu. Pokud se nám to povede s lepším výsledkem, než které měly předcházející položky, necháme toto umístění, pokud dosahujeme horších výsledků, vrátíme zpět položky původní.

Algoritmus je také navržen tak, aby mohl generovat řešení ze dvou počátečních stavů. Jedním stavem je úplně prázdný rozvrh a jeho následné vytvoření. Druhým je poté doplnění položek do stávajícího rozvrhu. Druhá varianta je výhodná, pokud například učitel zná časy a dny, kdy chce, případně může, učit, a tak si tyto položky umístí předem. Poté se dogeneruje zbytek rozvrhu.

6 Implementace

Postup, který byl představen v předešlé kapitole, byl naimplementován v platformě .NET. Tato implementace bude popsána v následující kapitole. V první části bude představen systém Škola OnLine, jehož součástí by měla v budoucnu naše aplikace být. Dále bude popis zaměřen na aplikaci z hlediska architektury a použitých technologií. V poslední části bude program komentován po jednotlivých namespace a k důležitým třídám z těchto namespace bude uveden jejich základní popis a dále i popis a význam klíčových metod, které tato třída využívá.

6.1 Data v aplikaci Škola OnLine

Celý program by měl v budoucnu sloužit jako součást aplikace Škola OnLine, proto se v této části seznámíme s datovým modelem, který používá.

Reprezentace dat

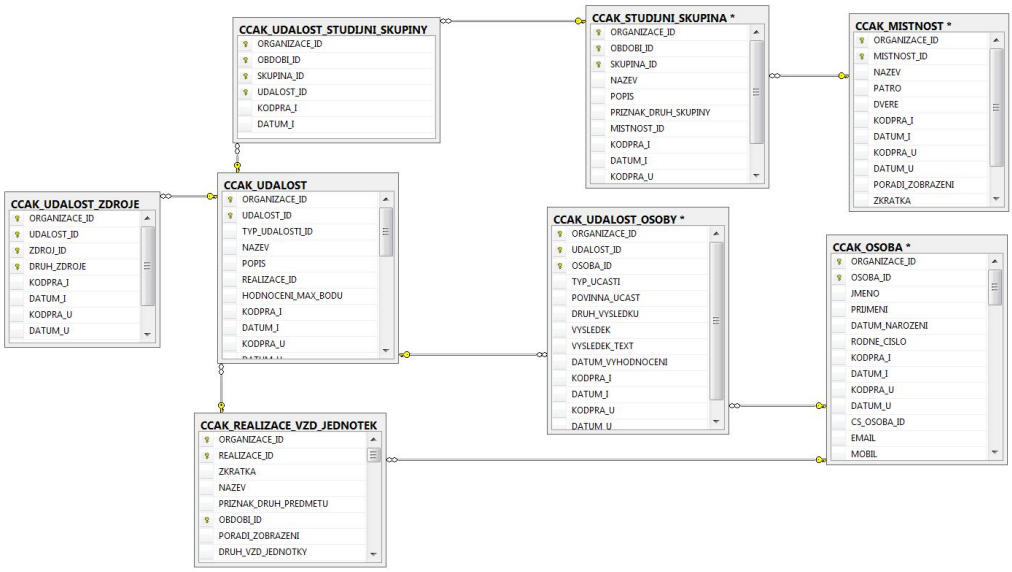
Data v aplikaci Škola OnLine jsou uložena v MS SQL databázi. Z uložených dat jsou pro nás d§ležité jen některé tabulky, konkrétně se jedná o následující (ERA model pro lepší přehled je na obrázku 6.1):

`CCAK_UDALOST`
: nejdůležitější tabulka, která obsahuje záznamy o událostech. Pro nás jsou z této tabulky nejdůležitější události typu ROZVRH, se kterými dále pracujeme. Výběr se ještě omezuje na organizace, období a verze

`CCAK_UDALOST_ZDROJE`
: informace o zdrojích u události

`CCAK_STUDIJNI_SKUPINA`
: tabulky s informací o studijní skupině. Jsou zde jak informace o třídách, tak i o skupinách

`CCAK_UDALOST_STUDIJNI_SKUPINY`
: rozkladová tabulka přiřazující studijní skupiny k události



Obrázek 6.1: Relevantní část ERA diagramu systému Škola OnLine.

`CCAK_MISTNOST`
: tabulka s místnostmi

`CCAK_OSOBA`
: kompletní informace o osobách, jako jsou jméno, příjmení, bydliště, atd.

`CCAK_UDALOST_OSOBY`
: přiřazení osob k události, jedná se především o učitele, jednotliví žáci se nepřiřazují

`CCAK_RELIZACE_VZD_JEDNOTEK`
: podrobnější informace o předmětu, jeho název, zkratka, ...

To, co nás nejvíce zajímá, je sloupec `XML_INFO` z tabulky `CCAK_UDALOST`, jeho podoba je k vidění v následujícím výpisu.

```
<T_CASOVE_UDAJE>
    <HODIN_TYDNE>2</HODIN_TYDNE>
    <DELKA_POCET_HODIN>1</DELKA_POCET_HODIN>
    <CYKLUS>-----</CYKLUS>
    <CYKLUS_POCET>1</CYKLUS_POCET>
    <DATUM_OD>2007-02-01T00:00:00+01:00</DATUM_OD>
    <DATUM_DO>2007-06-30T00:00:00+02:00</DATUM_DO>
```

```
  <PRESPOCET>0</PRESPOCET>
</T_CASOVE_UDAJE>
<T_ROZVRH>
  <DEN>5</DEN>
  <OBDOBI_DNE_ID>1</OBDOBI_DNE_ID>
  <MISTNOSTI_ID>#PKS101#</MISTNOSTI_ID>
</T_ROZVRH>
<T_ROZVRH>
  <DEN>5</DEN>
  <OBDOBI_DNE_ID>2</OBDOBI_DNE_ID>
  <MISTNOSTI_ID>#PKS101#</MISTNOSTI_ID>
</T_ROZVRH>
```

Význam jednotlivých částí je takový, že v elementu `T_CASOVE_UDAJE` jsou základní informace, jako kolik hodin týdně se hodina vyučuje, jaký je cyklus opakování atd. V elementu `T_ROZVRH` je potom údaj o samotném rozvrhování této události do některé místnosti a přiřazení ke dni a k času kdy se realizuje. Doplnění elementu `T_ROZVRH` je úkolem našeho programu. S dalšími elementy se v tomto programu nepracuje.

Jmenné konvence

Poněvadž má tento program být součástí systému Škola OnLine, je zapotřebí dodržovat jmenné konvence. Pravidla pro konvence jsou kompletně zkopírována z interních stránek společnosti CCA [3].

Obecná pravidla

- Vše se pojmenovává anglicky/česky podle následujících stanovených pravidel:
  - Všechny metody, atributy, parametry, třídy atd. jsou pojmenovány anglicky dle slovníku běžně používaných výrazů (Get, Set, Load,..). Pokud slovo v názvu není a nepatří do slovníku, je možné je zapsat česky.
  - Názvy odvozené z datového modelu (názvy sloupců, tabulek) jsou zapisovány stejně, jak jsou uvedeny v datovém modelu (převážně česky).

  - Špatný příklad: `osobaManager.vratOsobu`
  - Správný příklad: `osobaManager.findOsoba / getOsoba`

- Nepoužívají se zkratky ani zkracování
  - Názvy se nezkracují, zhoršuje se tím čitelnost kódu.
  - Zkratky (acronym) je povoleno používat pouze obecně známé a jednoznačné (DPH, ICO, http, apod.), není dovoleno používat nejasné a obecně víceznačné (OS)
  - Zkracování (abbreviation) je výslovně zakázáno (nikdo nebude luštit, co je např. `vrOsPodlPohl`).
- Nepoužívají se identifikátory pouze z velkých písmen oddělených podtržítky
- Neprefixují se lokální atributy – nepoužívá se nic takového: `mJmenoOsoby, _jmenoOsoby;` nýbrž vždy normálně `jmenoOsoby`
- Neprefixují se nikdy proměnné podle typu – žádná maďarská notace, nikdy se nepoužívá nic takového: `sJmeno, iPocet, hOdkaz`

Rozdělení, co se nazývá *camelCase* a co *PascalCase* metod a privátní atributy tříd.

- *camelCase* (začíná malým písmenem)
  - jedná se o lokální proměnné či konstanty, parametry
- *PascalCase* (začíná velkým písmenem)
  - namespace (`System.Collections.Generic`)
  - názvy typů: třídy, struktury, interface, enumu (`FontStyle, DateTime, IBusinessService, ErrorLevel`)
  - název metody (`RegisterClientScriptBlock`)
  - veřejná konstanta (`SecondsInAWeek`)
  - veřejná vlastnost
  - protected vlastnost (`DataSource`)
  - hodnoty enumu (v enumu nepoužívat nikdy ALLCAPS)

- *ALLCAPS_CASE*
  - tento způsob pojmenovávání není v prostředí .NETu používán (a to ani pro názvy konstant ani pro hodnoty enumu)

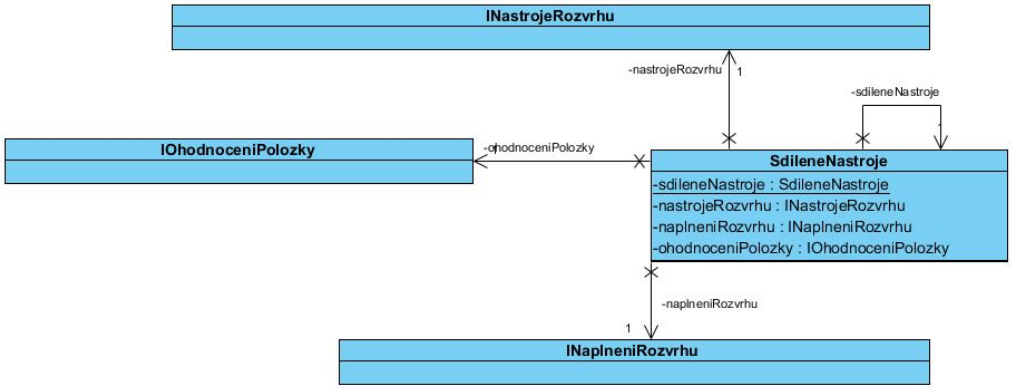
Konec převzatého textu.

6.2 Architektura a technologie

V této části si bude popsána implementovaná architektura programu. Dále zde budou uvedeny i zajímavé technologie, které byly při implementaci použity. Jedná se například o paralelní knihovny TPL, nebo LINQ.

Jádro

Základní kostra programu je navržena podle návrhového vzoru `Singleton`. Tento návrhový vzor zaručuje, že po celou dobu běhu programu bude existovat pouze jedna instance námi požadované třídy. Více informací nejen o tomto, nýbrž i dalších návrhových vzorech, jsou uvedeny v [11]. Tato kostra poskytuje třídy implementující rozhraní, která se starají o generování rozvrhu. Jedná se o naplnění rozvrhu, ohodnocení položek a různých nástrojů (vypočtení opakování, nalezení skupin, ...). Tyto nástroje jsou poté přístupné v celé aplikaci přes volání statické metody `GetInstance()` v singletonu. Tento přístup byl zvolen proto, neboť program obsahuje množství metod, které je potřeba volat na mnoha místech aplikace, a tak by bylo zbytečné vytvářet vždy nové instance. To, že `Singleton` nenabízí přímo konkrétní instance, ale jen rozhraní, je z důvodu lepší nahraditelnosti kódu, kdy nyní můžeme snadněji zaměnit jednotlivé třídy. Znázornění `Singletonu` nabízejícího jednotlivé třídy je na obrázku 6.2.



Obrázek 6.2: Singleton v aplikaci.

Přístup k datům

Pro přístup k datům slouží v našem programu DAO objekty, které pracují nad MS SQL databází. DAO je stejně jako ostatní třídy používán skrze rozhraní, které implementuje, což zaručuje budoucí snadný přechod na jinou databázi.

Přímý přístup k datům je poté zajištěn přes EntityFramework a k sestavování dotazů je použito LINQ, což je integrovaný jazyk pro dotazování. Samotné dotazy jsou poté reprezentovány přes lambda výrazy, které výrazně usnadňují a zpřehledňují celkový zápis dotazů. Příklad jednoduchého lambda výrazu:

```
public static void LambdaVyraz()
  {
      List<int> cisla = new List<int>{1,2,3,4,5,6,7};
      var vysledek = cisla.Where(n => n % 2 == 0);
      var vysledek2 = cisla.FindAll((int n) =>
                              {
                                    return n % 2 == 0;
                              });
  }
```

kde se v obou případech vrátí stejná čísla (jen sudá). Důležitý rozdíl mezi zápisy, kromě použité funkce, je ten, že ve druhém případě (vysledek2) definujeme explicitně návratový typ. Více o této technologii lze nalézt v [14].

Paralelismus

V aplikaci je využito paralelní zpracování, které zlepšuje a urychluje výsledky algoritmu. V programu nejsou žádné kritické sekce, a tak se zparalelizováním aplikace nebyl problém. Paralelismus je proveden pomocí instrukce `Parallel.ForEach` z knihovny TPL, více informací o těchto paralelismech je sepsáno v [16].

Logování

Pro lepší kontrolu fungování programu je v programu použito logování. Konkrétní použitá technologie je knihovna `log4net`, která je rozšířením frameworku `log4j` od Apache pro framework .NET. Více informací o této technologii je možné nalézt na oficiálních stránkách viz [10].

ReSharper

Pro ještě větší kvalitu kódu byl při vývoji používán program Resharper od společnosti JetBrains. Tento doplněk do Visual Studia zajišťuje kontrolu a odstranění hlavních problémů v kódu, jako jsou špatné komentáře, nepoužité privátní metody, nebo složitý zápis kódu. V neposlední řadě kontroluje i formátování celého kódu podle přednastavených parametrů (otevírací závorka na vlastní řádce, ...).

6.3 Rozdělení programu

Program je rozdělen do několika základních namespace, které jsou umístěny v kořenovém namespace `cz.cca.SOL.RozvrhDP`. Jednotlivé namespace jsou popsány v následujícím textu. Jedná se jednak o jádro aplikace, různé podpůrné třídy a v neposlední řadě o reprezentaci dat. Nadpisy uvedené v následujícím textu reprezentují názvy jednotlivých namespace a dále je k nim uveden základní popis. Většina tříd implementuje rozhraní pro jejich pozdější snažší nahraditelnost, tato rozhraní jsou umístěna v namespace Interface, jednotlivá rozhraní budou uvedena společně se třídami, které je implementují.

Model

V tomto namespace jsou umístěny třídy, které reprezentují nějaká data. Pro reprezentaci dat, která jsou načítána z aplikace Škola OnLine, byly navrženy následující entity:

`Trida`
: reprezentuje jednu třídu, nebo skupinu. Obsahuje hlavní atributy název, ID, ID rodičovské třídy, dělení

`Mistnost`
: reprezentuje jednu místnost. Obsahuje hlavní atributy název, ID

`Ucitel`
: reprezentuje jednoho učitele. Obsahuje hlavní atributy jméno, příjmení, ID

Dále bylo nutno navrhnout reprezentaci dat samotného rozvrhu. Pro tento účel jsou navrženy následující třídy:

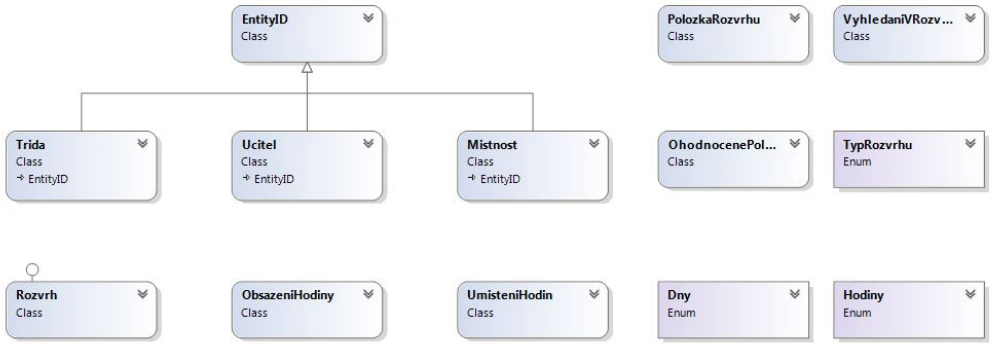
`ObsazeniHodiny`
: obsahuje základní informace o jedné hodině, která obsahuje údaj o předmětu, který se v této hodině vyučuje. Je zde uveden název předmětu, opakování, obsazenost a ohodnocení

`Rozvrh`
: celý rozvrh třídy, učitele, nebo místnosti. Jedná se o třídu, jež obsahuje 5 polí instancí třídy `ObsazeniHodiny`, kde každé pole reprezentuje jeden vyučovací den a každé políčko v poli jednu vyučovací hodinu

`PolozkaRozvrhu`
: reprezentace položek načtených z databáze. Obsahuje kompletní informace o všech předmětech, které se budou rozvrhovat

`UmisteniHodin`
: reprezentuje konkrétní umístění hodin, jež se generuje. Je součástí třídy `PolozkaRozvrhu`

V tomto namespace se nachází i výčtové typy, které reprezentují `Dny, Hodiny` a `TypRozvrhu` (rozvrh učitele, třídy, místnosti).



Obrázek 6.3: Přehled entit a výčtů v programu.

Algoritmus

V tomto namespace jsou umístěny nejdůležitější třídy programu. Jedná se vlastně o jádro celé aplikace, které bylo popsáno v 6.2. Jelikož jsou třídy z tohoto namespace velice důležité, popíšeme si jejich třídy a základní metody z rozhraní (které každá třída implementuje) podrobněji. V tomto namespace je umístěn další namespace s názvem `Common`, kde jsou umístěny sdílené třídy. Jedná se o `SdileneNastroje, Extensions a GenericCopier`.

SdileneNastroje

Třída je realizovaná jako Singleton. Poskytuje instance ostatních sdílených tříd, které budou popsány dále. Implementované metody jsou následující:

`static SdileneNastroje GetInstance()`
: vrací instanci jedináčka

`INastrojeRozvrhu GetNastrojeRozvrhu()`
: vrací instanci třídy NastrojeRozvrhu

`IOhodnoceniPolozky GetOhodnoceniPolozky()`
: vrací instanci třídy OhodnoceniPolozky

`INaplneniRozvrhu GetNaplneniRozvrhu()`
: vrací instanci třídy NaplneniRozvrhu

Extensions

Jedná se o třídu obsahující tzv. Extensions Method, což je vylepšení obsažené v .NET Frameworku 3.5. Extension methods nám umožňují přidat třídě instanční metodu „zvenčí“, aniž bychom museli modifikovat třídu samotnou. Více o tomto rozšíření je možné se dočíst například v [5] nebo [15]. V našem případě používáme následující rozšiřující metody:

`static IList<PolozkaRozvrhu> SeradPolozky(this IList<PolozkaRozvrhu> rozvrh)`
seřadí položky v seznamu podle pravděpodobnosti vybrání položky

`static IList<IList<ObsazeniHodiny> > SeradPodleHodnoceni(this IList <IList<ObsazeniHodiny> > ohodnoceni)`
seřadí položky v seznamu podle ohodnocení

`static void SeradPodleObsazeni(this IList<ObsazeniHodiny[ ]> neserazeneObsazeni)`
seřadí dny v týdnu podle počtu hodin v jednolivých dnech

`static bool PodobnePolozky(this PolozkaRozvrhu polozka, PolozkaRozvrhu porovnavanaPolozka)`
vrací true, pokud dvě položky v parametru mají stejnou třídu, nebo učitele

GenericCopier

Jedná se o třídu, která obsahuje jen jednu metodu

```
public static T DeepCopy(object objectToCopy)
```

Používá se pro kopírování objektů v paměti. Pokud generujeme více možností, načte se jen jeden seznam, který se touto metodou nakopíruje a poté se spustí vyhodnocení pro každou kopii zvlášť.

NastrojeRozvrhu

Třída se základními nástroji pro práci s rozvrhem. Implementuje rozhraní INastrojeRozvrhu, které má následující metody:

`bool MuzeSOpakovanim(IList<bool[ ]> opakovani)`
jedná se o metodu, která zjistí, zda je možné umístit pole s opakováním, která jsou předána jako parametr, na stejnou hodinu

`void SetMistnosti(IList<Mistnost> mistnosti)`
při generování potřebujeme mít k dispozici seznam místností, se kterým můžeme pracovat (pokud se nenajde volná místnost, vybere se náhodně jiná z tohoto seznamu), přes tento setter se tento seznam nastaví

`bool ObsahujeID(string ID, IList<EntityID> kolekce)`
metoda, která prozkoumá, zda předaná kolekce obsahuje ID, pokud se jedná o třídu, zkontroluje ještě ID rodiče

`Rozvrh SpojRozvrhy(IList<Rozvrh> rozvrhy)`
metoda, která spojí rozvrhy zadané v parametru do jednoho rozvrhu a ten vrátí. Důležitá metoda pro hledání volných míst v rozvrhu

`int VhodnostOpakovani(IList<bool[ ]> opakovani)`
metoda, která zjistí jak vhodné je opakování; vrací počet „vyplněných“ polí při spojení polí předaných v parametru

`IList<Rozvrh> NactiRozvrhyPolozky(PolozkaRozvrhu polozka, int pokus, ref string mistnostPouzita, IList<PolozkaRozvrhu> polozky)`
pro danou položku načte rozvrhy všech jejích účastníků (tříd, učitelů a místností)

`int GetRandomCisloHodiny(int pocetPrvku)`
náhodně vygeneruje a vrátí číslo podle exponenciálního rozdělení. S pravděpodobností 60 % se vrátí index 0, další pravděpodobnosti záleží na počtu prvků.

`int SpoctiPravdepodobnostPolozky(PolozkaRozvrhu polozka)`
vypočte s jakou pravděpodobností bude vybrána položka jako první – uvažuje se délka hodiny, počet účastníků a zda se jedná o skupinu či nikoli.

NaplneniRozvrhu

Třída zajišťující naplnění rozvrhů. Implementuje rozhraní `INaplneniRozvrhu`, které obsahuje následující metody:

`Rozvrh NactiRozvrhMistnosti(string id, IList<PolozkaRozvrhu> polozky)`
načte rozvrh místnosti podle ID.

`Rozvrh NactiRozvrhTridy(string id, IList<PolozkaRozvrhu> polozky)`
načte rozvrh třídy podle ID

`Rozvrh NactiRozvrhSkupiny(Trida trida, IList<PolozkaRozvrhu> polozky)`
načte rozvrh skupiny podle ID

`Rozvrh NactiRozvrhUcitele(string id, IList<PolozkaRozvrhu> polozky)`
načte rozvrh učitele podle ID

`IList<Rozvrh> GetRozvrhySkupin(string tridaParentID, int? deleniID, IList<PolozkaRozvrhu> polozky)`
načte rozvrhy všech skupin podle ID rodičovské třídy a dělení.

OhodnoceniPolozky

Třída obsahující metody pro ohodnocení vhodnosti jednotlivých časových úseků pro umístění položky. Implementuje rozhraní `IOhodnoceníPolozky`, které obsahuje následující metody:

`void OhodnotVysledny Rozvrh(OhodnocenePolozkyRozvrhu ohodnocenePolozky)`

metoda sloužící pro finální určení kvality rozvrhu. Hodnotí se, kolik se nepodařilo umístit položek a jak jsou splněna další kritéria (výuka dopoledne, minimum volných hodin,...)

`void VypoctiOhodnoceni(Rozvrh rozvrhOhodnoceny, Rozvrh rozvrhUmisteny)`
ohodnotí celý rozvrh voláním dále uvedených metod

`void OhodnotRozvrhPodleCasu(Rozvrh rozvrh)`
ohodnotí rozvrh podle času – dopoledne lepší ohodnocení

`void OhodnotRozvrhPodleOpakovani(Rozvrh rozvrh, bool[ ] opakovaniPolozky)`
ohodnotí rozvrh podle opakování

`int PocetVolnychHodinPred(int hodina, ObsazeniHodiny[ ] obsazeni)`
vrátí počet volných hodin před aktuální hodinou

`void OhodnotSkupiny(Rozvrh ohodnocovany, Rozvrh skupina)`
přiřadí rozvrhu ohodnocení podle rozvrhu skupiny

GenerovaniRozvrhu

Třída pro samotné generování rozvrhu. Implementuje rozhraní `IGenerovaniRozvrhu`, které obsahuje pouze jedinou metodu, a to metodu

```
void Generuj(IList<PolozkaRozvrhu> polozky);
```

která spustí celé generování rozvrhu. Výsledky generování se ukládají přímo do předávaného seznamu položek.

UmisteniPolozky

Tato třída zajišťuje umístění položky do nějaké volné hodiny podle ohodnocení. Má jedinou public metodu, a to

```
bool UmistiPolozku (Rozvrh spojenyRozvrh, PolozkaRozvrhu polozka,
    string pouzitaMistnost, int poradi, ref boo jednaUmistena)
```

která má jako parametry spojený rozvrh všech entit v události, umisťovanou položku, místnost, která byla položce přiřazena, pořadí, kolikátá položka je umisťována, a nakonec informaci, zda se podařilo položku umístit, či nikoli.

DataSets

Přestože tato aplikace využívá především možnosti EntityFramework, pro načítání XML informací z tabulky `CCAK_UDALOST` je používán dataset. Tento přístup byl zvolen proto, nebot' je tato technika použita v systému Škola OnLine a pro načtení malých dat je naprosto vyhovující. Ukázka těchto načítaných dat je v části 6.1, kde dataset obsahuje dvě tabulky, jejichž sloupce odpovídají elementům ze zmíněné ukázky.

EntityConverter

V tomto namespace se nachází třída `Converter`, která slouží k převedení dat načtených z databáze do entit, se kterými dále pracuje příslušný program. Třída `Converter` implementuje rozhraní `IConverter`, jež obsahuje následující metody:

`IDAO dao { get; set; }`
: nastavení instance `DAO`, které tato třída dále využívá k získání dat z databáze

`IList<PolozkaRozvrhu> ModelToEntity(IList<CCAK_UDALOST> udalosti)`
: převede seznam typu `CCAK_UDALOST` na list typu `PolozkaRozvrhu`, s nímž se dále pracuje v programu.

`IList<Mistnost> MistnostiEntityToModel()`
: převede seznam místostí z DB na seznam typu `Mistnost`, data získává z `DAO` objektu

`IList<Trida> TridyEntityToModel()`
: převede seznam tříd z DB na seznam typu `Trida`, data získává z `DAO` objektu

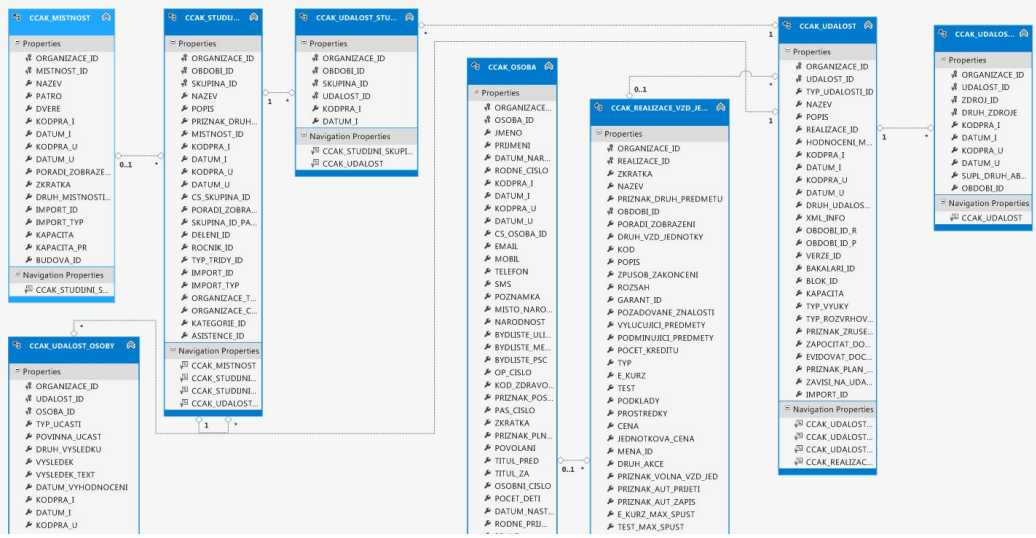
`IList<Ucitel> UciteleEntityToModel()`
: převede seznam učitelů z DB na seznam typu `Ucitel`, data načítá z `DAO` objektu

`CCAK_UDALOST UdalostiEntityToModel(PolozkaRozvrhu polozka, string obdobiID_R, string obdobiID_P, string organizaceID)`
: převede položku typu `PolozkaRozvrhu` na typ CCAK_UDALOST, zároveň je možné i změnit období a organizaci, které se mají nastavit.

EntityFramework

UdalostEntityModel

Jedná se o model databáze, k němuž poté přistupujeme přes Container z `DAO` objektu. Model je na obrázku 6.4.



Obrázek 6.4: Model databáze Entity Frameworku. V modelu jsou ořízlé některé tabulky kvůli velkému počtu sloupců.

DAO

`DAO` objekt je použit pro přístup k datům. Standardní implementace, která je použita, vypadá tak, že je vytvořen Container, který zpřístupňuje data z databáze. Pro zpřístupnění je poté použit jazyk LINQ, jenž generuje příkazy select, insert a delete do databáze. `DAO` implementuje rozhraní `IDAO`, jehož základní metody jsou následující:

`IList<CCAK_UDALOST> GetUdalosti()`
: vrátí všechny události pro danou organizaci a období

`CCAK_MISTNOST GetMistnostByID(string id)`
: vrátí místnost podle ID

`CCAK_STUDIJNI_SKUPINA GetStudijniSkupinaByID(string id)`
vrátí studijní skupinu podle ID

`Ucitel GetOsobuByID(string id)`
vrátí učitele podle ID

`IList<CCAK_MISTNOST> GetMistnosti()`
vrátí seznam všech místností podle organizace

`IList<CCAK_STUDIJNI_SKUPINA> GetStudijniSkupiny()`
vrátí seznam všech studijních skupin dané organizace

`IList<Ucitel> GetOsoby()`
vrátí seznam všech učitelů dané organizace

`void SaveUdalostOsoby(CCAK_UDALOST_OSOBY udalostOsoby)`
uloží záznam do tabulky `CCAK_UDALOST_OSOBY`

`void SaveUdalostStSkupiny(CCAK_UDALOST_STUDIJNI_SKUPINY udalostStSkupiny)`
uloží záznam do odpovídající tabulky

`void SaveNewUdalost(CCAK_UDALOST udalost)`
uloží novou událost do tabulky `CCAK_UDALOST`

`void UpdateUdalosti(CCAK_UDALOST udalost)`
aktualizuje záznam v tabulce `CCAK_UDALOST` tak, že zamění sloupec `XML_INFO`

`void SaveRealizaceJednotky(CCAK_REALIZACE_VZD_JEDNOTEK realizace)`
uloží záznam do odpovídající tabulky

UlozeniRozvrhu

Třída sloužící pro uložení nového nebo aktualizovaného rozvrhu do databáze. Implementuje rozhraní `IUlozeniRozvrhu`, které má pouze jednu metodu, a to metodu

```
void UlozPolozky(IList<PolozkaRozvrhu> polozky,
    string obdobiID_R, string obdobiID_P, string organizaceID,
    string verzeID, bool novy)
```

tato metoda se na základě parametru `novy` rozhodne, zda vytvoří nový rozvrh, anebo bude aktualizovat ten stávající. Podle toho založí metodami z `DAO` objektu nové záznamy nebo aktualizuje stávající.

6.4 Ovládání aplikace

Aplikace disponuje několika způsoby, jak spustit rozvrhování, a tím získat výsledek. V dalším textu budou popsány programové možnosti tohoto spuštění a získání výsledků. Také zde budou popsány možnosti změny nastavení základních parametrů. Zprovoznění aplikace a její následné ovládání je popsáno v částech A.1 a A.2.

6.4.1 Programové volání

Pokud chceme vyvolat generování z nějakého externího programu, máme několik možností. V obou případech se jedná o volání některé metody z třídy `GeneratorStart`, která je součástí namespace `Algoritmus`.

Načtení předem

Pokud budeme mít předem k dispozici načtená data, můžeme generátor spustit voláním metody

```
static IList<PolozkaRozvrhu> GenerujRozvrh(IList<PolozkaRozvrhu>
    polozka, IList<Mistnost> mistnosti, int pocetGenerovanych)
```

která jako parametry převezme seznam načtených položek, seznam místností a počet generovaných rozvrhů.

Načtení v metodě

Další možností je to, že nemáme k dispozici předem žádná data, ale známe údaje potřebné pro jejich získání, jako je období, organizace atd. Poté je pro nás vhodnější volání druhé metody

```
static IList<PolozkaRozvrhu> GenerujRozvrh(string obdobiID,
    string obdobiID_P, string organizaceID, string verze,
    int pocetGenerovanych)
```

která přebírá více parametrů, avšak poté si vše od načtení až po vrácení výsledku zařídí sama.

6.4.2 Změna nastavení

Jak bylo vedeno v předchozím textu, tato aplikace funguje tak, že na základě vah se snaží umístit položku na to nejlepší možné místo. Tyto váhy se dají přednastavit, a tím výrazně ovlivnit podobu generovaného rozvrhu. Přednastavení se provádí tak, že se přepíše hodnota v `.resx` souboru, který je umístěn ve složce `Resources`. V základu je generátor řízen několika ohodnoceními. Chceme-li rožšířit náš generátor o další ohodnocovaní, uděláme to snadno přidáním definice tohoto ohodnocení do `.resx` souboru, přidáním metody s vykonáním samotného ohodnocení do programu a zavoláním této metody. V základu jsou použita následující ohodnocení s přednastavenými hodnotami hodnoty (hodnoty byly získány empiricky):

OhodnoceniNulta
: penalizace nebo zvýhodnění nulté hodiny – defaultní hodnota je –150

OhodnoceniVolnaPredTrida
: penalizace za volnou hodinu před příslušnou hodinou pro třídy defaultně –20

OhodnoceniVolnaPredUcitel
: penalizace za volnou hodinu před příslušnou hodinou pro učitele – defaultně –5

OhodnoceniDopoledne
: bonus za hodinu umístěnou dopoledne – defaultně +25

OhodnoceniOdpoledne
: ohodnoceni za hodinu umístěnou odpoledne – defaultně –3

OhodnoceniPodvecer
: penalizace za umístění hodiny v podvečer – defaultně –10

OhodnoceniOpakovani
bonus za kvalitu umístění opakování – defaultně +5 za obsazený týden

OhodnoceniSkupiny
bonus za umístění skupin vedle sebe – defaultně +40

OhodnoceniNeobsazenaDopol
penalizace za neobsazenou dopolední hodinu – defaultně –150

HodnoceniNeumisteno
penalizace za neumístěnou hodinu – defaultně –150

StejnyPocetHodinVTydnu
bonus za to, pokud se nám podaří mít stejný počet hodin v jednotlivých dnech v týdnu – defaultně –50

Kromě těchto hodnot, které znamenají ohodnocení, je v `.resx` souboru i několik dalších hodnot, které značí jak bude rozvrh vypadat. Jedná se o:

PocetHodin
značí, kolik maximálně hodin bude v jednom dni, zadané číslo je bez nulté hodiny – defaultně +13

PocetDopolednichHodin
značí, kolik hodin se označuje jako dopoledních, což jsou ty, které se snažíme maximálně zaplnit

PocetOdpolednichHodin
označuje, kolik hodin je takových, které nejsou stejně vhodné jako dopolední, ale pořád se vyplatí do nich umisťovat. Zbylé hodiny, které nejsou mezi dopoledními a odpoledními, se berou jako podvečerní a mají velké penalizace.

6.5 Integrace do Školy OnLine

Systém Škola OnLine je ASP.NET FORMS aplikace napsána v jazyku C#. Pro některé své funkčnosti již nyní využívá některé externí knihovny, které jsou naimportovány do projektu. Integrace tohoto generátoru rozvrhu bude podobná jako u jejich stávajících knihoven, tedy taková, že generátor bude integrován do systému jako DLL knihovna. Tato integrace není složitá, stačí

ve vývojovém prostředí Visual Studio vygenerovat DLL (Dynamic-link library). Celý generátor byl od začátku vyvíjen tak, aby bylo tuto knihovnu možné jednoduše vytvořit (rozdělení na části generátor a vizualizace). Tuto knihovnu je poté nutné připojit do projektu Škola OnLine a následně zavolat ze stránky tvorby rozvrhu.

7 Zhodnocení výsledků

7.1 Porovnání s aSc rozvrhy

Pro účely testování byla vytvořena data pro dvě školy (malou a střední) a dále byla k dispozici data z reálné školy (velmi velké), jejíž rozvrh je vygenerován programem aSc Rozvrhy. Všechny rozvrhy byly vygenerovány v navrženém programu a následně výsledky porovnány s rozvrhem vygenerovaným programem aSc Rozvrhy. Porovnávány jsou vždy tři verze navržených rozvrhů.

Z našeho generátoru se jedná o verze s deseti, padesáti a sty verzemi rozvrhu. Pro program aSc Rozvrhy jsou testovány výpočty s normální, velkou a velmi velkou kvalitou generování. Rozvrhy zobrazované na obrázcích jsou pro verze se sty verzemi z navrženého generátoru a s velmi velkou složitostí z aSc Rozvrhu, což by měly být v obou případech nejkvalitnější výsledky. Testování probíhalo na počítači s procesorem Intel Core i7.

Podrobnější popis programu aSc Rozvrhy je k dispozici v části 2.2.1.

7.1.1 Rozvrh pro malé školy

Porovnávaný rozvrh

Pro porovnání v této části byl zvolen rozvrh pro školu o málo třídách. Konkrétně se jedná o školu, která má jen první stupeň, tedy pět tříd. V těchto třídách učí většinu předmětů třídní učitel, pouze anglický jazyk má jen jeden učitel. Data pro tento rozvrh byla vytvořena ručně, inspirací byl rozvrh prvního stupně na škole 31. ZŠ Plzeň [1], který byl autorem práce mírně upraven. Rozvrh se skládá z následujících týdenních úvazků:

1. třída 10× český jazyk, 4× matematika, 1× hudební výchova, 1× pracovní činnosti, 2× prvouka, 2× tělesná výchova a 1× výtvarná výchova. Celkem tedy 21 hodin.

2. třída 10× český jazyk, 5× matematika, 1× hudební výchova, 1× pracovní činnosti, 1× výtvarná výchova, 2× tělesná výchova a 2× prvouka. Celkem 22 hodin.

3. třída 9× český jazyk, 5× matematika, 1× hudební výchova, 1× výtvarná výchova, 2× prvouka, 2× tělesná výchova, 3× anglický jazyka a 1 × dvouhodinová hodina pracovních činností. Celkem 25 hodin.

4. třída 7× český jazyk, 4× matematika, 4× anglický jazyk, 1× hudební výchova, 2× vlastivěda, 2× výtvarná výchova, 2× přírodověda, 1× dvouhodinová hodina tělesné výchovy a 1× dvouhodinová hodina pracovních činností. Celkem 26 hodin.

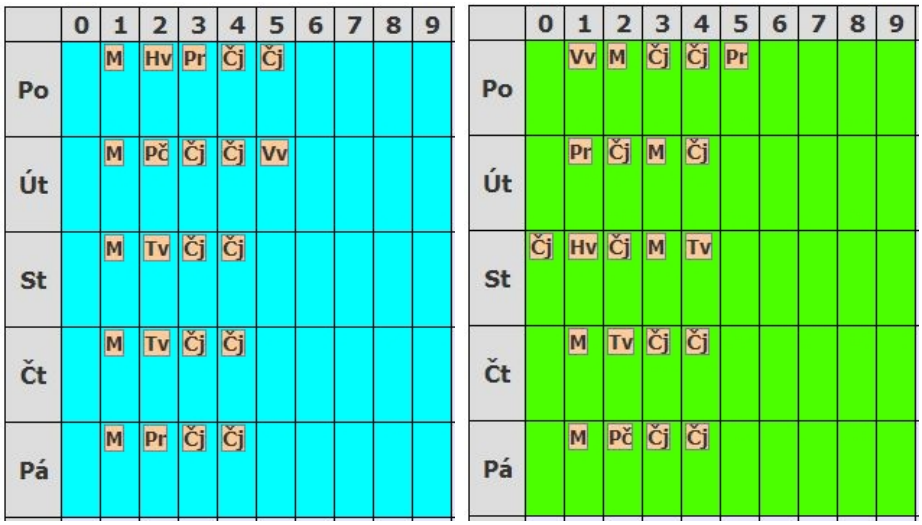
5. třída 6× český jazyk, 5× matematika, 3 × anglický jazyk, 2× vlastivěda, 2× výtvarná výchova, 1× dvouhodinová hodina tělesné výchovy, 1× dvouhodinová hodina pracovních činností, 1× dvouhodinová hodina informatiky a 1× hudební výchova. Celkem tedy 27 hodin.

Aby generátor měl komplikovanější generování, byl pro všechny hodiny anglického jazyka nastaven jako vyučující třídní učitel 5. třídy. Zkomplikování je zvoleno z toho důvodu, neboť jinak by úloha byla příliš triviální. Celkem je rozvrhováno 121 hodin.

Kvalita generování pro třídy

Kvalita je v tomto případě srovnatelná s rozvrhem generovaným programem aSc Rozvrhy. Jelikož je rozvrh velice malý, podařilo se vždy umístit všechny položky. Příklady umístění jednotlivých hodin pro třídy jsou na obrázku 7.1, 7.2, 7.3, 7.4 a 7.5. Popis k jednotlivým rozvrhům je uveden u obrázků. Na všech obrázcích je vlevo výsledek navrženého generátoru (světle modré pozadí), vpravo je výsledek generátoru aSc Rozvrhy (světle zelené pozadí).

Obrázek 7.1: Porovnání rozvrhu pro první třídu. aSc Rozvrhy zvolil nultou hodinu, zatímco navržený generátor raději umisťoval do odpoledních hodin.

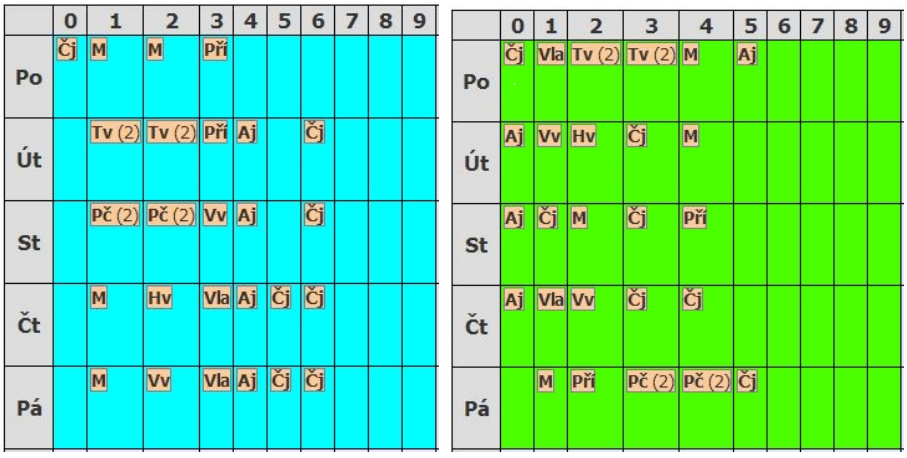


Obrázek 7.2: Porovnání rozvrhu pro druhou třídu. aSc Rozvrhy opět zvolil nultou hodinu, zatímco navržený generátor raději umisťoval do odpoledních hodin.

	0	1	2	3	4	5	6	7	8	9
Po		M	M	Pr	Aj	Aj	Aj			
Út		Tv	M	Čj	Čj	Čj				
St		Tv	M	Čj	Čj	Vv				
Čt		Hv	M	Pr	Čj	Aj				
Pá		Pč (2)	Pč (2)	Čj	Čj	Čj				

	0	1	2	3	4	5	6	7	8	9
Po	Aj	Pr	Čj	M	Čj					
Út		M	Čj	Tv	Hv	Aj				
St		Pč (2)	Pč (2)	Čj	M	Čj	Aj			
Čt		Vv	Čj	Tv	Čj	M	Aj			
Pá		M	Čj	Čj	Pr					

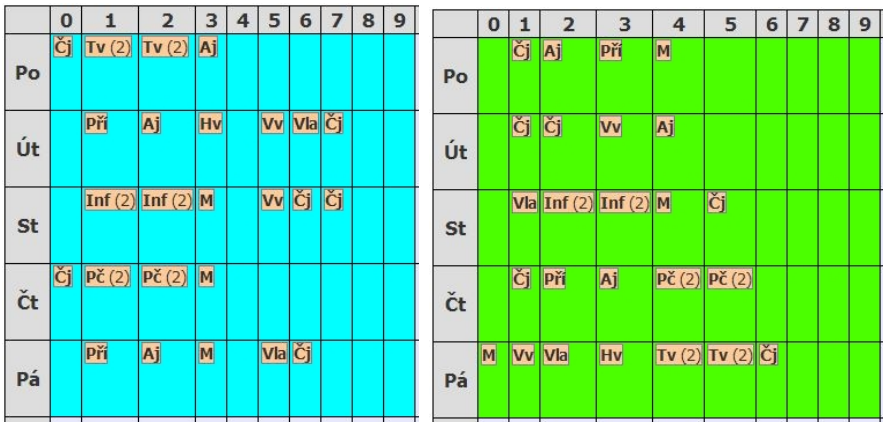
Obrázek 7.3: Porovnání rozvrhu pro třetí třídu. aSc Rozvrhy umístil výuku angličtiny (jiný učitel) do nulté hodiny, zatímco navržený generátor ji umístil do odpoledního rozvrhu.



	0	1	2	3	4	5	6	7	8	9
Po	Čj	M	M	Při						
Út		Tv (2)	Tv (2)	Při	Aj		Čj			
St		Pč (2)	Pč (2)	Vv	Aj		Čj			
Čt		M	Hv	Vla	Aj	Čj	Čj			
Pá		M	Vv	Vla	Aj	Čj	Čj			

	0	1	2	3	4	5	6	7	8	9
Po	Čj	Vla	Tv (2)	Tv (2)	M	Aj				
Út	Aj	Vv	Hv	Čj	M					
St	Aj	Čj	M	Čj	Při					
Čt	Aj	Vla	Vv	Čj	Čj					
Pá		M	Při	Pč (2)	Pč (2)	Čj				

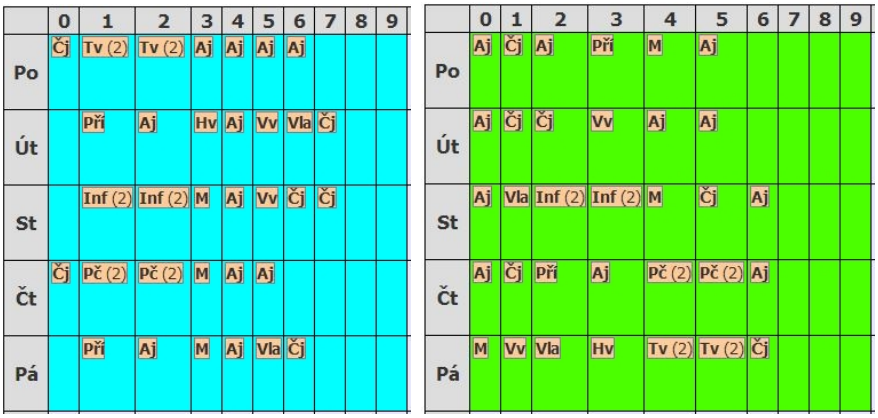
Obrázek 7.4: Porovnání rozvrhu pro čtvrtou třídu. aSc Rozvrhy využívá opět nulté hodiny, tentokrát hlavně pro anglický jazyk. Navržený generátor vygeneroval rozvrh velice dobře, zvláště rovnoměrné rozprostření po jednotlivých dnech je povedené.



Obrázek 7.5: Porovnání rozvrhu pro pátou třídu. aSc Rozvrhy i navržený generátor zvolili nultou hodinu. Jelikož učitel páté třídy učí anglický jazyk v jiných třídách, vznikly nám zde volné hodiny. Ty jsou maximálně jednohodinové, tudíž je nepovažujeme za chybu.

Kvalita generování pro učitele

Jelikož je na prvním stupni většinou rozvrhu třídy stejný jako rozvrh učitele, uvedu zde jen příklad učitele páté třídy, který učí i angličtinu i v jiných třídách a tak je jeho rozvrh obsáhlejší. Porovnání je vidět na obrázku 7.6.



Obrázek 7.6: Porovnání rozvrhu pro učitele páté třídy. Učitel dále učí v ostatních třídách anglický jazyk. Rozvrhy jsou velice podobné, aSc rozvrhy umisťoval raději do dopoledne, navržený generátor spíše do odpoledních hodin.

Rychlost generování

Rychlost generování je u takto malých rozvrhů zcela vyhovující – probíhá v rámci několika vteřin. Nejrychlejší byl navržený generátor s deseti variantami, který vygeneroval rozvrh za čtvrt vteřiny. Sto verzí poté trvalo 2 vteřiny a 23 setin. Porovnávaný program aSc Rozvrhy bohužel nemá zobrazení na setiny vteřiny, a tak jsou údaje z něho pouze orientační, ale zhruba odpovídají navrženému generátoru. Přehled časů je k dispozici v následující tabulce.

Tabulka 7.1: Porovnání rychlostí našeho generátoru a aSc Rozvrhy na jednoduchém rozvrhu. Časy generování jsou v tomto případě skoro stejné.

	aSc rozvrhy			**navržený generátor**		
Složitost	Normální	Velká	Velmi velká	10 verzí	50 verzí	100 verzí
Čas [sec]	1	1	2	0,25	0,88	2,23
Neumístěno	0	0	0	0	0	0

7.1.2 Rozvrh pro středně velké školy

Porovnávaný rozvrh

Rozvrh reprezentující středně velkou školu byl autorem vytvořen ručně v programu aSc Rozvrhy a poté importován do systému Škola OnLine. Jako středně velká škola je zde brána taková škola, která má první a druhý stupeň a každý ročník má dvě třídy. Celkově tedy máme 18 tříd. Třídy se dále dělí i na skupiny kvůli tělesné výchově. Celkem je požadováno rozvrhnout 444 hodin. Rozvrh pro třídy prvního stupně vychází z rozvrhu pro malé školy, který je představen v předchozí části. Rozvrh pro druhý stupeň obsahuje následující hodiny:

Šestá třída 4× matematika, 5× český jazyk, 3× tělesná výchova, 1× fyzika, 1× přírodopis, 2× zeměpis, 2× dějepis, 2× anglický jazyk, 2× dvouhodinová hodina informatiky, 1× pracovní činnosti, 1× výtvarná výchova, 1× hudební výchova.

Sedmá třída 4× matematika, 5× český jazyk, 3× tělesná výchova, 1× fyzika, 2× přírodopis, 2× zeměpis, 2× dějepis, 2× anglický jazyk, 2×

dvouhodinová hodina informatiky, 1× pracovní činnosti, 1× výtvarná výchova, 1× hudební výchova.

Osmá třída 4× matematika, 5× český jazyk, 3× tělesná výchova, 1× fyzika, 2× přírodopis, 2× zeměpis, 2× dějepis, 2× anglický jazyk, 2 × dvouhodinová hodina informatiky, 1× pracovní činnosti, 2× výtvarná výchova, 1× hudební výchova.

Devátá třída 4× matematika, 5× český jazyk, 3× tělesná výchova, 1× fyzika, 2× přírodopis, 1× zeměpis, 3× dějepis, 3× anglický jazyk, 2× dvouhodinová hodina informatiky, 1× pracovní činnosti, 1× výtvarná výchova, 1× hudební výchova.

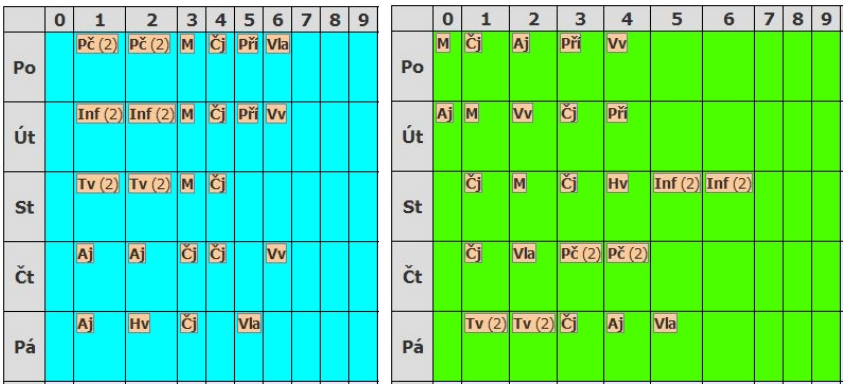
Kvalita generování pro třídy

Jelikož je zde skoro čtyřikrát více rozvrhů než v první části, nebudou zde zobrazovány jednotlivé rozvrhy, ale jen některé ukázkové na obrázcích 7.7 a 7.8. Kvalitu generovaného rozvrhu lze nejlépe vyčíst z následující tabulky.

Tabulka 7.2: Porovnání rychlostí našeho generátoru a aSc Rozvrhy na středně těžkém rozvrhu. Časy generování vypadají lépe pro aSc Rozvrhy, ale v takto krátkých časech nemá smysl zdůrazňovat rozdíly; kvalita rozvrhu je ovšem srovnatelná, což je vidět z počtu nultých hodin a počtu hodin po sedmé vyučovací.

	aSc rozvrhy			**navržený generátor**		
Složitost	Normální	Velká	Velmi velká	10 verzí	50 verzí	100 verzí
Čas [sec]	1	2	4	1	5	12
Nulté	24	21	20	6	3	2
Po sedmé	2	1	0	7	9	8

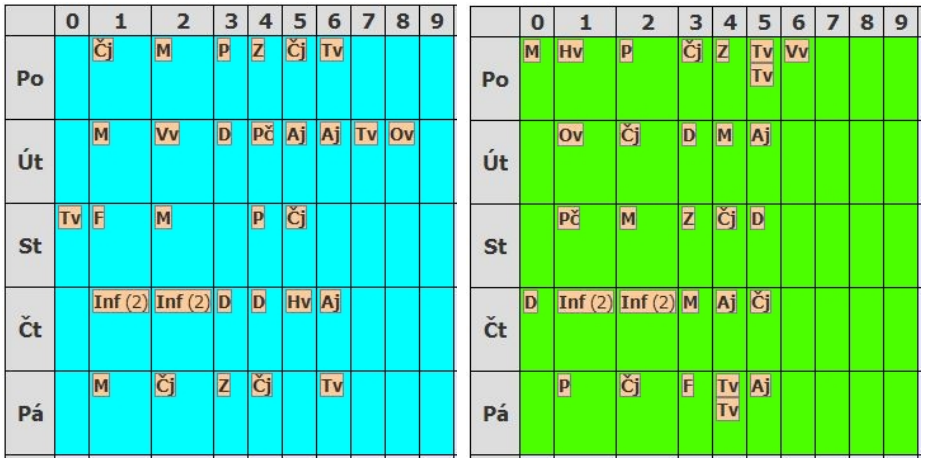
Z tabulky vyplývá, že kvalita rozvrhu je pro navržený generátor i aSc Rozvrhy podobná. Liší se hlavně tím, že navržený generátor upřednostňuje odpolední vyučování, zatímco aSc Rozvrhy dával přednost nultým hodinám. Tyto nastavení se ovšem dají upravit. Co se týče rychlosti, je aSc Rozvrhy rychlejší, ovšem i navržený generátor zvládl vygenerovat rozvrh dostatečně rychle.



	0	1	2	3	4	5	6	7	8	9
Po		Pč (2)	Pč (2)	M	Čj	Pří	Vla			
Út		Inf (2)	Inf (2)	M	Čj	Pří	Vv			
St		Tv (2)	Tv (2)	M	Čj					
Čt		Aj	Aj	Čj	Čj		Vv			
Pá		Aj	Hv	Čj		Vla				

	0	1	2	3	4	5	6	7	8	9
Po	M	Čj	Aj	Pří	Vv					
Út	Aj	M	Vv	Čj	Pří					
St		Čj	M	Čj	Hv	Inf (2)	Inf (2)			
Čt		Čj	Vla	Pč (2)	Pč (2)					
Pá		Tv (2)	Tv (2)	Čj	Aj	Vla				

Obrázek 7.7: Porovnání rozvrhu pro pátou třídu (nejproblémovější, neboť její učitel, stejně jako v předešlé části, učí angličtinu v ostatních třídách). Vlevo navržený generátor, Vpravo výsledek aSc Rozvrhy. Jedná se o kvalitativně podobné rozvrhy, aSc Rozvrhy opět volil nulté hodiny. Navržený generátor umisťoval do odpoledních.



	0	1	2	3	4	5	6	7	8	9
Po		Čj	M	P	Z	Čj	Tv			
Út		M	Vv	D	Pč	Aj	Aj	Tv	Ov	
St	Tv	F	M		P	Čj				
Čt		Inf (2)	Inf (2)	D	D	Hv	Aj			
Pá		M	Čj	Z	Čj		Tv			

	0	1	2	3	4	5	6	7	8	9
Po	M	Hv	P	Čj	Z	Tv Tv	Vv			
Út		Ov	Čj	D	M	Aj				
St		Pč	M	Z	Čj	D				
Čt	D	Inf (2)	Inf (2)	M	Aj	Čj				
Pá		P	Čj	F	Tv Tv	Aj				

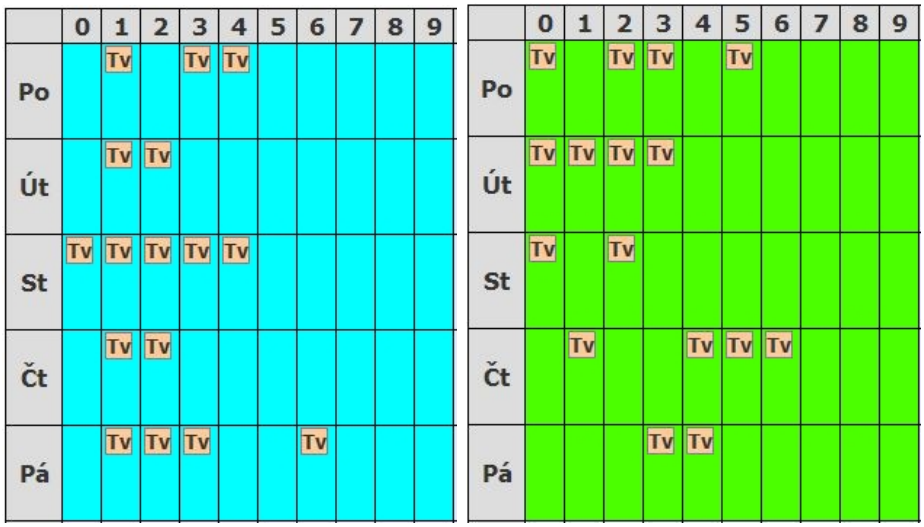
Obrázek 7.8: Porovnání rozvrhu pro devátou třídu. Vlevo navržený generátor, vpravo výsledek aSc Rozvrhy. Opět se jedná o kvalitativně podobné rozvrhy, aSc Rozvrhy opět volil nulté hodiny. Navržený generátor umisťoval do odpoledních.

Kvalita generování pro učitele

Rozvrh pro druhý stupeň této školy už je z hlediska učitelů zajímavější. Učitelé už zde neučí pouze jednotlivé třídy, ale jsou specializováni na předměty.

Porovnání vygenerovaných rozvrhů učitelů jsou na obrázcích 7.9 a 7.10.

Obrázek 7.9: Porovnání rozvrhu pro učitele matematiky a zeměpisu. Vlevo navržený generátor, vpravo výsledek aSc Rozvrhy. Rozvrhy odpovídají předchozím pokusům, kdy aSc Rozvrhy umisťoval položky do nultých hodin, zatímco navržený generátor do odpoledních.



Obrázek 7.10: Porovnání rozvrhu pro učitele tělesné výchovy. Vlevo navržený generátor, vpravo výsledek aSc Rozvrhy. Ani jeden z generátorů se nevyhnul dvěma volným hodinám. Rozvrhy jsou jinak opět velice podobné.

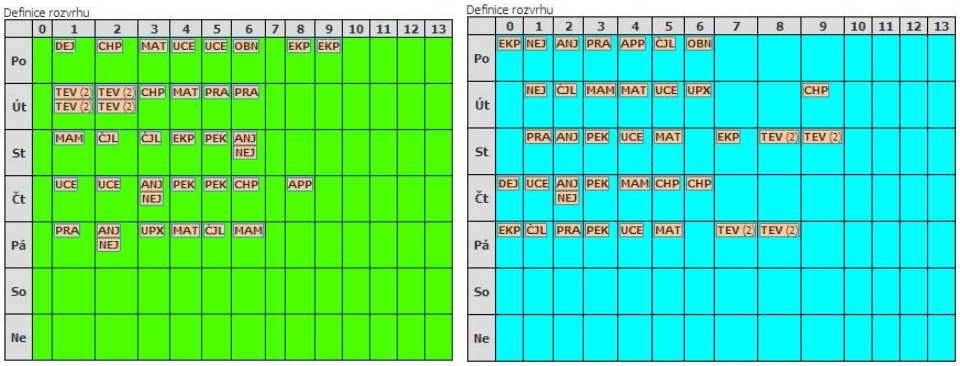
7.1.3 Rozvrh pro velké školy

Porovnávaný rozvrh

Jako rozvrh na porovnávání byl k dispozici rozvrh z reálné školy, který obsahuje 897 událostí v databázi (tj. vyučovaných předmětů, z nichž je dohromady rozvrhováno 2262 hodin). Škola sestává z 61 tříd a ty jsou dále rozděleny na 438 studijních skupin. Počet evidovaných učitelů je zde 900 (za všechna období), z nichž samozřejmě ne všichni vyučují. Reálný počet vyučujících je 65.

Kvalita generování

Pro toto zadání se, i přes velký objem dat, podařilo vygenerovat akceptovatelný rozvrh bez větších mezer ve výuce. U některých tříd bohužel nebylo možné vyhnout se odpoledním vyučování, jelikož kombinování tolika hodin je složité. Porovnání generovaného a ukázkového rozvrhu je na obrázku 7.11, kde je k vidění rozvrh, který se kvalitativně neliší od ukázkového rozvrhu. Navrženému generátoru se bohužel ve většině případů nepodařilo umístit všechny položky (při smazání celého rozvrhu a přegenerování), z celkového počtu 897 se jednalo o 5–15 neumístěných položek při generování. Pokud se smazal například jenom rozvrh jedné třídy, nebo učitele a poté se přegeneroval rozvrh jen pro tyto položky, pak se počet neumístěných předmětů pohyboval kolem nuly. Ukázka jednoho z rozvrhů je k dispozici na obrázku 7.11.



Obrázek 7.11: Porovnání generovaných rozvrhů. Vlevo původní rozvrh generovaný aSc Rozvrhy, vpravo rozvrh vygenerovaný navrženým programem. Je vidět, že navržený generátor více využívá nulté hodiny, což se ovšem dá redukovat nastavením příslušného parametru. Co se týče kvality, navrženému generátoru se podařilo umístit jednou správně němčináře a angličtináře, jelikož se dají vyučovat současně, bohužel se mu to nepodařilo u Tělesné výchovy, což by ovšem mohlo napravit opětovné spuštění generátoru, a tím získání jiného řešení.

Rychlost generování

Program aSc Rozvrhy má několik variant složitosti generování. Měření rychlosti pro náš ukázkový rozvrh bylo provedeno pro všechny verze. Pro nejjednodušší složitost generování, která je v programu nazvána jako normální, trvalo generování kolem dvou minut. Na velkou složitost trvalo generování devět minut a na velmi velkou už potřebná doba dosahovala hodnoty 5 hodin a 7 minut, přičemž ani v jednom případě se nepodařilo umístit všechny položky.

U navrženého algoritmu záleží rychlost generování na počtu generovaných možností. Při standardním počtu deseti generovaných rozvrhů trvalo generování na testovacím stroji (Core i7, Windows 7 64 bit) necelou minutu, což je oproti aSc Rozvrhu výrazně rychlejší. Pokud bychom odstranili námi přidanou optimalizaci a spoléhali se čistě na heuristiku, trvalo by generování jen přibližně deset vteřin. Pokud přidáme další generované možnosti, doba běhu se prodlouží. Například u 100 verzí rozvrhu se doba prodloužila na 13 minut, ale výsledek generování byl samozřejmě lepší. Celkové porovnání je zobrazeno v následující tabulce.

Tabulka 7.3: Porovnání rychlostí navrženého generátoru a aSc Rozvrhy. Navržený generátor se ukázal být rychlejším. Je k dispozici i informace o počtu neumístěných hodin, která ovšem nevypovídá zcela o kvalitě rozvrhu. Program aSc Rozvrhy generoval kvalitnější rozvrhy a neumístěné položky měl proto, neboť při generování neměl povoleno umisťovat do nultých hodin a jen minimálně umisťoval do pozdějších hodin, než je desátá. V případě povolení této volby se počty neumístěných hodin výrazně snížily. Co se týče počtů nultých a podvečerních hodin u navrženého generátoru, jsou zde oproti aSc Rozvrhy poměrně velká čísla, musíme ovšem počítat s přepočtem na 61 tříd. Při tomto přepočtu nám vychází zhruba jedna nultá hodina týdně na třídu a jedna podvečerní.

	aSc rozvrhy			**navržený generátor**		
Složitost	Normální	Velká	Velmi velká	10 verzí	50 verzí	100 verzí
Čas[hod:min]	0:02	0:14	7:02	0:01	0:05	0:12
Neumístěno	24	40	18	18	12	10
Počet 0.	3	3	1	100	85	60
Počet po 10.	20	16	14	130	100	70

8 Závěr

Podařilo se vytvořit generátor rozvrhů dle požadavků zadavatele. Generátor je plně funkční a bude v nejbližší době integrován jako jedna ze služeb do internetové aplikace Škola OnLine. Aplikace se stává z 36 zdrojových souborů o celkové velikosti 1,84 MB zdrojového kódu (hlavně kvůli velikosti datasetu) a přibližně 3400 řádek kódu. Výsledný program byl podroben detailnímu testování a jeho výstupy byly srovnány s referenčním programem aSc Rozvrhy.

Porovnáním rozvrhů uvedených v části 7 je zřejmé, že pro malé a střední školy je navržený generátor srovnatelný s programem aSc Rozvrhy a může tak být jeho výhodnou alternativou, zejména z finančních důvodů. Výsledky nám generátor poskytne rychle a generované rozvrhy jsou dostatečně kvalitní. Nezanedbatelná je i výhoda mnohem jednoduššího použití.

Při porovnávání generovaných rozvrhů pro větší školy už rozvrh generovaný navrženým generátorem nebyl tak kvalitní jako rozvrh z generovaný aSc Rozvrhy. Navrženému programu se sice podařilo umístit více položek a za kratší dobu, ovšem za cenu většího zaplnění jednotlivých dnů (aSc Rozvrhy generoval max do 10. hodiny a minimálně využíval nulté, navržený generátor využíval v určitých případech nulté i 11. hodiny).

Výsledky generátorů jsou samozřejmě ovlivněny nastavením omezujících slabých podmínek, při jiném nastavení bychom mohli dostat úplně jiné výsledky. Je třeba zdůraznit, že nastavování omezujících slabých podmínek záleží na preferencích dané školy a není možné je stanovit obecně (například některé školy preferují ranní výuku, jiné odpolední, ...). Pro následnou úpravu vygenerovaného rozvrhu lze použít internetové rozhraní, které je popsané v části A.2.

Výše uvedené body dokazují, že cíle této diplomové práce byly úspěšně splněny.

Použité zdroje

[1] 31. ZŠ Plzeň. *Stránky 31. ZŠ Plzeň dostupné na http://www.zs31.plzen-edu.cz/.*

[2] aSc Rozvrhy. *Stránky aSc Rozvrhy, dostupné na http://www.asctimetables.com/.*

[3] CCA Group a.s. *Interní stránky společnosti CCA Group a.s. dostupné na http://wiki.cca.cz.*

[4] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* Cambridge, 2009.

[5] J. Glynn, K. Watson, M. Skinner, S. Robinson, C. Nagel, K. Allen, O. Cornes, Z. Greenvoss, and B. Harvey. *C# Programujeme profesionálně.* Praha, 2008.

[6] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* 1989.

[7] B. Johnson. *Professional Visual Studio 2012.* 2012.

[8] M. T. Jones. *Artificial Intelligence: A Systems Approach.* 2008.

[9] Škola OnLine. *Stránky Škola OnLine, dostupné na http://www.skolaonline.cz.*

[10] log4net. *Stránky knihovny log4net, dostupné na http://logging.apache.org/log4net/.*

[11] R. Pecinovský. *Návrhové vzory.* Praha, 2007.

[12] Podmínky. *Přednášky prof. RNDr. Romana Bartáka, Ph.D. dostupné na http://kti.mff.cuni.cz/ bartak/podminky.*

[13] Rozvrhování. *Přednášky prof. RNDr. Romana Bartáka, Ph.D. dostupné na http://kti.mff.cuni.cz/ bartak/planovani/index.html.*

[14] M. Russo and P. Pialorsi. *Microsoft LINQ.* Praha, 2009.

[15] J. Sharp. *Microsoft Visual C# 2010.* Praha, 2010.

[16] Task Parallel Library. *Popis knihovny TPL pro paralelní zpracování dostupné na http://msdn.microsoft.com/en-us/library/dd460717.aspx.*

[17] P. Töpfer. *Algoritmy a programovací techniky.* Praha, 2007.

[18] M. Virius. *Základy algoritmizace.* Praha, 1995.

[19] P. Wróblewski. *Algoritmy: Datové struktury a programovací techniky.* Brno, 2004.

A Přílohy

A.1 Zprovoznění aplikace

Aplikace byla vyvíjena ve vývojovém prostředí Visual Studio 2012 (popis tohoto produktu najdete například v [7]). Pro její spuštění je potřeba mít k dispozici knihovnu log4net (je součástí předávané aplikace) a změnit v souboru `app.config` connection string aby odpovídal připojené databázi. V connection stringu je třeba změnit část Data source. Podoba je následující:

```
Data Source=Jmeno_Stroje;Initial Catalog=Jmeno_Databaze;
```

Dodávaná databáze je testovací a neobsahuje data skutečných škol.

Aplikace je dodána jako Solution, která se dá otevřít v prostředí Visual Studia 2012 a následně spustit. Současně je dodán i přeložený program pro spuštění bez nutnosti tohoto otevření.

Nastavení parametrů

Parametry pro ohodnocení jednotlivých voleb se prozatím nastavují pouhým přepsáním hodnoty v `.resx` souboru prostředky Visual Studia. Tento soubor je umístěn ve složce `Resources`. Po zobrazení `.resx` souboru uvidíme tabulku jako na obrázku A.1, kde přepsáním libovolné hodnoty změníme chování programu.

Name	Value	Comment
HodnoceniNeumisteno	-150	Penalizace za každou neumístěnou položku
OhodnoceniDopoledne	40	Ohodnocení dopolední hodiny
OhodnoceniNeobsazenaDopol	-150	Penalizace za neobsazenou dopolední hodinu
OhodnoceniNulta	-200	Ohodnocení nulté hodiny
OhodnoceniOdpoledne	-3	Ohodnocení odpoledních hodin
OhodnoceniOpakovani	5	Ohodnocení vhodnosti opakování
OhodnoceniPodvecer	-10	Ohodnocení večerních hodin
OhodnoceniSkupiny	40	Ohodnocení skupiny
OhodnoceniVolnaPredTrida	-40	Penalizace za jednu volnou hodinu před - nepočítá se nultá
OhodnoceniVolnaPredUcitel	-5	Penalizace volná před pro učitele
PocetDopolednichHodin	5	Počet dopoledních hodin
PocetHodin	13	Kolik hodin bude maximálně denně
PocetOdpolednichHodin	2	Počet hodin odpoledne - zbytek je podvečer
StejnyPocetHodinVTydnu	100	Bonus za stejný počet hodin v jednotlivých dnech

Obrázek A.1: Nastavení hodnot generátoru.

A.2 Ovládání aplikace

Aplikace zatím není přímo integrována do aplikace Škola OnLine, ale její budoucí integraci by nemělo nic bránit. Integrace nebyla součástí diplomové práce, bude provedena pracovníky Školy OnLine později.

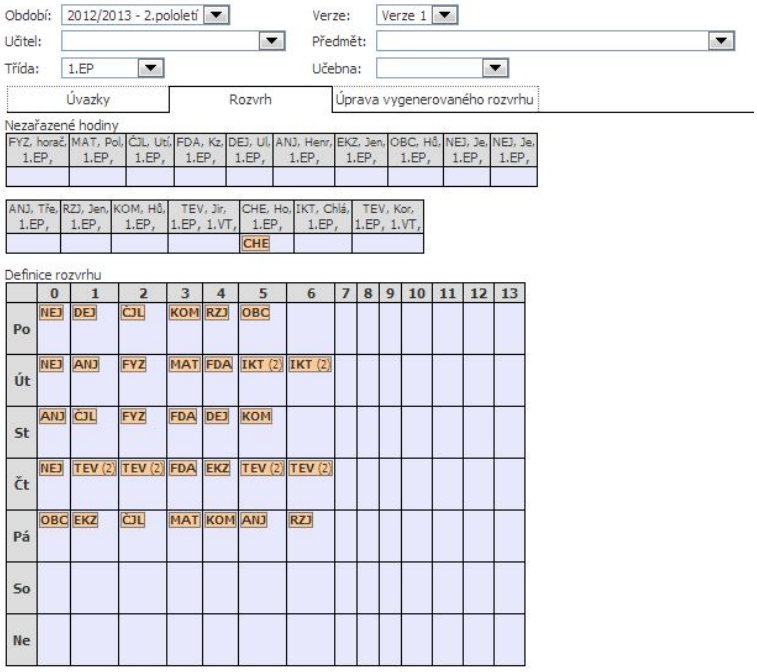
Data z aplikace se nyní ukládají do databáze a v aplikaci Škola OnLine se poté dají zobrazit. Příklad zobrazení těchto dat a popis práce s nimi je v následujících odstavcích.

A.2.1 Zobrazení v aplikaci Škola OnLine

Data vygenerovaná naší aplikací je možno uložit do databáze systému Školy OnLine a z této databáze je poté zobrazit v grafickém rozhraní. Toto grafické rozhraní najdeme pod položkou `Rozvrh` v hlavním menu aplikace, dále zvolíme nabídku `Tvorba rozvrh`u a poté stejnojmennou nabídku v podmenu. Přes nabídkové seznamy, které jsou na této stránce k dispozici, si můžeme vybrat učitele, třídu, nebo předmět, jehož rozvrh chceme zobrazit. Kromě toho si také můžeme zvolit období (školní rok a pololetí), kdy se podle rozvrhu vyučuje a nakonec i jeho verzi. Vybraný rozvrh se poté zobrazí v tabulce ve spodní části stránky, kde s ním můžeme dále pracovat. Kromě tvorby rozvrhu můžeme z hlavního menu vybrat i jiné možnosti, jako jsou například kontroly rozvrhu, dělení tříd, definice úvazků atd.

Nezařazené položky

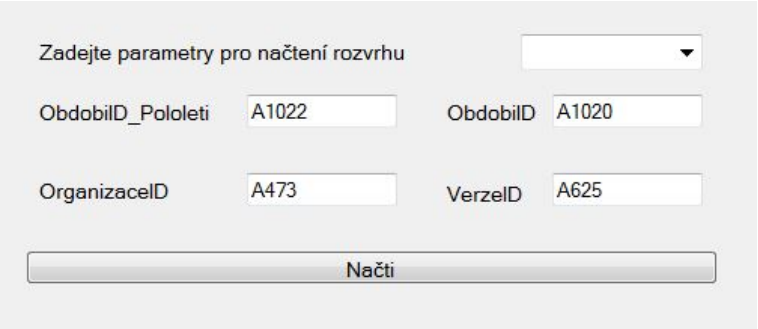
Jelikož se nemusí vždy podařit umístit všechny položky (nenašla se vhodná kombinace nebo to není vůbec možné), umožňuje systém Škola OnLine doumist'ování, popřípadě přemist'ování položek ručně. Nezařazená položka je vidět na obrázku A.2, kde se jedná o položku CHE v tabulce nad rozvrhem. Pokud chceme nezařazenou položku umístit, pak ji pouze přetáhneme do rozvrhu na požadovanou pozici a vybereme třídu, kde se má tato hodina vyučovat. Pro definitivní umístění položky poté zvolíme možnost `Potvrdit` v dialogu. Stejný princip použijeme i v případě, kdy chceme některou položku přemístit.



Obrázek A.2: Nezařazená položka v rozvrhu.

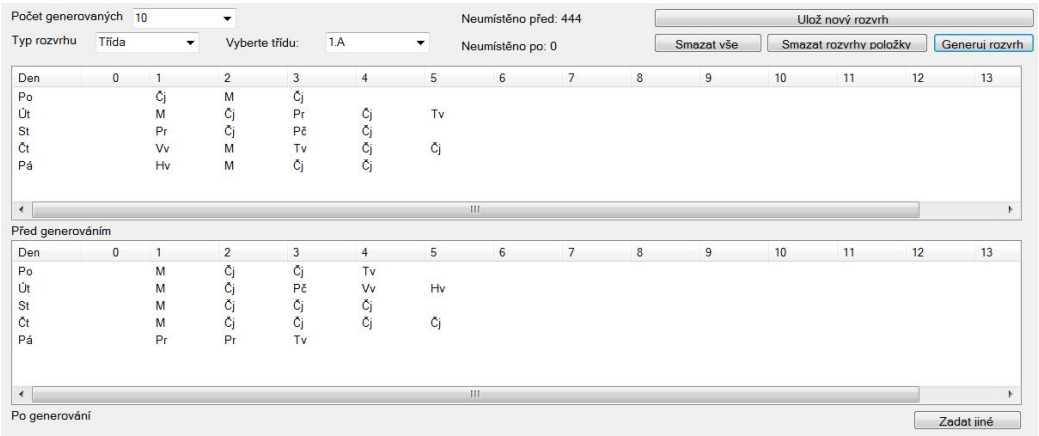
A.3 Vizualizace v aplikaci

Pro testovací účely bylo vytvořeno jednoduché uživatelské rozhraní pro zobrazení rozvrhu. Toto uživatelské rozhraní bylo vytvořeno na platformě .NET, konkrétně ve WinForms. Jedná se pouze o jednoduché GUI, které je využíváno jen pro ladění. Po spuštění (spustitelný program má název VizualizaceRozvrhu.exe) se nám zobrazí možnost zadání dat potřebných pro načtení rozvrhu. Jedná se o ID roku a pololetí, verzi rozvrhu a organizaci – tyto údaje se zjistí z číselníků Škola OnLine. V aplikaci je možnost předvyplnit tyto hodnoty na ty, které byly použity při testování (viz A.3).



Obrázek A.3: Nastavení parametrů pro načtený rozvrh. V pravém horním rohu lze vybrat předdefinované parametry, pokud nějaké existují.

Po načtení se nám zobrazí okno s načtenými rozvrhy viz A.4. V tomto okně si můžeme v horní části zvolit počet možností, kolik rozvrhů se bude generovat (a z nich vybírat nejlepší), dále jestli zobrazujeme rozvrh třídy, učitele nebo místnosti a poté můžeme vybrat konkrétní třídu, učitele, nebo místnost. Ve velkých oknech se nám poté zobrazí samotný rozvrh. Horní rozvrh je ten, co se načítá ze serveru Škola OnLine, dolní je náš vygenerovaný. Pokud chceme generovat nový rozvrh, musíme nejdříve smazat ten starý tlačítky Smazat vše (pro úplně čistý rozvrh), nebo smazat rozvrh položky, pro smazání rozvrhu jen pro jednu entitu. Generování spustíme stiskem tlačítka `Generuj rozvrh`. Vygenerovaný rozvrh můžeme uložit do databáze tlačítkem `Uložit`. Při tomto ukládání musíme vzít na vědomí, že ukládaný rozvrh přepíše rozvrh původní.



Obrázek A.4: Hlavní okno pro generování rozvrhu.

A.4 Obsah přiloženého DVD

Na DVD, jež je přiloženo k této diplomové práci, se nachází dvě složky:

- Dokumentace – obsahuje elektronickou podobu této dokumentace a dokumentaci vygenerovanou z kódu
- Program – program vytvořený v rámci diplomové práce a databáze potřebná ke spuštění programu, dále jsou tyto složky.
  - Databáze – obsahuje vývojovou databázi ze Školy OnLine
  - RozvrhDP – v této složce je umístěna solution pro otevření ve Visual Studiu 2012, dále jsou zde dva projekty

    RozvrhDP – samotný generátor rozvrhu

    RozvrhVizualizace – zobrazení rozvrhu – program na testování generování.
- Spustitelná verze – obsahuje přeložené soubory potřebné pro spuštění programu (kromě databáze).