VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

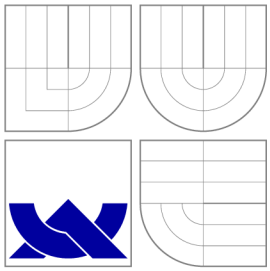
# SOFTWARE PRO TVORBU ROZVRHŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

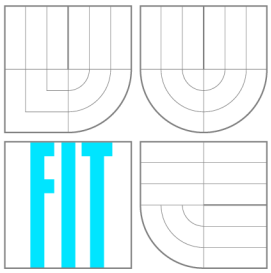
AUTOR PRÁCE
AUTHOR

Bc. JAN BUREŠ

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# SOFTWARE PRO TVORBU ROZVRHŮ

SOFTWARE FOR DESIGN OF TIMETABLES

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN BUREŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2015

# Abstrakt

Cílem této diplomové práce je navrhnout software pro střední a základní školy, který bude sloužit k vytváření náhradních rozvrhů při zastupování pedagogických pracovníků. V současnosti sice existují podobně zaměřené programy, ale přesto je většina škol nevyužívá. Tyto programy obsahuji mnoho nedostatků, které jsou v této práci taktéž popsány. Tvorba náhradního rozvrhů je poměrně náročná a člověk samotný musí hlídat mnoho podmínek. Kontrola všech podmínek by měla být hlavní výhodou programu. Návrh a tvorba programu je založena na požadavcích zaměstnanců škol, kteří v současnosti vytváří náhradní rozvrhy ručně. Velký důraz je kladen na přívětivost uživatelského rozhraní a jeho jednoduché osvojení.



# Abstract

The aim of this thesis is to design a software for elementary, middle and high schools, which will serve to create alternative timetables for staff substitutions. There are similarly oriented programs, but most schools don't use them. These programs contain many flaws, which are also discussed in this work. Creating substitution schedules is complicated, and a lot of conditions must be taken care of. Checking all the conditions should be the main advantage of the application. Design and implementation of the application is based on requirements by a school, where substitution timetables are produced manually. Great emphasis is placed on user friendliness and ease of use.



# Klíčová slova

Uživatelské rozhraní, škola, náhradní rozvrhy, zastupování pedagogických pracovníků, teorie grafů.

# Keywords

User interface, school, alternative timetables, substitute teachers, graph theory.

# Citace

Jan Bureš: Software pro tvorbu rozvrhů, diplomová práce, Brno, FIT VUT v Brně, 2015

# Software pro tvorbu rozvrhů

## Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Prof. Dr. Ing. Pavla Zemčíka. Další informace mi poskytli zaměstnanci firmy MP-Soft, a.s. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Bureš
27. května 2015

## Poděkování

Velké poděkování patří zaměstnancům firmy MP-Soft, a.s. - Ing. Jiřímu Soukenkovi, Ing. Radomilu Maškovi a panu Tomáši Borkovi za vytvoření tématu diplomové práce, odborné konzultace a praktické informace ze školního prostředí. Velké díky patří i Prof. Dr. Ing. Pavlu Zemčíkovi za užitečné rady a vedení diplomové práce.

# Obsah

# Kapitola 1

# Úvod

V současnosti jsou počítače rozšířeny do každé oblasti lidské působnosti. Člověk tedy přichází do styku s počítačovými aplikacemi denně ve své práci nebo ve svém volném čase. Každá počítačová aplikace komunikuje s člověkem pomocí uživatelského rozhraní. V dnešní době se tak stává, že kvalita uživatelského rozhraní je nejdůležitějším kritériem při hodnocení aplikace a závisí na něm úspěch či neúspěch celého jejího vývoje.

Tato práce obsahuje poznatky týkající se tvorby uživatelského rozhraní. Informace jsou uvedeny z pohledu člověka, který vytváří uživatelské rozhraní. Nejedná se o kompletní rozbor problematiky, ale pouze o souhrn informací, které jsem využil při návrhu aplikace pro tvorbu náhradních rozvrhů především středních a základních škol. Práce také popisuje postup při návrhu uživatelského rozhraní této aplikace.

Téma práce se úzce opírá o teorii grafů. Poznatky z této oblasti jsou potřebné zejména při návrhu algoritmu, který na základě vybrané hodiny vyhledá náhradní varianty. Tedy vyhledá možnosti, jak danou hodinu nahradit nebo přesunout, tak aby to co nejlépe vyhovovalo všem zúčastněným.

Cílem této práce je navrhnout uživatelské rozhraní softwaru pro tvorbu náhradního rozvrhů a vytvoření algoritmu pro hledání náhradních variant v rozvrhu učitelů nebo tříd. Následně pak implementovat ukázku této aplikace.

Téma mi nabídla společnost, která vyvíjí systém pro správu školní agendy. Tento systém je určen jak pro základní školy, tak pro školy střední. Jejich aplikace pokrývá veškerou potřebu škol od klasifikace až po správu majetku. Jako jednu ze slabin systému označili tvorbu náhradních rozvrhů neboli suplování. Většina škol, která jejich software využívá, tvoří náhradní rozvrhy ručně. Pro pracovníky škol tato činnost představuje velkou zátěž. Pro člověka je velmi problematické uhlídat veškeré podmínky, které jsou pro náhradní rozvrh rozhodující. Provést kruhovou výměnu hodin, je pro člověka skoro neuskutečnitelné. Přitom pro počítačový program nemusí být hlídání velkého množství podmínek, případně nalezení kruhové výměny, až tak náročné.

Při konzultacích na školách jsem se setkal s pracovníky, kteří veškeré náhradní rozvrhy tvořili ručně. Na mnoho papírů nejprve rozepsali všechny potřebné informace a poté hledali nejvhodnější variantu. Velké množství takto shromážděných informací zvyšuje nárok na lidský faktor a zvyšuje se pravděpodobnost výskytu chyby. Při návrhu rozvrhu se tak stávalo, že například na suplování byl napsán chybějící učitel. Nebo naopak při náhlé absenci učitele nebylo ošetřeno dříve nastavené suplování. Hlavní výhodou mnou navrženého programu spočívá v odstranění těchto vlivů při plánování náhradních rozvrhů. V neposlední řadě by taktéž nabídka možností náhradních variant měla být bohatší a to z důvodů, které jsem jmenoval v předchozím odstavci.

Celý proces se odehrával ve spolupráci s pracovníky škol, kteří tvoří náhradní rozvrhy, aby výsledný program co nejvíce reflektoval jejich nároky a požadavky na výslednou aplikaci. Spolupráce spočívala především v konzultacích ve školách s pracovníky škol a prezentování programu od rané fáze jeho vývoje.

V následující kapitole číslo 2 se nachází teoretický přehled problematiky tvorby uživatelských rozhraní. Kromě vysvětlení pojmu uživatelské rozhraní jsou zde uvedeny informace, jak postupovat při jeho návrhu. V další kapitole číslo 3 jsem uvedl přehled znalostí z oblasti teorie grafů, které jsem využil při hledání náhradních variant v rozvrhu tříd. Kapitola 4 se zaměřuje na porovnání současných systémů pro tvorbu náhradních rozvrhů. V kapitole je taktéž uveden postup tvorby náhradního rozvrhu ve školách a některá kritéria, která se při tvorbě náhradního rozvrhu uplatňují. Kapitola obsahuje výtah z důležitých části zákona o zastupováni pedagogických pracovníků. V 5. kapitole se nachází popis návrhu uživatelského rozhraní aplikace, způsobu jak bude uživatel s aplikací pracovat a v čem mu bude aplikace napomáhat. Implementace takto navržené aplikace je pak vysvětlena v kapitole 6. Tato kapitola se také věnuje srovnání s konkurenčními programy a návrhům, jak dále v mé práci pokračovat. Poslední 7. kapitolou práce je závěr, která obsahuje shrnutí dosažených výsledků.

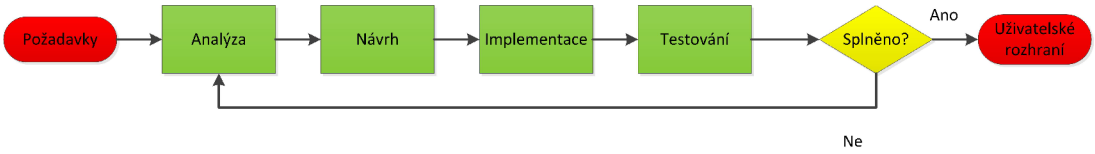
# Kapitola 2

# Proces tvorby uživatelského rozhraní

Kapitola stručně popisuje proces vytváření uživatelského rozhraní. Cíl tohoto procesu je většinou stejný – spokojený uživatel [1]. V kapitole jsou uvedeny informace, které se v současné době využívají k dosažení tohoto cíle. Na začátku kapitoly jsou vyjmenovány jednotlivé části tvorby uživatelského rozhraní. Následuje přehled typů uživatelských rozhraní a jejich výhody a nevýhody. Dále se kapitola věnuje pouze grafickému uživatelskému rozhraní a jednotlivým částem jeho vytváření.

## 2.1 Průběh tvorby uživatelského rozhraní

Možností jak postupovat při návrhu uživatelského rozhraní existuje mnoho. Nicméně v základních bodech se neliší [2]. Proces návrhu uživatelského rozhraní obsahuje zpravidla několik fází, které se mohou cyklicky opakovat. Na začátku celého procesu jsou požadavky na výslednou aplikaci (funkčnost, zaměření, cílová skupina uživatelů, ...). Na základě požadavků vytvoříme analýzu daného problému a návrh řešení. Následně tento návrh implementujeme do prototypu, který je dále testován ve spolupráci s uživateli. Poznatky získané z testování nám poslouží při následujícím cyklu návrhu. Tento postup je znázorněn graficky na obrázku 2.1.




Obrázek 2.1: Diagram vývoje uživatelského rozhraní

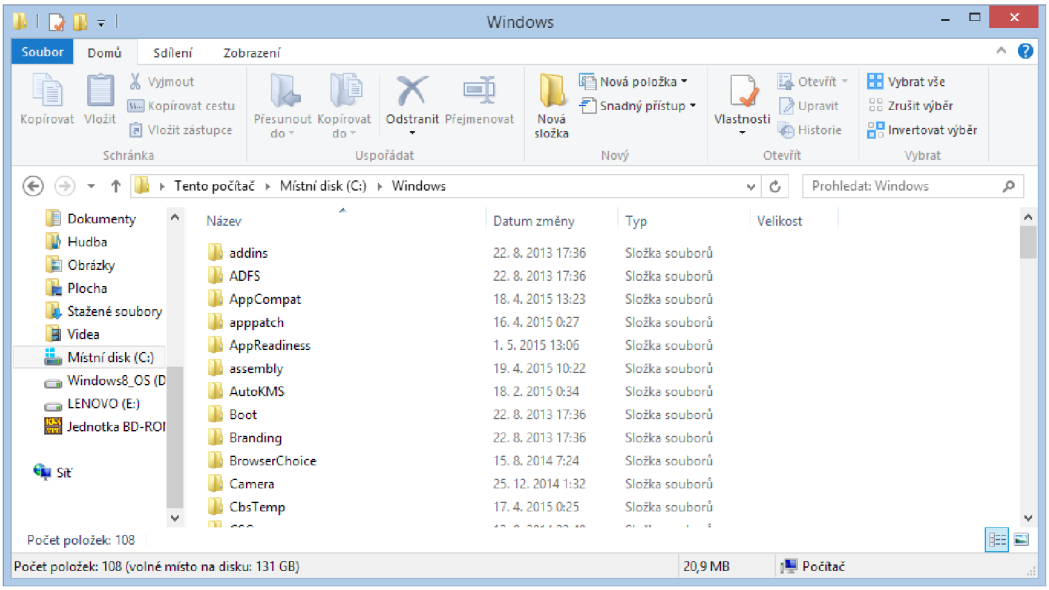
Protože je velmi těžké určit, kdy je uživatelské rozhraní hotové, je potřeba na začátku procesu stanovit metodiku, jak změřit úspěšnost navrženého rozhraní [3]. Tato metodika je většinou založena na některých měřitelných faktorech. Jedná se buď o objektivní veličiny (chybovost, rychlost apod.) nebo subjektivní pocity uživatelů. Výsledné rozhraní se také může porovnat s již hotovou, stejně zaměřenou aplikací (počet kroků k dosažení určitého cíle apod.) [4].

## 2.2 Typy uživatelských rozhraní

Dnes jsou počítače součástí každé oblasti lidské činnosti [5]. Lidé s počítačem komunikují téměř denně, protože jej využívají ke své práci. Komunikace člověka s počítačem je interaktivní [1]. Uživatel prostřednictvím ovládacích prvků definuje vstup a počítač na základě vstupu vytvoří výstup, který prezentuje uživateli [2]. V počátcích výpočetní techniky se jednalo o různé přepínače a tlačítka na straně vstupu a o světelné signály na straně výstupu [5]. V současnosti existuje několik typů uživatelských rozhraní. Nejčastěji jsou to tyto tři typy:

- **Příkazový řádek:** Rozhraní bez grafických prvků, které se ovládá pomocí textových příkazů. Výstup je taktéž textový. Je velmi efektivní tam, kde s počítačem pracuje odborník, který se naučí veškeré příkazy. Hardwarově nejméně náročné rozhraní. Historicky nejstarší uživatelské rozhraní.

- **Grafické uživatelské rozhraní:** Komunikace mezi člověkem a počítačem probíhá pomocí objektů na obrazovce. S tímto uživatelským rozhraním se nejlépe pracuje převážné většině uživatelů [2]. Náročnější na výpočetní výkon počítače. Nejvíce využívané v osobních počítačích. Nejmodernější uživatelské rozhraní. Ukázka grafického uživatelského rozhraní je znázorněna na obrázku 2.2.




Obrázek 2.2: Ukázka grafického uživatelského rozhraní

- **Textové uživatelské rozhraní:** Jde o vývojový mezistupeň mezi příkazovým řádkem a grafickým uživatelským rozhraním. Na obrazovce se nachází několik znaků, které vytváří prvky uživatelského rozhraní. Mohou to být jak znaky textové, tak znaky reprezentující například roh okna. Nejvíce využívané bylo v době operačního systému DOS [5]. Dnes se již moc nevyskytuje.

## 2.3 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní se skládá z objektů, se kterými uživatel manipuluje. Objekty jsou umístěny na obrazovce. Výsledek této akce pak počítač také zobrazí na této obrazovce [1]. Tento způsob ovládání se nazývá *přímá manipulace* [2]. Používají se zde techniky *Drag and drop* (uchop a přemísti), kdy uživatel vybraný objekt přesouvá stejně jako v reálném světě (přesouvání souborů do koše pro smazání nebo přesunutí souboru ze složky do složky). V poslední době se tato technika rozšířila na *Look and feel* (shlédni a porozuměj), kdy se kromě přetahování objektů používají i různá jiná gesta, která jsou podobná gestům v reálném světě. Uživateli tak připadá prostředí povědomé a lehce se v něm zorientuje [2].

V dnešní době je nejpoužívanějším konceptem grafického uživatelského rozhraní takzvaný WIMP [2]. Jedná se o zkratku slov windows (okna), icons (ikony), menu (nabídka) a pointer (polohovací zařízení). Pomocí oken je rozdělena plocha obrazovky mezi několik současně spuštěných aplikací popřípadě mezi více kontextů v rámci aplikace. Ikony znázorňují programy, adresáře, složky nebo zařízení, jako například tiskárny. V menu nalezne uživatel dostupné akce, které slouží k ovládání aplikace. Pomocí polohovacího zařízení (nejčastěji se používá myš [2], ale v současné době se rozšiřují i dotekové obrazovky) uživatel manipuluje s objekty, které vidí na obrazovce.

Při návrhu uživatelského rozhraní se využívají dva pohledy na problematiku. V počátcích vývoje výpočetní techniky se nejvíce využíval přístup *machine-centered*. Obsluhu počítačů tvořili technicky vzdělaní lidé a nebylo jich mnoho. Proto se uživatelské rozhraní přizpůsobovalo co nejvíce skutečné funkci stroje [5]. Ovládacími prvky se nastavovaly přímo vlastnosti zařízení. Stroj bylo často určený pouze k vykonávání jedné činnosti popřípadě více činností podobného zaměření [5]. Postupem času se výpočetní technika rozšiřovala mezi obyčejné lidi. A stávala se více a více univerzální. Bylo tedy nutné uživatelská rozhraní přizpůsobit co nejvíce běžnému člověku. Tento přístup se nazývá *people-centered* [2]. Ovládací prvky se více zaměřovaly na způsob, jakým uživatel vykonává svou práci v reálném prostředí. Uživatel tedy pouze převádí úkony, které jsou mu známé, do prostředí počítače.

Práce návrháře spočívá ve správném sestavení těchto prvků ve výsledné aplikaci. Obor, který se zabývá studií jednoduchosti a přívětivosti uživatelského rozhraní, se nazvývá *Human-Computer Interaction* [3]. Tento obor se skládá z mnoha disciplín [1]. Návrhář musí znát základy psychologie, sociologie, fyzické možnosti, kulturní zvyklosti konkrétního člověka nebo skupiny lidí, pro které je výsledná aplikace určena [1]. Proto je způsob návrhu, kdy se do středu staví uživatel náročnější, ale s výslednou aplikací se uživateli pracuje daleko lépe [2].

Existuje mnoho způsobů, jak vytvářet uživatelská rozhraní [3]. V této práci je dále uveden pouze přístup nazývaný *PACT*, protože na popis všech možných způsobů není dostatečný prostor. Zkratka *PACT* se skládá z těchto slov:

- People – lidé, kteří budou s aplikací pracovat (uživatelé).
- Activities – aktivity, které by měly být dostupné.
- Contexts – souvislosti, ve kterých budou aktivity vykonávány.
- Technologies – technologie, které použijeme.

Jedná se o 4 pohledy na uživatelské rozhraní, na které by se měl návrhář zaměřit. Všechny 4 body se navzájem ovlivňují.

## 2.4 Analýza požadavků

Při vytváření uživatelského rozhraní je možné využít přístup *PACT*. O takové analýze se hovoří jako o *PACT analýze* [2].

### Lidé

Lidmi se myslí skupina lidí, kteří budou prostřednictvím navrženého uživatelského rozhraní komunikovat s aplikací (dále cíloví nebo koncoví uživatelé). Koncoví uživatelé jsou rozděleni do skupin (podle schopností, odbornosti, věku, pohlaví atd.). Každou skupinu zastupuje takzvaná *persona*. Jedná se o fiktivní postavu, která je typická pro danou skupinu lidí[1]. Je důležité si uvědomit, že i přes některé společné vlastnosti je každý člověk jedinečný. Návrhář by proto měl mít na mysli právě odlišnosti jednotlivců, aby mohl navrhnout přívětivé uživatelské rozhraní pro všechny uživatele [4].

Na první pohled jsou nejvíce viditelné různé fyzické vlastnosti člověka (velikost, váha atd.). Návrhář by se měl snažit vytvořit takové uživatelské rozhraní, aby s ním kromě průměrných lidí mohli pracovat i ti nějakým způsobem vyčnívající (nejmenší, nejvyšší, pravoruký, levoruký atd.) [1].

Kromě fyzických vlastností se lidé různí i ve vnímání světa pomocí smyslových orgánů (zrak, sluch, hmat, čich a chuť) [2]. Tyto vlastnosti mají největší vliv na to, jak zábavné, přístupné a použitelné bude uživatelské rozhraní pro konkrétního uživatele [2]. Každý člověk vnímá okolní svět trochu jinak než ostatní, ale jsou i určitá pravidla, která platí pro všechny. Například objekty, které jsou na obrazovce menší, jsou vnímány jako vzdálenější. Do stejné skupiny patří i vnímání barev (např. asociace červené barvy s nebezpečím). Tato obecná pravidla často vychází ze situací, které můžeme pozorovat v každodenním životě a v přírodě kolem nás [4].

Vnímání okolního světa mohou ovlivnit i určitá onemocnění a postižení člověka. Mezi taková postižení patří barvoslepost, kterou trpí asi 8% západoevropské populace [2]. Většinou se však jedná o částečnou poruchu, kdy člověku dělá problém rozeznat jen některé barvy (nejčastěji červenou se zelenou) [1]. Z tohoto důvodu je dobré rozlišovat objekty kromě barvy i jiným poznávacím znamením (např. tvarem nebo textem). Existují i další znevýhodnění, která je však potřeba vzít v potaz podle specifických vlastností konkrétního systému (například vada řeči při hlasovém ovládání aplikace).

To, co člověk uvidí nebo uslyší, si dále interpretuje na základě kulturního prostředí, ze kterého pochází. V uživatelském rozhraní se jedná hlavně o chápání různých symbolů [2]. Například Američané vnímají křížek jako zápor nebo odmítnutí, ale v našich podmínkách může znamenat i potvrzení správnosti odpovědi (formuláře na úřadech). Kultura a náboženství ovlivňují i celkové uvažování člověka a interpretaci informací [2].

Je také rozdíl, zda navrhované rozhraní bude sloužit pro zkušeného nebo pro začínajícího uživatele.[2] Tyto dvě skupiny se liší zejména nároky a očekáváním vzhledem ke svým dosavadním zkušenostem. Zejména u zkušeného uživatele můžeme předpokládat znalost různých symbolů, které se využívají napříč všemi aplikacemi v systému. Špatné použití takového symbolu více znepříjemní práci uživateli, který zná správný význam tohoto symbolu, než uživateli, který tento symbol vidí poprvé.

Uživatel si při interakci s nějakou věcí vytváří takzvaný *mental model* (jedná se o představu, jak daná věc uvnitř funguje) [2]. Pokud chce uživatel dosáhnout nějakého cíle, nejprve si představí provedení této akce na svém *mental modelu* a poté stejný postup aplikuje na model reálný. Proto musí návrhář svůj návrh co nejvíce přizpůsobit *mentalnímu modelu*, který

si uživatel vytvoří. Návrhář tedy musí předvídat to, jak uživatel bude vnímat uživatelské rozhraní aplikace.

### Aktivity

Aktivity popisují, jak bude s uživatelským rozhraním nakládáno. Kromě způsobu interakce určují i jak často bude daný úkon prováděn, kdo bude úkon provádět, jak rychlá by měla být reakce systému a vlivy okolního prostředí. Často prováděné aktivity by měly být uživateli nejdostupnější. Naopak úkony prováděné méně často by uživateli neměly překážet při každodenní činnosti [6]. Mohou být zapsány formálně (případy užití) nebo neformálně jako scénář. Scénářem se rozumí stručný popis uživatele, prostředí, aktivity, kterou bude provádět, a důvod, proč ji chce provádět. Tyto scénáře často formulují i uživatelé jako svoje přání, jak by chtěli se systémem pracovat [2].

### Souvislosti

Jedná se o souvislosti (kontext), během kterých probíhají aktivity. Kromě prostředí je analyzován i sociální a organizační kontext [2]. Prostředí, ve kterém probíhá aktivita, může být různorodé. Návrhář musí zohlednit hlučnost, prašnost, světlo a jiné podmínky podle toho, kde se aktivita bude odehrávat, případně podle toho, kde se bude dané zařízení nacházet. Z hlediska sociálního je sledováno, zda bude uživatel se systémem komunikovat sám nebo s více lidmi. Jednak je rozdílné chování lidí ve skupinách, ale také některá vstupní a výstupní zařízení nejsou vhodná do prostředí, kde se nachází větší počet lidí. Dále je také důležitá informace, zda budou v okolí jiní lidé, kteří mohou uživateli s ovládáním pomoci, pokud si nebude vědět rady (například rozdíl aplikace v bance, kde může úředník pomoci, namísto mobilní aplikace, kde je uživatel odkázán sám na sebe). Organizační kontext je zejména otázkou oprávnění a bezpečnosti. Tedy kdo bude mít k aplikaci přístup a v jakém rozsahu.

### Technologie

Do této části patří hardwarové a softwarové nástroje využité při komunikaci s aplikací. Analyzují se především vstupní (klávesnice, joystick atd.) a výstupní zařízení (obrazovka, tiskárna atd.), ale i způsob komunikace s okolním světem (síťové připojení) a uchovávání dat aplikace (zabezpečení, lokace atd.). Rozhodnutí se provádí na základě aktivity a kontextu používání aplikace. Špatně zvolené technologie můžou významně ovlivnit spokojenost uživatelů s výslednou aplikací (nejčastěji různé poruchy nebo pomalé reakce na uživatelský vstup) [6].

## 2.5 Návrh

Samotný návrh uživatelského rozhraní se liší podle toho, jaký účel bude aplikace plnit. První skupinu tvoří aplikace, které mají pobavit uživatele. Takových aplikací se na trhu nachází velké množství a často se liší pouze uživatelským rozhraním. Uživatelské rozhraní by mělo být tvořené co nejvíce originálně a jeho úkolem je zaujmout uživatele. Musí uživatele motivovat, aby aplikaci často využíval, aby se k ní vracel a aby si ji oblíbil [6]. Pokud uživatel nebude spokojen, poměrně lehce přejde k jiné aplikaci [4].

Druhou skupinu tvoří aplikace, které slouží jako pracovní nástroj uživatele. Zde je možné předpokládat, že uživatel bude tyto aplikace využívat každý den. Většinou není možné zvolit

alternativní program (ať už z důvodu finančního, neexistence podobné aplikace nebo integrace do firemního prostředí). Proto musí být uživatelské rozhraní co nejvíce formální, aby s nimi bylo příjemné pracovat za různých okolností [2]. Při návrhu uživatelského rozhraní pro tento druh aplikací se zpravidla využívají standardní prvky operačního systému. Případně je možné vytvořit vlastní ovládací prvky, které nebudou vzhledově vyčnívat. Pokud toto návrhář dodrží, uživatel se mnohem snáze zorientuje a jednodušeji porozumí celé aplikaci a jejímu ovládání. Opačná situace může vést až k frustraci z používání aplikace [6].

Při výběru vhodného prvku a jeho použití se návrhář rozhoduje na základě poznatků z mnoha oborů (psychologie, sociologie atd.). Tyto poznatky musí vhodně aplikovat na konkrétní problém. Z tohoto důvodu je každé uživatelské rozhraní jedinečné a jeho návrh je náročný [3].

Malým pomocníkem mohou být příručky, které dodávají výrobci operačních systémů. Jsou v nich uvedena pravidla, jak prvky správně používat a jak tyto prvky vzájemně kombinovat. Jedná se však pouze o základní a obecné rady. Některé společnosti tato pravidla vyžadují a jejich dodržování kontrolují. Všechny aplikace pak musí projít schvalovacím procesem před prodejem zákazníkovi. Tento postup platí například pro firmu Apple a operační systém iOS [7]. Firma Microsoft tato pravidla pouze doporučuje dodržovat. Ignorování těchto zásad může vést ke zmatení uživatele. V horším případě uživatel získá špatné návyky, které bude poté uplatňovat v ostatních aplikacích daného systému [4].

## 2.6 Testování

Testování uživatelského rozhraní se například provádí pomocí testů použitelnosti [2]. Uživatelské rozhraní je použitelné, pokud je práce s ním jednoduchá, bezchybná, rychlá a lehce naučitelná [4].

K těmto testům jsou potřeba uživatelé, kteří systém testují používáním a plněním běžných úkolů pro daný systém. Uživatelské rozhraní by se mělo testovat ihned od začátku vývoje [3], respektive ihned po dokončení prvních návrhů.

Z počátku vývoje se často používají takzvané *lo-fi (low fidelity)* prototypy [2]. Zpravidla jde o návrh uživatelského rozhraní načrtnutý na několika papírech, kdy návrhář manipuluje s jednotlivými papíry, jako by se jednalo o reakci počítače na chování uživatele. *Lo-fi* testy mají za úkol odhalit chyby v požadavcích na uživatelské rozhraní a vyjasnit představy cílových uživatelů [4]. Oprava těchto chyb by byla v pozdějších fázích vývoje časově i finančně náročnější než na počátku [4].

V pozdějších fázích testy probíhají na takzvaných *hi-fi (high fidelity)* prototypech. Tyto prototypy vzhledově odpovídají hotovému uživatelskému rozhraní. Na jejich sestavení se často používají prototypovací jazyky. Návrhář musí počítat s tím, že *hi-fi* prototypy se zpravidla po dokončení testování zahazují [2]. *Hi-fi* prototypy by měly obsahovat reálná data, aby co nejméně mátla uživatele a testované rozhraní působilo co nejvíce věrohodně. Při testování sedí uživatel ve specializované místnosti a návrhář je sleduje z vedlejší místnosti nebo sedí uživatel společně s návrhářem před počítačem [2]. Ve specializované místnosti se nachází zařízení, které zaznamenává uživatelovy reakce. Kromě kliků a vyplňovaných dat se často sleduje i pohyb očí uživatele. Díky tomu může návrhář zjistit, na jaké objekty a v jakém pořadí se uživatel díval. Návrhář by také v průběhu testování měl zjišťovat, jak uživatel vnímá rozhraní pocitově. Tyto informace pomáhají při optimalizaci uživatelského rozhraní [2].

V obou případech uživatel postupuje podle předem nachystaného scénáře a plní úkoly [2]. Návrhář sleduje uživatelovy reakce, ale nesmí ovlivňovat uživatele svým pohledem na

problematiku. Nejlepších výsledků lze dosáhnout, pokud se testování provádí na cílových uživatelích [2]. Po dokončení testů se ověřuje, zda jsou již dostatečně splněny požadavky. Pokud ne, poznatky z testování promítneme do návrhu a celý cyklus se opakuje. V opačném případě se návrh uživatelského rozhraní může považovat za dokončený.

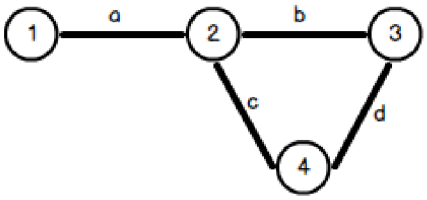
# Kapitola 3

# Vybrané kapitoly z teorie grafů

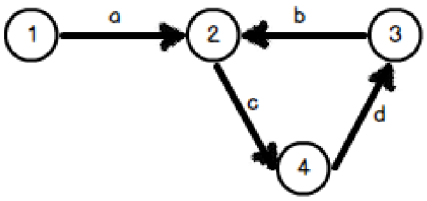
V této práci budu ve velké míře pracovat s rozvrhy hodin učitelů a tříd. Zejména při vyhledávání náhradních variant pro přesuny hodin v rozvrhu bude vhodné uvažovat o rozvrhu jako o grafu. Z tohoto důvodu zde popíši některé pojmy a algoritmy z teorie grafů. Přehled znalostí uvedených v této kapitole jsem čerpal z knihy o diskrétní matematice od autorů Matouška a Nešetřila [8].

### Obecné grafy

Za obecný graf (obrázek 3.1), někdy také nazýván jako neorientovaný graf, považujeme dvojici $G = (V, E)$. Kde $V$ je konečná množina uzlů grafu a $E = \{\{u, v\} : u, v \in U \wedge u \neq v\}$ je konečná množina hran mezi jednotlivými uzly. O hraně $e = \{u, v\}$ říkáme, že hrana $e$ spojuje uzly $u$ a $v$.




Obrázek 3.1: Obecný graf




Obrázek 3.2: Orientovaný graf

### Orientované grafy

Obdobně jako obecný graf je i orientovaný graf (obrázek 3.2) dvojice $G = (V, E)$. Kde $V$ je konečná množina uzlů grafu a $E = \{\{u, v\} : u, v \in U \wedge u \neq v\}$ je konečná množina orientovaných hran (šipek) mezi jednotlivými uzly. O šipce $e = \{u, v\}$ říkáme, že šipka $e$ vychází z uzlu $u$ a končí v uzlu $v$.

## 3.1 Důležité pojmy z oblasti grafů

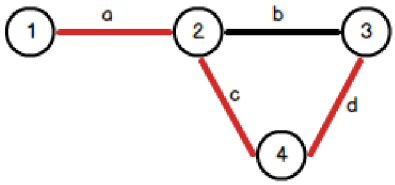
V grafech můžeme rozpoznat některé další útvary. Zejména zde uvedu co je to sled, cesta, kružnice a co znamená pojem souvislost grafu. Dále zde bude popsáno, co je to podgraf. Tyto pojmy nejspíše využiji při hledání různých variant změn rozvrhu.

### Sled

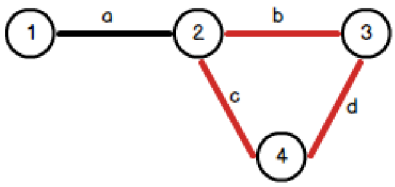
Pokud graf $G = (V, E)$, pak posloupnost $(v_0, e_1, v_1, \ldots, e_t, v_t)$ kde $\{v_0, v_1, \ldots, v_t\} \subseteq V$ a $\{e_0, e_1, \ldots, e_t\} \subseteq E$, můžeme nazvat sledem. Ve sledu se mohou uzly i hrany libovolněkrát opakovat. Podle počtu hran, přes který sled vede, můžeme určit délku sledu.

### Cesta

Na rozdíl od sledu se v cestě může uzel a hrana vyskytovat maximálně jednou. Pokud mezi dvěma uzly v grafu existuje sled, pak v něm lze nalézt i cestu mezi těmito dvěma uzly. Cestu si můžete prohlédnout na obrázku 3.3.




Obrázek 3.3: Cesta z uzlu 1 do uzlu 3



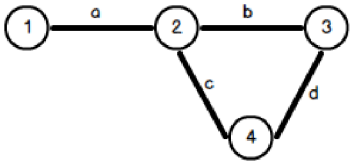

Obrázek 3.4: Kružnice o délce $t = 3$

### Kružnice

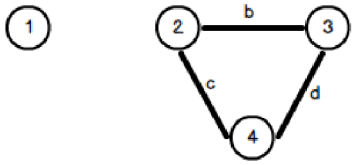
Pokud graf $G = (V, E)$, pak stejně jako cesta je i kružnice (obrázek 3.4) posloupnost $(v_0, e_1, v_1, \ldots, e_{t-1}, v_{t-1}, e_t, v_0)$ (počáteční a koncový uzel je shodný) kde $\{v_0, v_1, \ldots, v_t\}$ jsou navzájem různé uzly grafu G a $e_i = \{v_{i-1}, v_i\} \in E$ pro $i = 1, 2, \ldots, t-1$, a také $e_t = \{v_{t-1}, v_0\} \in E$ [9]. Číslo t je délka kružnice.

### Souvislý graf

Graf můžeme nazvat souvislý, pokud mezi jakýmikoliv dvěma uzly existuje sled. Obrázek 3.5 znázorňuje souvislý graf oproti obrázku 3.6, na kterém se nachází nesouvislý graf.



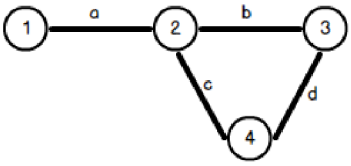
Obrázek 3.5: Souvislý graf



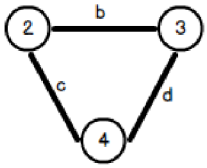
Obrázek 3.6: Nesouvislý graf

### Podgraf

O grafu $H = (V_H, E_H)$ lze prohlásit, že je podgrafem grafu $G = (V_G, E_G)$, jestliže $V_H \subseteq V_G$ a současně $E_H \subseteq E_G$. Podgraf (obrázek 3.8) tedy vznikne tak, že z původního grafu (obrázek 3.7) odebereme některé hrany a vrcholy.



Obrázek 3.7: Původní graf G



Obrázek 3.8: Podgraf H grafu G

## 3.2 Algoritmy pro hledání cest v grafu

V následující části popíši některé algoritmy pro hledání cest (ať už jakékoliv nebo té s nejlepším hodnocením) v grafu, popřípadě algoritmy pro prohledávání stavového prostoru. Z těchto algoritmů pak budu čerpat při tvorbě algoritmu pro hledání variant výměny hodin v daném rozvrhu.

### Dijsktrův algoritmus

Algoritmus pro hledání všech nejkratších cest v grafu. Vymyšlen nizozemským informatikem Edsgerem Dijkstrou v roce 1959. Jedná se o nejrychlejší známý algoritmus pro hledání nejkratších cest [10]. Jeho slabinou však je, že lze použít pouze pro grafy s nezáporně hodnocenými hranami.

Princip algoritmu spočívá v sestavení dvou front. Říkejme jim například *OPEN* a *CLOSED*, kde do fronty *OPEN* vložíme všechny uzly. V první iteraci má pouze zdroj vzdálenost 0 a ostatní uzly vzdálenost nekonečně velkou. V každé další iteraci se z fronty vybere uzel s nejmenší vzdáleností a všichni jeho potomci se vloží do fronty *OPEN*. Pokud se již tento uzel ve frontě *OPEN* nalézá, porovná se, zda by jeho nová vzdálenost nebyla menší, než s jakou vzdáleností se ve frontě *OPEN* právě nachází. Provede se tedy porovnání $vzdalenost_{zpracovany} + delkaHrany_{zpracovany,potomek} < vzdalenostpotomek$. Pokud by nově přidaný uzel dosahoval lepší vzdálenosti, provede se výměna uzlu ve frontě *OPEN*. Zpracovaný prvek je přesunut do fronty *CLOSED*. Následně se z fronty *OPEN* opět vybere uzel

s nejkratší vzdáleností. Celý algoritmus se opakuje až do doby, než se celá fronta *OPEN* vyprázdní [10].

Časová složitost tohoto algoritmu je závislá na implementaci fronty *OPEN*, kdy v případě sekvenčního vyhledávání ve frontě je složitost algoritmu $O(|U|^2)$ v případě binárního vyhledávání je složitost $O(|H| \log_2 |U|)$ [10].

### Depth-first search

Algoritmus též nazývaný jako prohledávání do hloubky. Slouží k průchodu stavového prostoru s jistotou průchodu všemi uzly [10]. Algoritmus prochází uzly postupně tak, že začíná na počátečním uzlu a rozgeneruje veškeré jeho potomky, na které rekurzivně zavolá stejnou funkci, tak jako by byly tyto uzly počáteční. Po návratu z rekurze označí právě projitý uzel za uzavřený. Tímto způsobem tedy projde veškeré možné uzly, do kterých vede libovolně dlouhá cesta z počátečního uzlu.

Jak již popis algoritmu napovídá, jeho asymptotická složitost je $O(|H| + |U|)$ [10].

### Backtracking

Backtracking, neboli slepé prohledávání se zpětným navracením, je algoritmus pro prohledávání stavového prostoru. Tento algoritmus je podobný prohledávání do hloubky (kontrola všech uzlů). Jeho vylepšení spočívá v odstranění některých uzlů, pokud se při zpracování daného uzlu ukáže, že žádný jeho následovník nemůže splnit požadované kritérium.

Na začátku algoritmus sestrojí zásobník *OPEN* a umístí do něj počáteční uzel. Následně bude brát ze zásobníku *OPEN* vždy jeden uzel, u kterého nejprve ověří, jestli splňuje cílové vlastnosti. Pokud ano, prohlásí ho za výsledný. Pokud ne, daný uzel je rozgenerován. To znamená, že všechny uzly, do kterých vede hrana z daného uzlu, jsou vloženy na vrchol zásobníku. Uzel, který nesplňuje dané kriterium a není ho možné rozgenerovat (popřípadě je jasné, že jeho rozgenerováním taktéž nedostaneme cílový uzel), odstraníme ze zásobníku a pokračujeme následujícím uzlem z vrcholu zásobníku [11].

Časová složitost tohoto algoritmu je dána vztahem $O(b^{\frac{b}{m}})$ kde b je maximální počet následovníků a m je počet všech různých stavů [11].

# Kapitola 4

# Nástroje pro tvorbu náhradních rozvrhů

V současnosti existuje několik aplikací pro práci s rozvrhy. Většina z nich je však placená. Vyzkoušel jsem proto dvě nejrozšířenější aplikace. Jednalo se o software Bakaláři s demo licencí a software SAS, který mi k těmto účelům zapůjčila společnost MP-Soft, a.s. Při konzultacích s pracovníky škol, kteří se tvorbou rozvrhu zabývají, jsem zjistil, že ačkoliv využívají jeden z výše uvedených systémů, přesto náhradní rozvrh tvoří vlastní silou za pomocí jiných pomůcek. V první podkapitole jsou uvedeny některé zákonné kritéria pro tvorbu náhradních rozvrhů. Další z podkapitol je věnována pohledu pracovníků tvořících rozvrh na problematiku a jejich požadavkům na výslednou aplikaci. Při porovnání informačních systémů jsem kromě uživatelského rozhraní zohlednil i postup při tvorbě rozvrhů pracovníky škol.

## 4.1 Omezení z pohledu zákona

Na začátek je nutné uvést, že většinu kritérií pro tvorbu náhradních rozvrhů si určuje každá škola sama. Velká variabilita těchto možností, není nijak omezená zákonem. Jediná zákonem ošetřená oblast je finanční odměňování pedagogických pracovníků.

Činnost pedagogického pracovníka je dle náležející finanční odměny rozdělena do dvou skupin [12]. Jednou z nich je přímá pedagogická činnost. Jedná se v podstatě o přímou výuku žáků. Všechno ostatní je práce související s pedagogickou činností. Do této skupiny kromě přípravy na hodinu (oprava prací žáků apod.) spadá i dozor nad žáky školy (např. na chodbách, ale i na soutěžích, výletech, exkurzích apod.). V součtu by měly obě tyto skupiny dosahovat dotace 40 hodin týdně. Tedy jako klasický pracovní úvazek.

Dotaci přímé pedagogické činnosti určuje ředitel školy a měla by být pro všechny pedagogické pracovníky dané školy shodná [12]. V případě, že je s učitelem sjednám pracovní úvazek v jiném rozsahu než plném, je tomu upravena i dotace přímé pedagogické činnosti tohoto pracovníka [12]. Dle § 3 odst. 3 vyhlášky č. 263/2007 Sb. může ředitel rozvrhnout přímou pedagogickou činnost na jednotlivé dny. Obdobně se v jednotlivých týdnech může rozvržená dotace přímé pedagogické činnosti lišit, avšak nejvýše tak, aby nebyl překročen průměrný stanovený týdenní rozsah přímé pedagogické činnosti za období 5 po sobě následujících měsíců.

Z pohledu finančního je přímá pedagogická činnost nad rámec stanoveného rozsahu (též nazývaná přespočetná) ohodnocena dvojnásobným příplatkem dle průměrné hodinové mzdy

pracovníka [12]. Ředitel školy může pracovníkovi nařídit maximálně 4 přespočetné hodiny (na dalším je možné se s pracovníkem dohodnout) [12]. V případě nerovnoměrného rozvržení přímé pedagogické činnosti se jako přespočetná počítá každá hodina nad rámec stanoveného rozsahu pedagogické činnosti pro daný týden. V praxi se zpravidla vyskytují dva typy přespočetných hodin. Prvním typem jsou hodiny pravidelné, které jsou vykonávány pravidelně každý týden dle rozvrhu. Druhým typem jsou takzvané nahodilé přespočetné hodiny, které vznikají při zastupování jiných pedagogických pracovníků (tzv. suplování) [13]. Z finančního hlediska jsou však oba tyto typy hodin ohodnoceny stejně.

Pokud pracovník nesplní úvazek z důvodu, že vykonával činnost související s přímou pedagogickou činností namísto přímé pedagogické činnosti, nejsou nadúvazkové hodiny počítané jako přespočetné, až do rozsahu takto neodučených hodin [12]. Opačným případ nastává, pokud přímá pedagogická činnost učiteli odpadne například kvůli dovolené nebo svátku. V takovém případě se hodiny přímé pedagogické činnosti v daném týdnu počítají jako odučené [12].

Z pohledu zákona se tak jeví jako kritické rozlišit, zda se jedná o přímou pedagogickou činnost či činnost pouze související. O typu činnosti rozhoduje zpravidla ředitel. V některých případech je takové rozdělení jasné. Například pokud učiteli odpadne hodina z důvodu nepřítomnosti třídy nebo pokud učitel provádí dozor nad žáky na chodbě. Náročnější případ nastane v případě exkurzí a podobných akcích, na kterých učitel doprovází žáky. V těchto případech je rozhodující právě přímé pedagogické působení na žáky [13].

## 4.2 Hlavní kritéria z pohledu pracovníka školy

V této kapitole jsou uvedeny postupy a kritéria, která byla nejčastěji zmiňována při konzultacích s pracovníky škol a které jsem vyhodnotil jako nejdůležitější.

Při tvorbě náhradního rozvrhu postupuje pracovník nejprve tak, že vytvoří seznam chybějících učitelů a tříd. V této fázi by měl mít pracovník přehled o tom, kolik kdo odučil hodin nad rámec úvazku nebo naopak, kolik kterému učiteli chybí odučit pro naplnění úvazku. Dále musí také pracovník vědět, jak často odpadaly konkrétní předměty. K tomu zpravidla slouží další listy papíru s poznámkami nasbírané v průběhu celého školního roku (pro předměty) nebo měsíce (pro úvazky učitele). Poté pracovník prochází rozvrh chybějícího učitele a snaží se nalézt zastupujícího učitele podle různých kritérií. Nejprve se rozhodne (na základě počtu dříve odpadlých hodin, podle preference školy, ...) jestli je potřeba hodinu odučit nebo stačí dozorovat. Následně vybírá vhodného učitele. Nejdříve se snaží obsadit pedagogického pracovníka, kterému odpadly některé hodiny. Může se ovšem stát, že ani jedna varianta z preferovaných řešení není dostupná (není volný aprobovaný učitel, není volný učitel s nesplněným plánem přímých vyučovacích hodin, ...) a tak se při tvorbě náhradního rozvrhu často pracovník spoléhá na svůj úsudek. Mnohdy se taktéž stává, že sami učitelé nebo žáci přijdou s návrhem změny rozvrhu, který by jim více vyhovoval (výměna 2 předmětů, přesunutí do volného okna, ...). Pracovník, který rozvrh vytvořil, totiž zpravidla nemá přehled o všech rozhodujících podrobnostech.

Plánování náhradního rozvrhu pracovníci škol rozdělili na dva základní postupy. Při prvním je suplování plánováno dopředu (minimálně den před platností plánovaného rozvrhu). Obecně je u tohoto přístupu více možností jak daného pracovníka nahradit. Druhý postup nastává v případě neočekávané nepřítomnosti pedagogického pracovníka (například z důvodu náhlého onemocnění). V tomto okamžiku nezbývá mnoho možností jak pracovníka zastoupit. Musí být počítáno s tím, že se na danou hodinu nemohou připravit ani žáci ani suplující učitel. Některé školy mají pro tyto situace sestavenu tzv. „suplovací pohoto-

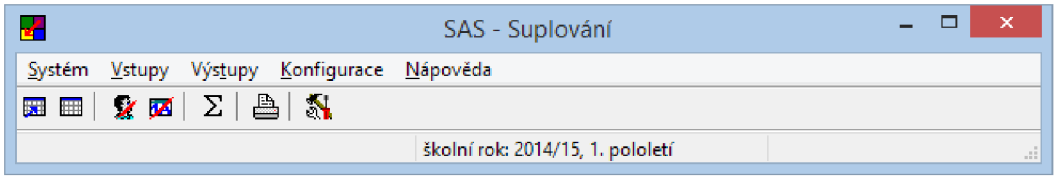
vost“. Takto nahrazená hodina je však spíše dozorovaná než odučená, a proto některé školy preferují odpadnutí takové hodiny.

Pokud ale hodina odpadne, je potřeba kontrolovat, zda neodpadává příliš často. Zejména pokud jde o předmět vyučovaný pouze jednu hodinu týdně. V takovém případě by vícenásobné zrušení tohoto předmětu mohlo znemožnit splnění učebního plánu. Hodina by taktéž neměla odpadnout uprostřed učebního bloku. Řada škol proto odpadání hodin umožňuje pouze u prvních nebo posledních hodin. V případě, že hodina odpadne nezletilým žákům, musí škola včas informovat rodiče.

Z finančního pohledu, který je pro většinu škol nejdůležitější, je nejvhodnější nasazovat učitele, kterým dříve odpadla hodina. V takovém případě totiž nenáleží učiteli příplatek dle zákona. Finanční kritérium má zpravidla přednost i před aprobovaným učitelem z daného předmětu. Aprobované učitele pracovník školy do náhradního rozvrhu taktéž nasazuje podle určitých kritérií. Nejprve podle toho jestli učitel daný předmět na škole vyučuje (nemá význam nasazovat učitele, který předmět několik let neučil) a poté jestli učitel učí ve třídě (znalost prostředí, ve kterém bude vyučovat).

## 4.3 SAS – Systém agend pro školy

Tuto aplikaci vyvíjí firma MP-Soft, a.s.[1] Je to rozsáhlý systém, který svou funkčností pokrývá většinu potřeb základní a střední školy. Modul pro tvorbu náhradních rozvrhů se nazývá „Suplování“. Všechny moduly v SAS mají stejnou nabídku a jsou si hodně podobné. Vzhled tohoto modulu lze prohlédnout na obrázku 4.1. Slouží spíše jako ovládací panel, jehož prostřednictvím jsou zobrazeny další okna. Názvy některých položek nabídky jsou nejasné a myslím, že stejná terminologie napříč moduly neodpovídá funkčnosti modulu.



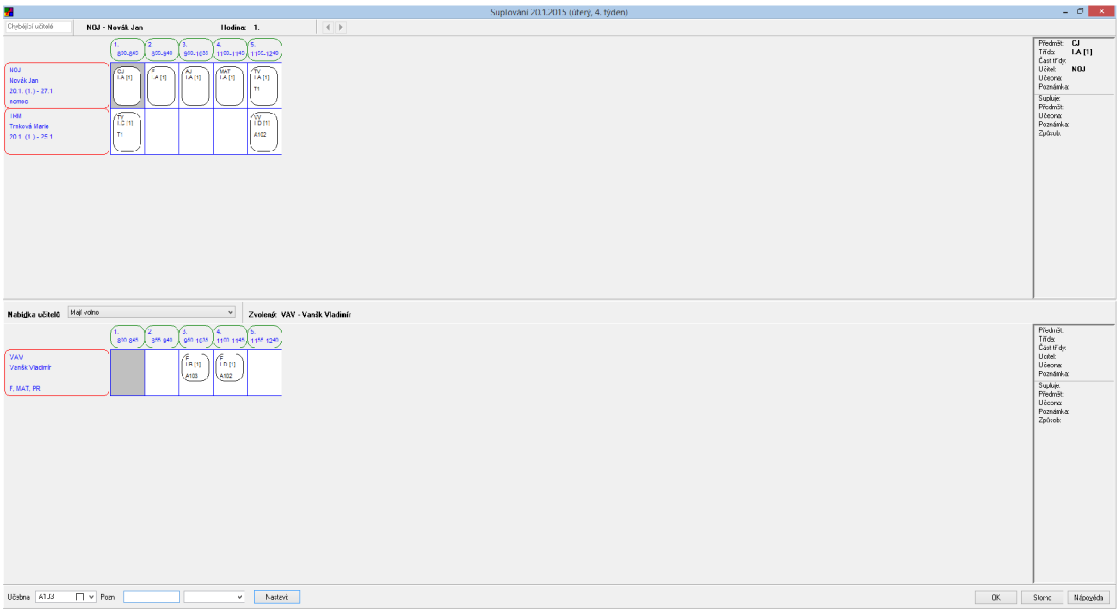

Obrázek 4.1: Modul Suplování

Kromě samostatné tvorby náhradního rozvrhu umožňuje modul uživatelům zadat chybějící učitele nebo třídy. Zadávání absencí není nikterak složité. Zvolí se pouze učitel nebo třída, která chybí a zadají se informace o začátku a konci absence. Zadává se den a hodina počátku a konce nepřítomnosti. Taktéž je dostupná kolonka pro zadání důvodu absence, ale ta má spíše informativní charakter. Více by se hodilo zadat, zda takováto absence vznikla například z důvodu výjezdu se třídou na nějakou akci, či jde o dovolenou nebo návštěvu lékaře.

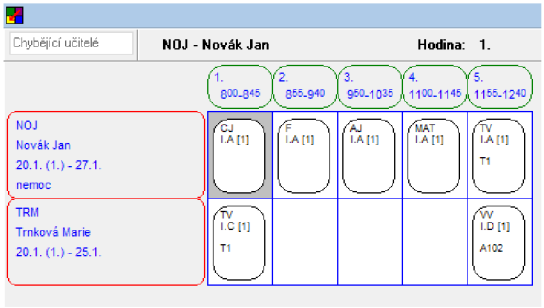
Před zobrazením okna pro tvorbu náhradního okna musí uživatel zvolit datum, na které se bude náhradní rozvrh vztahovat. Samostatná tvorba náhradního rozvrhu se odehrává v jednom okně na obrázku 4.2. Okno suplování je rozděleno do několika částí. První z nich jsou nepřítomní učitelé (obrázek 4.3). Výběrem jedné z hodin se zobrazí učitelé podle zvoleného kriteria (obrázek 4.4). Nabídku kriterií je možné měnit v nastavení modulu. Lze vytvořit kritérium na základě hodiny (mají volno, volno na oběd, odpadla jim hodina, ...),

[1] http://www.mp-soft.cz/sas/

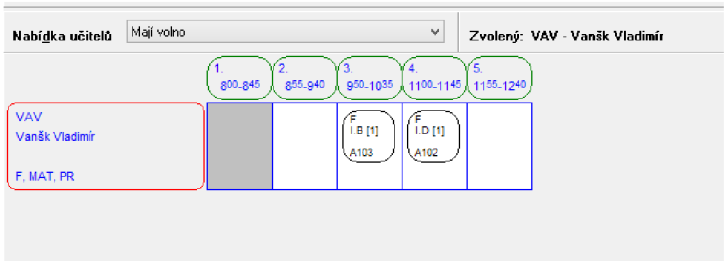
předmětu (mohou učit, mají aprobaci, . . . ), třídy (učí ve třídě, učí část třídy, . . . ) a ostatních možností (odpadla jim hodina, nemají splněný počet odučených hodin). Vytvořené kritérium lze pojmenovat a je pak zobrazeno v comboboxu pro filtrování náhradních učitelů. Potvrzení zvoleného učitele se provede v poslední části okna 4.5. Zde se nachází pole pro volbu učebny, zadání poznámky a výběr typu suplování z číselníku. Tato potvrzovací část okna je poměrně malá a nenápadná vzhledem k celkové velikosti okna.



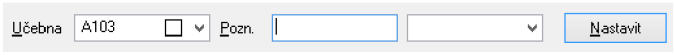
Obrázek 4.2: Okno pro tvrobu náhradního rozvrhu




Obrázek 4.3: Nepřítomní učitelé



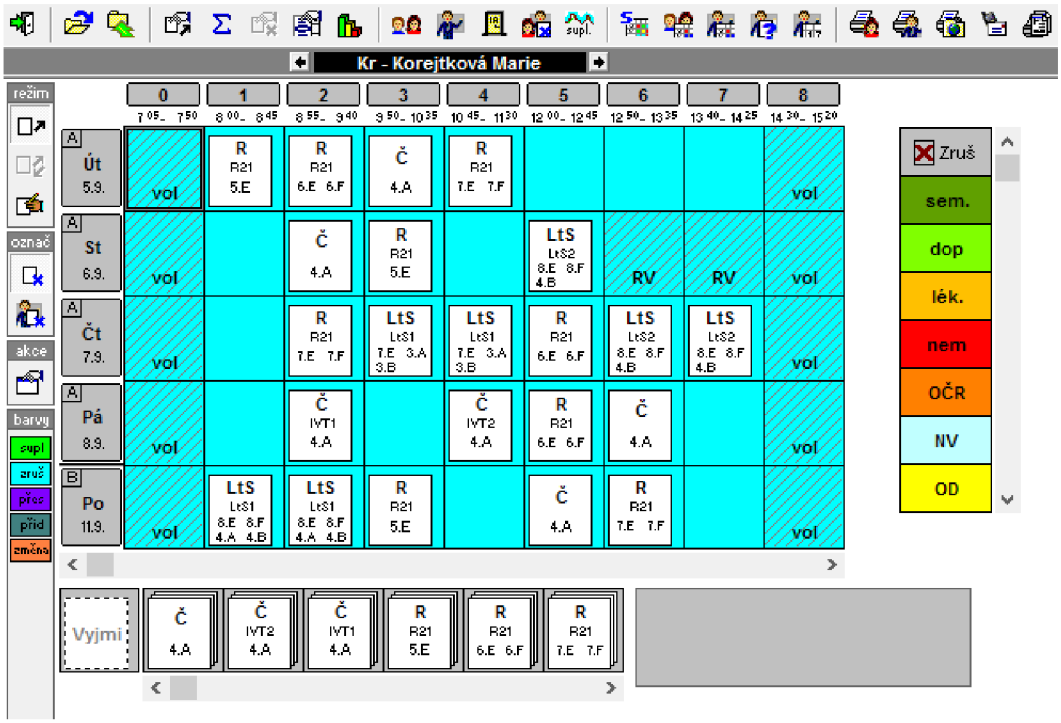

Obrázek 4.4: Náhradní učitelé




Obrázek 4.5: Nastavení zvolené varianty

Aplikace zhruba splňuje potřeby učitelů. Nejvíce asi schází přehled o odučených hodinách pedagogických pracovníků. Praktické by taktéž bylo zobrazit rozvrh dané třídy, která může pomoct pracovníkovi s výběrem varianty suplování. V podstatě není možné pomocí tohoto okna vyměnit dvě hodiny mezi sebou. Popřípadě změnit vyučovací hodinu. Z hlediska uživatelského rozhraní se na formuláři nachází velká nevyužitá plocha. Dalším problémem by mohl být velký počet filtrů. V nastavení je však možné smazat některé filtry, popřípadě si vytvořit vlastní.

## 4.4 Bakaláři

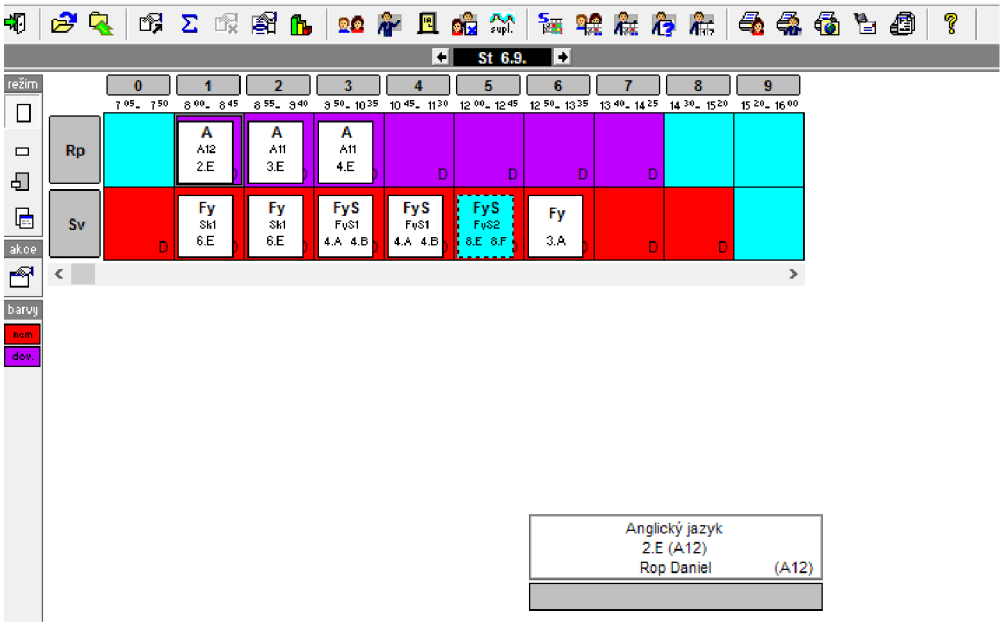



Obrázek 4.6: Rozvrh učitele

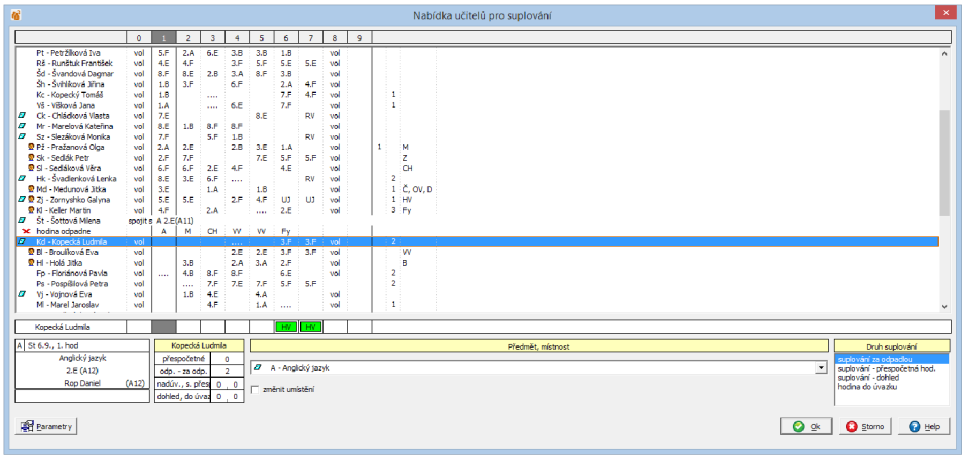
Stejně jako předchozí software jsou Bakaláři[2] velice rozsáhlý systém s mnoha funkcemi. Hlavní nabídka zde chybí a je nahrazena zástupci, pomocí kterých se spouští jednotlivé moduly. Těchto modulů je 22, což je opravdu hodně. jedním z nich je modul suplování.

Ihned po spuštění modulu se zobrazí okno pro vybrání dnů popřípadě týdnů, kterých se náhradní rozvrh bude týkat. Po zobrazení modulu se zobrazí okno pro práci s rozvrhy. Nejprve je nutné vybrat učitele a označit nepřítomnost učitelů (viz obrázek 4.6). Ve sloupci napravo lze vidět druhy nepřítomnosti. Dle zvoleného důvodu se pak počítá množství odpadlých hodin pro případné suplování. Nepřítomnost celých tříd se zadává stejným způsobem.

[2] http://www.bakalari.cz




Obrázek 4.7: Nepřítomní učitelé




Obrázek 4.8: Výběr zastupujícího učitele

Po nastavení nepřítomnosti se pomocí nástrojové lišty uživatel přesune k tvorbě náhradního rozvrhu (viz obrázek 4.7). Na rozdíl od rozvrhu učitele, popřípadě třídy, jsou zde zobrazeni učitelé po jednotlivých dnech. Stejné prvky jsou tedy v různých situacích využity jinak, což může být pro uživatele problematické. Po kliku na hodinu u chybějícího učitele se zobrazí seznam všech učitelů s informací o jejich výuce v celém dni. Okno je znázorněno na obrázku 4.8. Velké množství učitelů může komplikovat výběr. V okně je sice zobrazena statistika (přespočetné hodiny, odpadlé hodiny, ...) učitele, ale až po jeho označení. Proto má taková statistika spíše informativní než rozhodovací charakter.

Aplikace tedy obsahuje všechny informace, podle kterých se v současné době pracovník školy při tvorbě rozvrhu rozhoduje. Avšak z hlediska uživatelského rozhraní jsou tyto informace obtížně dostupné. Nejvíce problémové je asi zobrazení jen jednoho rozvrhu. Dalším problémem je nutnost prohlédnout všechny dostupné učitele, aby se zobrazily statistiky

o přespočetných hodinách. Místo všech učitelů by bylo lepší věnovat více místa jen volným učitelům. Nevyužité místo na základním formuláři by mohlo být vyplněno více rozvrhy.

Celkově se s aplikací pracovalo obtížně. Zabralo mi mnoho času nalézt některé funkce pro nahrazení učitele. Většina nástrojů se nachází na nástrojové liště bez popisků. Navíc všechny jsou si velice podobné a činilo mi problémy je od sebe opticky rozlišit. Nástrojová lišta aplikace je viditelná na obrázcích 4.6 a 4.7.

# Kapitola 5

# Návrh aplikace pro tvorbu náhradních rozvrhů

V této kapitole jsou shrnuty počáteční požadavky na celou aplikaci. První část tovří analýza školního prostředí a cílových uživatelů. V dalších částech kapitoly je popsán proces návrhu uživatelského rozhraní softwaru pro tvorbu náhradního rozvrhu. Na základě požadavků pracovníků škol a specifických kritérií, kdy se při tvorbě rozvrhu pracovníci často řídí jinými pravidly pro každou hodinu, jsem se rozhodl, že proces tvorby náhradního rozvrhu nebude plně automatický, ale bude vyžadovat částečnou interakci uživatele.

## 5.1 Analýza školního prostředí

Základní informace o uživatelích a školním prostředí jsem zíkal na konzultací ve vybraných školách s lidmi, kteří tvoří náhradní rozvrhy. Další informace jsem pak čerpal od zaměstnanců firmy MP-Soft, a.s. Při analyzování jsem postupoval podle principu PACT:

- **Lidé** – Typickým uživatelem nové aplikace bude učitel, který využívá osobní počítač nebo notebook. Při ovládání počítačů preferuje myš. Uživatel umí pracovat se systémem SAS. Předpokládám tedy, že se jedná o mírně pokročilého uživatele. Většinou bude na škole s výslednou aplikací pracovat jeden člověk, který má i v současnosti na starost tvorbu náhradních rozvrhů. Nicméně uživatelské rozhraní by mělo být jednoduché, aby práci s ním zvládl i kdokoliv jiný v případě nepřítomnosti zodpovědného pracovníka. Většina učitelů je české národnosti [14]. Předpokládám tedy, že pro tuto skupinu uživatelů platí stejné vlastnosti jako pro jiné lidi české národnosti.
- **Aktivity** – Nejčastěji bude uživatel s aplikací pracovat den až dva před platností náhradního rozvrhu. Nejprve zadá chybějící učitele a třídy, poté hledá zastupující učitele. Při výběru se rozhoduje podle vykonaných hodin přímé pedagogické činnosti a odbornosti učitelů. Přednost mají učitelé, kteří nemají odučený svůj základní úvazek.Je třeba počítat s tím, že po zveřejnění náhradního rozvrhu mohou přijít učitelé nebo žáci s vlastní variantou, která je pro ně z nějakého důvodu výhodnější. Druhé použití aplikace nastane v případě náhlé nepřítomnosti učitele. V takové situaci se nejprve zvolí učitel dle suplovací pohotovosti (nemusí být na škole zavedena). Pokud nelze vybrat zastupujícího učitele se suplovací pohotovostí, musí pracovník školy zvolit variantu s co nejmenším dopaden na rozvrh (nelze přehazovat předměty, přesouvat hodiny, ...). Často v takových případech dochází k dozorování ve vybrané hodině či

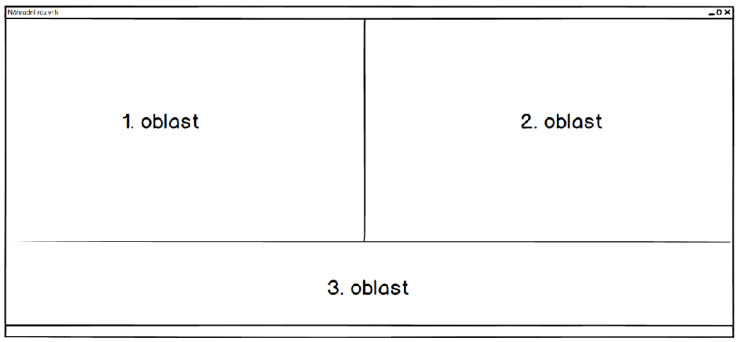
k odpadnutí hodiny (problémové u nezletilých dětí). Při tvorbě náhradního rozvrhu je však důležité hlídat, aby jeden předmět neodpadal příliš často.

- **Souvislosti** – Zpravidla bude stačit s aplikací pracovat jednou denně. V ojedinělých případech i vícekrát za den, podle toho, jak se bude měnit situace. Je důležitý zásah učitele do průběhu tvorby rozvrhu hlavně z důvodu lidského faktoru (někteří učitelé budou rádi za příplatek z přespočetných hodin, jiní bude zase vděční za více volna). Nejčastěji bude aplikace nainstalována na stolním počítači v kabinetu. Společně se systémem SAS.

- **Technologie** – Rozhodl jsem se navrhnout uživatelské rozhraní pro operační systém Windows, protože na školách je tento OS nejrozšířenější. Jako jazyk implementace jsem zvolil Delphi kvůli pozdější integraci se systémem SAS a jejich jednoduššímu provázání. Navržené rozhraní optimalizuji pro obrazovky s rozlišením 1366 x 768 pixelů a více, protože se jedná o nejrozšířenější rozlišení [15]. Uživatel bude aplikaci ovládat zejména pomocí myši. Méně často pak využije klávesnici. Jako datová základna poslouží existující databáze Firebird, kterou využívá již hotová aplikace SAS. Připojení na server s databází bude dostupné prostřednictvím sítě internet nebo vnitřní sítě školy.

## 5.2 Uživatelské rozhraní

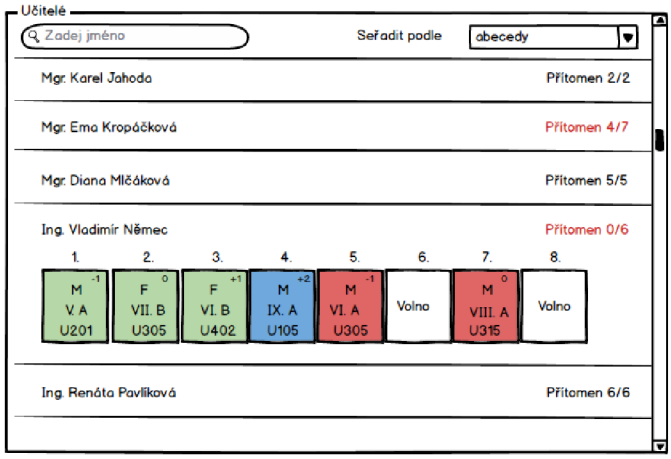
Při návrhu uživatelského rozhraní jsem vycházel z výše uvedené analýzy a z požadavků uživatelů. Rozhodl jsem využít jeden hlavní formulář a tři vedlejší pro zadávání absencí učitelů nebo tříd a pro volbu data náhradního rozvrhu. V první fázi jsem rozdělil formulář do tří oblastí (viz obrázek 5.1). V první oblasti bude rozvrh chybějících učitelů. Druhá část bude sloužit pro přehled rozvrhu třídy. V poslední třetí části by měly být dostupné různé varianty zastupování pedagogických pracovníků.




Obrázek 5.1: Rozvržení obrazovky

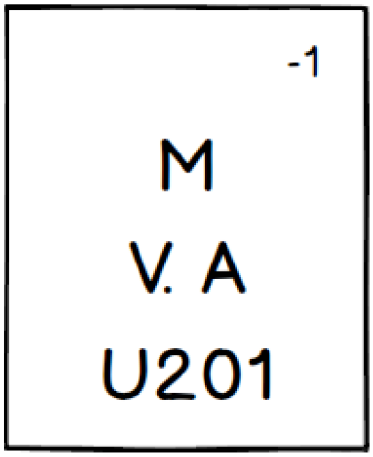
V první části se nachází seznam pedagogických pracovníků (viz obrázek 5.2). Zde by měl být zobrazen seznam všech pracovníků. Uživatel tak bude mít možnost zadat nepřítomnost učitele pomocí zvláštního formuláře, ale také vybrat přímo hodinu, ve které chybí (to bude vhodnější, pokud půjde třeba jen o jednu hodinu nebo obecně o menší počet hodin). Druhá verze tohoto návrhu počítá s tím, že by byli zobrazeni pouze chybějící učitelé. Poté by ale veškerou nepřítomnost musel uživatel zadávat přes zvláštní formulář. Po kliknutí na jméno učitele se zobrazí pod jménem jeho rozvrh na zvolený den. Zeleně jsou již vyřešené hodiny (nahrazené někým jiným), červeně pak hodiny, pro které musí uživatel vybrat zastupujícího

učitele. Modře podbarvená hodina je aktuálně vybraná hodina a jsou k ní zobrazeny další informace v oblastech popsaných níže.



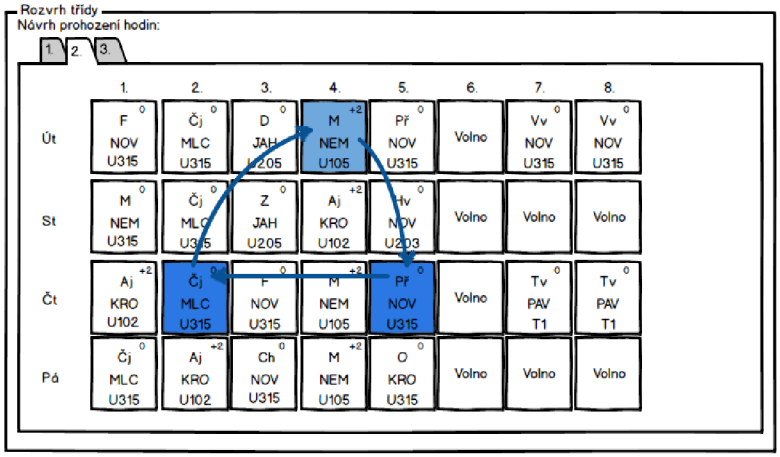

Obrázek 5.2: Oblast volby nepřítomného učitele

Jednotlivé hodiny jsou v celém návrhu zobrazené podobně. Základ tvoří obdélník, který obsahuje další informace. Vzhled hodiny u rozvrhu učitele je vidět na obrázku 5.3. V horním pravém rohu je uveden rozdíl hodin skutečně odučených vůči počtu hodin podle učebního plánu. Hodnota -1 znamená, že bylo odučenu o jednu hodinu méně než mělo. Plusové hodnoty pak, že je daný počet hodin odučen nad rámec učebního plánu. Tento případ není tak častý, ale může nastat při přesouvání hodin. Na řádcích je pak vypsána zkratka předmětu, zkrácený název třídy, název učebny.



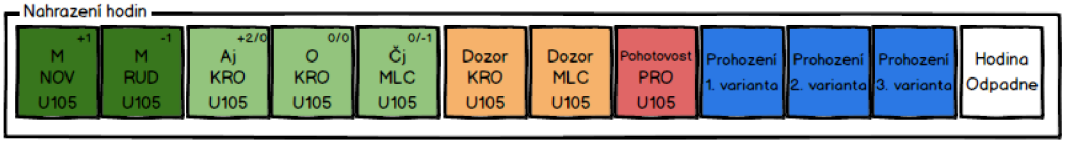

Obrázek 5.3: Hodina rozvrhu učitele

Ve druhé oblasti návrhu se zobrazí rozvrh třídy podle vybrané hodiny u nepřítomného učitele. Rozvrh třídy je znázorněn na obrázku 5.4. V této oblasti se zobrazí skutečný rozvrh třídy a možnosti prohození zvolené hodiny. V horní části je možné mezi jednotlivými variantami přepínat. Šipky pak naznačují výměnu hodiny. Při pohledu na rozvrh lze také lépe určit, zda se jedná o krajní hodinu dané třídy a zda je možné ji nechat odpadnout. Zobrazené hodiny mají stejný význam jako v rozvrhu učitelů, až na výměnu zkratky třídy za zkratku vyučujícího učitele.




Obrázek 5.4: Oblast rozvrhu třídy

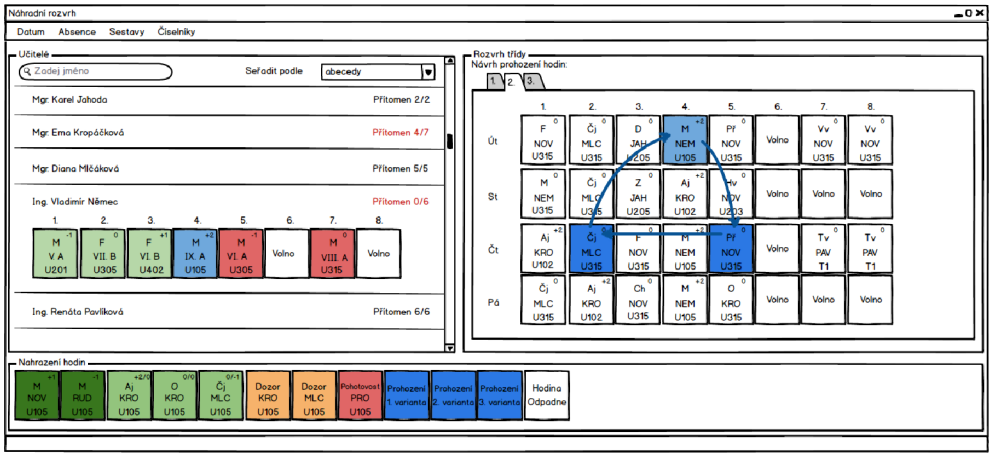
V poslední části obrazovky jsou zobrazeny vybrané možnosti pro zastoupení chybějícího učitele (viz obrázek 5.5). Uživatel v této části vybere jednu z variant změny. Jednotlivé skupiny variant jsou od sebe rozlišeny barevnou výplní. Tmavě zeleně je podbarvena odborně vyučená hodina při zachování předmětu. Světle zelená je taktéž hodina přímé pedagogické činnosti, ale se změnou předmětu oproti rozvrhu. Zde by stálo za zvážení, jestli ještě nevytvořit jednu skupinu, která by od sebe oddělila změnu za předmět dnes vyučovaný a ostatní předměty (na jednu stranu toto lze vypozorovat z rozvrhu zobrazeného v druhé oblasti návrhu, ale uživateli by to asi ulehčilo orientaci v případě zvolení jiné barvy). V této skupině jsou pouze učitelé, kteří v dané třídě vyučují (z pohledu škol nemá význam měnit předmět a odborně vyučovat někým jiným, než na koho jsou žáci zvyklí). Třetí skupinu tvoří oranžová políčka, ve kterých jsou uvedeni všichni učitelé, kteří mají v danou hodinu volno. Čtvrtá skupina (červené pole) souvisí s předcházející skupinou, ale jde o člověka, který má suplovací pohotovost. Pátou skupinu s modrou barvou je přehození podle zobrazených variant v druhé oblasti.




Obrázek 5.5: Varianty pro nahrazení hodiny

Náhradní varianty jsou hodně podobné s hodinami u dříve zmíněných rozvrhů. U tmavě zelených variant je v horním pravém rohu uveden počet přespočetných hodin přímé pedagogické činnosti. U světle zelených jsou uvedeny dvě čísla oddělené pomlčkou. První z nich nese stejnou informaci jako číslo u rozvrhu učitele nebo třídy, tedy statistiku předmětu. Druhé číslo za pomlčkou pak oznamuje počet přespočetných hodin daného učitele.

Na obrázku 5.6 se nachází celkový návrh obrazovky pro tvorbu náhradního rozvrhu. Kromě výše popsaných části je zde ještě zobrazena hlavní nabídka. V hlavní nabídce se nalézají položky pro zadávání absencí tříd nebo učitelů. Další položka slouží ke zvolení data náhradního rozvrhu.




Obrázek 5.6: Celkový pohled na navržené uživatelské rozhraní

## 5.3 Algoritmus výběru zastupujících pedagogů

Při práci s programem uživatel nejprve zadá vstupní údaje. Tedy chybějící třídy a učitele. Pro lepší výpočet přespočetných hodin je vhodné zadat tyto údaje na celé období, pro které bude vytvářen náhradní rozvrh. Poté se přesune na vytváření samostatného rozvrhu. Nejprve vybere učitele, za kterého chce vyhledat náhradu. Program zjistí z databáze jeho rozvrh na daný den. Podle třídní knihy spočítá pro každý předmět aktuální stav počtu hodin vzhledem k učebnímu plánu. Tyto informace zobrazí a uživatel zvolí konkrétní hodinu, za kterou chce hledat náhradu. Proces hledání náhradních variant můžeme rozdělit do několika částí.

**V první** části se vytvoří seznam učitelů, kteří mají danou hodinu volno (ať už z důvodu odpadnutí hodiny či volného místa v rozvrhu). Každému volnému učiteli se spočítá rozdíl hodin oproti jeho úvazku (přespočetné hodiny nebo naopak hodiny které schází do naplnění úvazku). U všech volných učitelů je pak zjištěno, zda učí daný předmět a zda učí ve třídě. Podle toho se vytvoří náhradní varianty pro suplování. V první skupině jsou učitelé, kteří učí ve třídě a mohou učit daný předmět. Do druhé skupiny patří učitelé, kteří sice učí daný předmět, ale neučí třídu, ve které mají zastupovat. Třetí skupinu tvoří učitelé, kteří sice neučí vybraný předmět, ale ve třídě učí nějaký jiný předmět. Do seznamu náhradních variant jsou tak přidáni tito učitelé vícekrát. Pokaždé s jiným předmětem, který ve třídě vyučují.

Všechny čtyři skupiny se zobrazí na formuláři v místě pro náhradní varianty. U každé varianty bude zobrazena zkratka předmětu, zkratka učitele, počet hodin do/nad úvazek a zda měl učitel volno v danou hodinu nebo zda měl v rozvrhu určený jiný typ hodiny (například suplovací pohotovost, oběd, zda mu hodina odpadla, ...). Pokud by vybranou náhradní variantou došlo ke změně předmětu, bude u této náhradní varianty zobrazen i rozdíl odučených hodin oproti plánovanému stavu dle rozvrhu.

**Druhou** částí algoritmu bude vyhledání hodin, na jejichž pozice je možné hodinu přesunout. Tedy na hodinu, ve které má učitel i třída volno. V této fázi se načte seznam hodin učitele a celé třídy. Vyhledají se volná okénka v rozvrhu. A tato okénka pro přesun se zobrazí společně s rozvrhem třídy.

Poněkud komplikovanější části algoritmu bude hledání hodin pro kruhovou výměnu. Nejprve je nutné z rozvrhu třídy vytvořit graf. Jako uzly grafu se budou brát jednotlivé hodiny rozvrhu. Zda existuje v grafu hrana mezi jednotlivými uzly se rozhodne dle toho, zda bude možné první hodinu odučit v době druhé hodiny. Pro vytvoření hran se tedy nejprve zjistí seznam volných hodin učitele v době hodiny zvolené třídy. Pokud je učitel dostupný pro výměnu, existuje v grafu orientovaná hrana, která směřuje z vybrané hodiny do hodiny v rozvrhu třídy, kdy má učitel volnou hodinu. Hrana takto nevznikne mezi hodinami jednoho učitele, i když by prohození bylo možné. Avšak z hlediska hledání náhradní varianty rozvrhu by to znamenalo akorát zvětšení délky kružnice. Ovšem je možné, že vznikne hrana již do uzlu, který byl do kružnice zahrnut. Tento stav bude ošetřen při procházení grafu algoritmem.

Protože hrany mezi jednotlivými uzly nakonec nebudou ohodnocené (v počátcích práce na tomto návrhu jsem uvažoval o vhodnosti umístění jednotlivých předmětů), zvolil jsem pro procházení grafu algoritmus backtracking (zpětné navracení). Oproti klasickému backtrackingu nebude algoritmus končit, pokud nalezne orientovanou hranu z aktuálního uzlu do počátečního uzlu. Pouze si tuto kružnici uloží a bude pokračovat rozgenerováním dalšího uzlu ve frontě. Algoritmus tedy projde všechny možnosti a nalezne veškeré kružnice. V případě vzniku dlouhé kružnice není taková kružnice žádoucí, protože značně znepřehlední celý rozvrh jak pro učitele, tak pro žáky. Většina učitelů se tedy shodla, že maximální délka kružnice je 4 hrany (počáteční uzel a tři další). Algoritmus tedy prohlásí danou kružnici za neexistující, pokud se dostane do čtvrtého uzlu a z toho uzlu nepovede hrana do počátečního. Naopak nejkratší možná kružnice může dosahovat délky 2 hrany (2 různé hodiny). V grafu se však kratší kružnice ani nacházet nemůže, protože při sestavování grafu nemůže existovat hrana mezi dvěma uzly stejného učitele.

**Algoritmus 1:** Hledání kružnic v rozvrhu

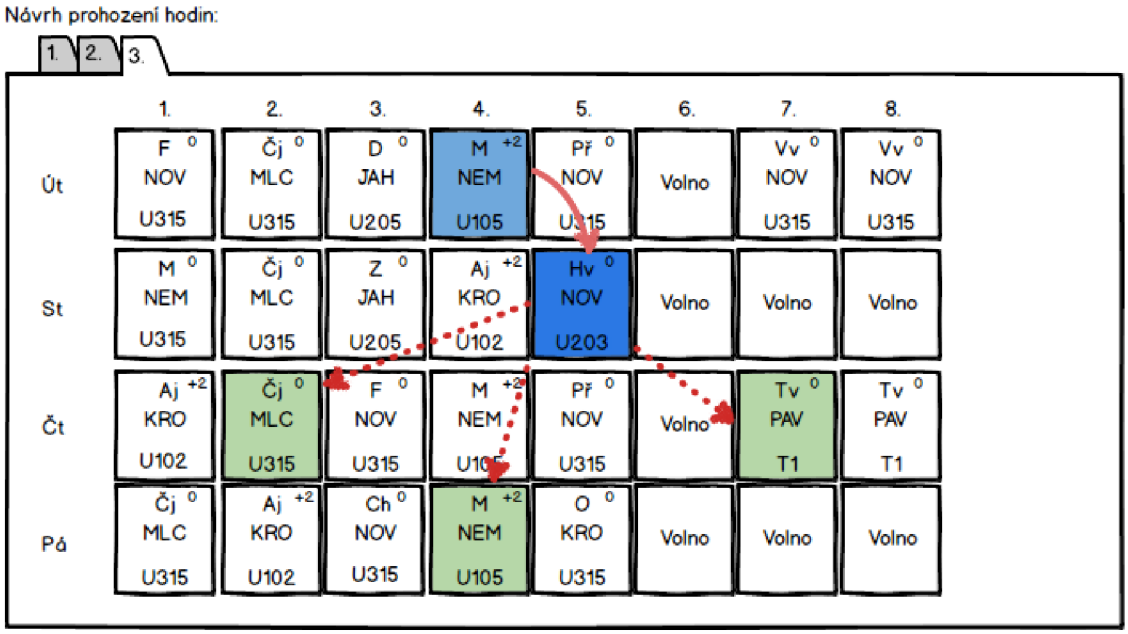
**Vstup**: Sestavený graf, aktuální hodina
**Výsledek**: Seznam kružnic v grafu

```
1  projdiGraf(AktualniHod,AktualniHod,1,[AktualniHod]);
2  Procedure projdiGraf(uzel zacatek, uzel u, int zanoreni, seznamUzlu cesta)
3  begin
4      if (u<>zacatek) or (zanoreni =1) then
5          if ExistujeHrana (u,zacatek) then
6              Vysledek.add(cesta);
7          else
8              if zanoreni < MAX_DELKA_KRUZNICE then
9                  forall the potomek in u.potomci() do
10                     projdiGraf(zacatek, potomek, zanoreni+1, cesta + [u]);
11                 end
12             end
13         end
14     end
15 end
```

Postup algoritmu 1 je znázorněn na obrázku 5.7. Kdy jako počáteční hodina byla označená úterní čtvrtá hodina. Z ní zvolil algoritmus hranu vedoucí do středeční páté hodiny. Nyní má tedy algoritmus na výběr jednu ze tří hran. Na obrázku 5.7 jsou tyto hrany zná-

zorněny červenou přerušovanou šipkou.

Stejně jako zastupující učitelé se i varianty prohození hodin zobrazí v sekci náhradních variant. Přesun hodin bude také nutné potvrdit v oblasti náhradních variant. K těmto možnostem navíc ještě přibude možnost nechat hodinu odpadnout. Pokud uživatel vybere jednu z možností uloží se do rozvrhu třídy a v možnostech se zobrazí volba pro vrácení takto vybrané hodiny do původního stavu. Uživatel tedy v průběhu tvorby náhradního rozvrhu může procházet už vyřešené nepřítomnosti a může zvolit jinou možnost řešení.




Obrázek 5.7: Ukázka kroku algoritmu pro hledání kružnice v rozvrhu hodin

## 5.4 Testování návrhu

Pro účely testování návrhu jsem v programu Balsamiq Mockups[3] vytvořil několik náčrtů obrazovek, mezi kterými se dalo kliknutím na objekt přecházet. Takto vytvořená množina náčrtů sloužila jako *lo-fi* prototyp aplikace. Jako testovací data jsem využil reálně vypadající jména učitelů, názvy tříd a předmětů. Takto vytvořenou ukázku programu jsem představil uživatelům na školách.

Samotný uživatel postupoval podle několika scénářů, kdy měl zaprvé zadat zástup za dlouhodobou a předem známou absenci učitele. Tedy v případě, kdy uživatel chybí nejméně celý den a tato absence je ohlášena minimálně dva dny předem. Druhým scénářem bylo vyřešit náhlou absenci učitele ohlášenou těsně před hodinou nebo neohlášenou vůbec. Poslední scénář počítal s úpravou již vytvořeného náhradního rozvrhu.

Testování probíhalo tak, že jsem před uživateli spustil prototyp programu. Poté jsem jej požádal, ať před kliknutím na jednotlivé prvky uživatelského rozhraní vyjádří své očekávání, co po kliknutí nastane.

V průběhu testování nedošlo k větším zádrhelům při práci s programem. Pouze z počátku bylo nutné uživatelům popsat, k čemu jednotlivé oblasti uživatelského rozhraní slouží.

[3] https://balsamiq.com/products/mockups/

Uživatelé poté již očekávali správné reakce systému na jejich interakci.

V závěrečném hodnocení pak uživatelské rozhraní hodnotili převážně kladně. Nicméně měli několik konstruktivních připomínek. První z nich byla absence dělených hodin v návrhu a s tím související možnost spojení dvou hodin v rámci suplování. Problémem byl taky poměrně malý počet hodin rozvrhu v jednotlivém dni, kdy jsem zvolil rozsah hodin od 1. do 8., ale na některých školách mají více vyučovacích hodin (například 0. až 12.). V takovém případě by se celý rozvrh hodin nemusel vlézt do k tomu určeného místa. Tento problém by mohlo vyřešit posouvání viditelné oblasti rozvrhu dle aktuální potřeby uživatele. Některým uživatelům se taktéž zdály nejasné popisy u hodin. Tyto popisky se liší například u rozvrhu učitele a třídy, protože u učitele není potřebná informace o vyučujícím učiteli a u rozvrhu třídy zase není potřeba uvádět zkratku třídy. V tomto případě jsme se však shodli, že to může být dáno i tím, že na jednotlivé zkratky v návrhu nejsou uživatelé zvyklí. V případě, že by místo nich byly zkratky jejich učitelů a jejich tříd, se kterými přichází do kontaktu denně ve škole, tak by tento problém pravděpodobně neexistoval.

Na základě tohoto testování jsem se tedy rozhodl zakomponovat do návrhu rozdělené hodiny učitelů a promyslet spojování hodin ve výsledné aplikaci. Dále pak navrhnout jiné rozložení jednotlivých oblastí pro zobrazené informace uživatelům. Protože těchto úprav není mnoho, rozhodl jsem se již dále netestovat návrh uživatelského rozhraní a přesunout se k implementaci ukázky aplikace.

# Kapitola 6

# Implementace aplikace pro tvorbu náhradních rozvrhů

V této kapitole jsou popsány jednotlivé kroky při implementaci ukázky aplikace. Nejprve se zaměřím na datovou základnu, nad kterou poběží celá aplikace. Poté zde popíši rozložení celé aplikace a po jednotlivých částech projdu funkčnost aplikace. Velká část bude věnována vyhledávání kružnic pro výměnu hodin v rozvrhu. Poslední kapitolu pak bude tvořit zhodnocení dosažených výsledků.

## 6.1 Databáze

Z testování uživatelského rozhraní v kapitole 5.4 vyplynula potřeba reálných dat při dalším testování uživatelského rozhraní. Z toho důvodu jsem se rozhodl využít databázi, kterou využívá systém SAS. Při uživatelském testování ve školách pak budou mít pracovníci škol k dispozici software pro řešení reálných situací. Dobře známé údaje a zkratky by jim měly pomoct s orientací v aplikaci.

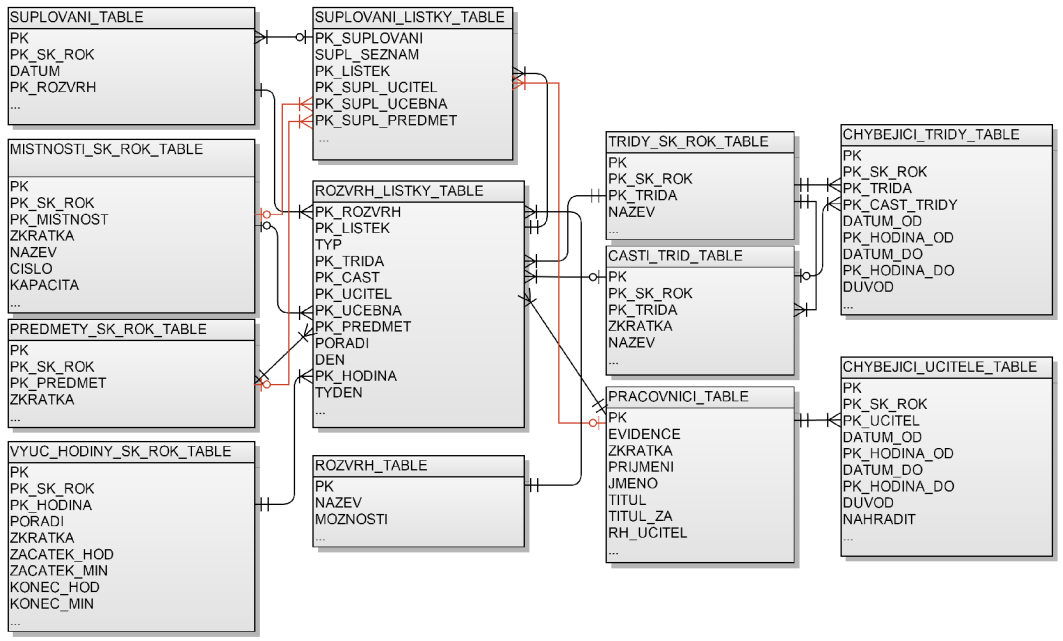
V následujícím textu budu často zmiňovat pojem lístek rozvrhu, a proto se ho zde pokusím nejprve vysvětlit. Lístek je základní jednotka v rozvrhu. V rámci jedné hodiny může být více lístků pro jednu třídu nebo jednoho učitele. Učitel tedy může vyučovat více tříd nebo jejich částí současně (například při seminářích). Obdobně může být třída rozdělena do více částí. Každá část může být v jiné učebně s jiným učitelem a studovat jiný předmět (například při výuce cizích jazyků).

Databáze systému SAS obsahuje mnoho tabulek, většinu z nich ve své práci nevyužívám a proto jim zde nebude věnován prostor. Schéma využívané části databáze si lze prohlédnout na obrázku 6.1. Uprostřed obrázku se nachází tabulka rozvrhu, ve které jsou uloženy všechny lístky rozvrhu na dané škole. Atributy DEN, TYDEN, PK_HODINA udávají umístění lístku v rozvrhu. Ostatní atributy pak poskytují podrobnější informace o lístku. Tyto informace odkazují na okolní tabulky ve schématu.

Informace o jednotlivých rozvrzích je uložena v tabulce ROZVRH_TABLE. V této tabulce se také nachází sloupec MOZNOSTI, který uchovává nastavení ohledně každého rozvrhu. Zejména zda se jedná o týdenní cyklus nebo dvoutýdenní, dny v týdnu kterými začíná a končí vyučovací týden nebo popis u speciálních hodin jako oběd, suplovací pohotovost a podobně.

Tabulka SUPLOVANI_LISTKY_TABLE pak slouží pro uložení různých variant suplování hodin, které učitelé nemohou odučit. Vztahy znázorněné červenou spojnicí předsta-

vují změny oproti běžnému rozvrhu, kdy může být vyplněna jedna nebo více z položek učitel (PK_SUPL_UCITEL), učebna (PK_SUPL_UCEBNA) nebo vyučovaný předmět (PK_SUPL_PREDMET). Nelze tedy změnit třídu, která je z pohledu suplování dána jako pevná. Pro přidání nebo rušení hodiny se používá sloupec SUPL_SEZNAM, kde hodnota 1 znamená zrušenou hodinu, 2 pro přidanou hodinu (v případě pouhé změny obsahuje sloupec hodnotu 0). Tabulka SUPLOVANI_TABLE slouží pro přiřazení změn v rozvrhu ke konkrétnímu datu a zároveň k rozvrhu, ke kterému se změny vztahují. Díky oběma tabulkám pro suplování je možné kontrolovat, zda nevzniklo více variant nahrazení jednoho lístku v jednom dni.




Obrázek 6.1: Schéma využité části databáze

Informace ohledně chybějících učitelů jsou uloženy v CHYBEJICI_UCITELE_TABLE. Chybějící třídy uloženy v tabulce CHYBEJICI_TRIDY_TABLE. Tyto dvě tabulky jsou téměř totožné. Rozdílné jsou pouze ve sloupcích, které identifikují chybějící entitu (dvojice PK_TRIDA a PK_CAST_TRIDY oproti PK_UCITEL). Do tabulky chybějících učitelů jsem navíc přidal sloupec NAHRADIT pro rozlišení absencí učitelů ze zákonného důvodu a z důvodu dohledu nad žáky. Tedy kdy učitel čerpá dovolenou nebo zákonné volno a tudíž není možné tyto hodiny počítat jako neodučená přímá pedagogická činnost. V opačném případě učitel vykonával pouze činnost související s přímou pedagogickou činností a je možné tyto hodiny nahrazovat při suplování. Dalšími sloupci jsou DATUM_OD a DATUM_DO sloužící pro datum začátku a konce absence. Obdobně sloupce PK_HODINA_OD a PK_HODINA_DO slouží pro zadání hodiny začátku a konce absence. Oba sloupce slouží jako cizí klíče z tabulky VYUC_HODINY_SK_ROK_TABLE. Na obrázku 6.1 není tato vazba naznačena z důvodu lepší přehlednosti schématu.

Tabulky, které ve svém názvu obsahují řetězec „SK_ROK“ jsou vázány i na konkrétní školní rok. A to z toho důvodu, že jednotlivé atributy položek v těchto tabulkách se mohou měnit při přechodu na jiný školní rok, avšak relace mezi položkami by měly zůstat neměnné. Popřípadě při pohledu na jiný školní rok by atributy platné v daném roce měly zůstat

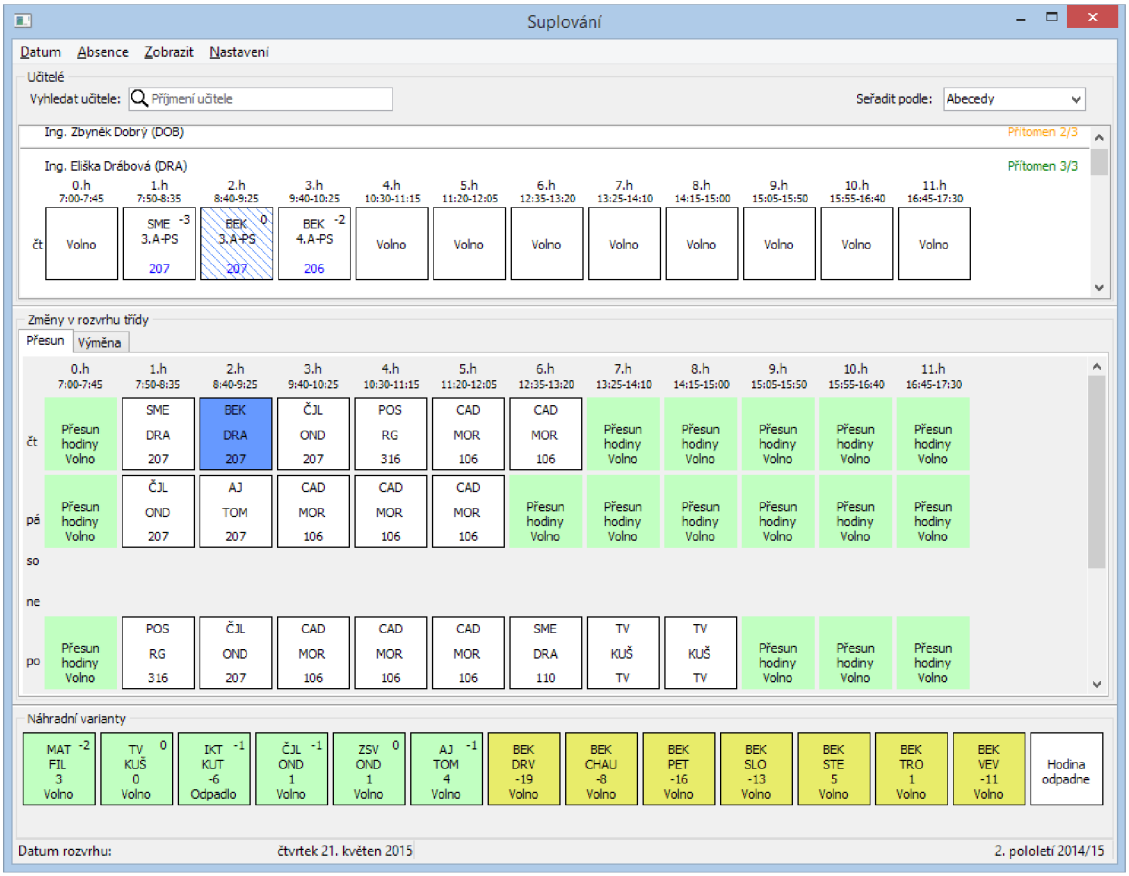
zachovány.

Kromě popsaných tabulek jsem využil ještě některé další. Jedná se především o tabulky nastavení. Tabulka MOZNOSTI_TABLE slouží k uchování aktuálního školního roku a pololetí. Z tabulky SKOLNI_ROK_TABLE lze zjistit doplňující informace o školním roku. Platnosti jednotlivých rozvrhů jsou uloženy v tabulce PLATNOST_ROZVRHU_TABLE. Zde je u každého rozvrhu uveden začátek platnosti a jako konec platnosti se uvažuje datum u následujícího rozvrhu.

## 6.2 Jednotlivé části aplikace

V této části popíši nejprve celkové rozložení aplikace a poté se budu věnovat jednotlivým částem aplikace. U každé části shrnu důležité body, které program vykonává a význam jednotlivých hodnot.

Uspořádání hlavního okna aplikace se oproti návrhu mírně změnilo. Nové rozložení lze vidět na obrázku 6.2. Oproti návrhu se zde změnilo zejména umístění seznamu rozvrhu učitelů. Tuto změnu jsem provedl z ohledem na větší počet vyučovacích hodin než jsem předpokládal v prvním návrhu.




Obrázek 6.2: Náhled na rozložení celé aplikace

Na obrázku 6.2 je vidět, že aplikaci nedělá problém zobrazit 12 vyučovacích hodin. V současné podobě je však méně místa pro zobrazení jmen učitelů. Při běžném používání by pracovníkům škol mělo stačit zobrazit pouze jednoho rozbaleného učitele s rozvrhem.

Popřípadě 5 učitelů v seznamu bez rozvrhu. Při své práci však uživatel potřebuje vidět pouze aktuálního učitele a zvolenou hodinu. Z toho důvodu si myslím, že zmenšení počtu viditelných učitelů nebude vadit. Navíc pokud uživatel hledá konkrétního učitele, může využít funkci pro vyhledání učitele. Učitele je taktéž možné seřadit podle počtu nevyřešených hodin. Tzn. podle hodin, které musí učitel ještě vyřešit pro aktuální den.

Pokud by toto rozdělení pro nějakého uživatele představovalo problém, je možné změnit poměr rozdělení jednotlivých sekcí pomocí *splitterů*, které jsou umístěny vždy mezi sousedními oblastmi. Toto nastavení se zachová i při dalším spuštěním aplikace, kdy se kromě rozložení načte také ostatní nastavení, jako pozice oken, údaje pro připojení k databázi a podobně.

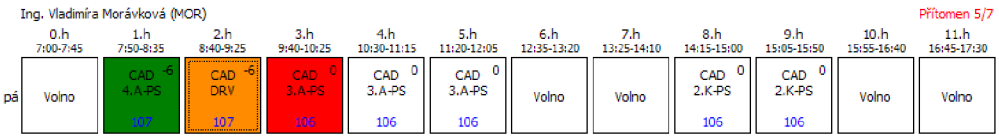
V menu aplikace zůstala nabídka téměř totožná s původním návrhem. Došlo pouze k přejmenování položek, aby více odpovídaly zvyklostem ve školním prostředí. V menu jsou tedy položky pro změnu pracovního dne, absenci tříd nebo učitelů, položka pro zobrazení změn v rozvrhu a nabídka nastavení pro zadání informací pro připojení k databázi nebo nastavení vlastnosti aplikace. Nabídka *Absence* a *Zobrazit* není implementována z důvodu dostupnosti těchto nástrojů z původního systému SAS.

Ve spodní části okna je pak stavová lišta, na které se zobrazuje aktuálně zvolené pracovní datum a taktéž aktuální školní rok a pololetí nastavené v systému SAS.

### Seznam učitelů

Jak již bylo zmíněno, první oblastí aplikace je sekce pro zobrazení seznamu učitelů. Seznam učitelů se načítá po každé změně pracovního data. Učitelé jsou načítáni jednotlivě a při načtení rozvrhu každého učitele se načte i suplování učitele pro daný den. Kromě vyhledání hodin učitele se spočítá taktéž plus/mínus statistika na základě úvazku učitele a suplovaných hodin.

Na obrázku 6.3 je zobrazen prvek, který obsahuje informace o učiteli. V horní části prvku je nejprve zobrazeno jméno zvoleného učitele. Pod ním se nachází zkratky vyučovacích hodin (dle pojmenování v databázi). Pod zkratkou je menším písmen umístěn začátek a konec hodiny. I když bude uživatel pravděpodobně čas začátku a konce vyučovacích hodin dobře znát, pomůže mu tato informace při orientaci v rozvrhu.




Obrázek 6.3: Pohled na rozvrh učitele

Pod informací částí s rozpisem hodin jsou zobrazeny jednotlivé lístky učitele. Na každém lístku se nachází zkratka předmětu, třídy, části třídy (pokud je vyplněna) a učebny. V pravém horním rohu prvku uživatel nalezne informace o počtu hodin učitele ve zvolený den a počtu hodin, ve kterých je učitel přítomen.

Kliknutím na hodinu z rozvrhu učitele je tato hodina vybrána a následuje výběr možností jejího nahrazení. O nabídce těchto možností budu psát v následujících částech kapitoly spolu s jejich zobrazením. Po zvolení náhrady se vybraná hodina učitele změní na oranžovou a zkratku třídy vyměním za zkratku nahrazujícího učitele. Kromě oranžové může být hodina zbarvena zeleně, červeně nebo bíle. Červená signalizuje, že učitel danou hodinu chybí a je nutné tuto hodinu nahradit. Zelená pak, že třída, kterou má učitel vyučovat, není přítomná.

Běžná hodina je označena bílou barvou. Detailní pohled na rozvrh učitele je znázorněn na obrázku 6.3.

Výpočet statistiky je v této části aplikace stěžejní. Nejprve je vypočten úvazek hodin z rozvrhu učitele, kde sečtu počet jedinečných trojic hodin, dnů a týdnů. Protože jsou lístky v rozvrhu uloženy buďto pouze pro jeden týden nebo pro dva týdny (v případě dvoutýdenního cyklu), vydělím základní úvazek počtem týdnů rozvrhu. Od základního úvazku učitele dále odečtu předepsaný úvazek pro všechny učitele na škole. Tatko získané číslo porovnám s absencí učitele v daném měsíci a to pouze s absencí takovou, která se dle zákona nepovažuje jako výkon práce. Nejprve jsou vybrány jednotlivé úseky absence v rámci týdne a poté jsou rozděleny na dny. Pro každý den zjistím z rozvrhu, kolik hodin měl učitel v daný den a v určeném rozsahu hodin absence odučit. Obdobně jsou vyřešeny i absence tříd, kdy zjistím, kolik hodin měl třídu učit který učitel. K tomuto číslu přičtu počet navíc odučených hodin (suplování za jiné učitele, popřípadě přesunuté hodiny z jiného dne). Veškeré tyto údaje jsou počítány vzhledem k aktuálnímu týdnu dle zvoleného pracovního data.

Uživatel tedy nemá přehled o přespočetných hodinách za celý měsíc. Nicméně výpočet těchto hodin by byl značně komplikovanější. Musely by se brát v potaz pouze přespočetné hodiny za každý týden, protože neodučené hodiny nelze nahrazovat v jiném týdnu. Obdobně by se řešil i úvazek učitele, který se sice může lišit v jednotlivých týdnech určeného rozvrhu, nicméně základní úvazek se počítá jako průměr těchto různých úvazků. Při počítání přespočetných hodin se ale opět uvažuje nadúvazek v rámci jednoho týdne.

Navíc by bylo nutné v rozvrhu vyznačit stálé přespočetné hodiny. Tyto hodiny by se poté braly jako přespočetné pokaždé, když by byly odučeny bez ohledu na to, zda v daném týdnu učitel splnil či nesplnil základní úvazek. Pokud by tyto hodiny nebyly zvláště vyznačeny, pak se jako přespočetná bere každá. Naopak by zase nebylo z této statistiky zřejmé, kolik hodin může být učiteli nařízeno jako náhrada za neodučené hodiny.

Mnou vypočítána statistika je tedy vhodnější pro učitele při rozhodování, koho z učitelů zvolit jako náhradu, avšak neukazuje, kolik daný učitel již odučil přespočetných hodin v rámci měsíce. S ohledem na zaměření mnou implementované aplikace jsem tedy zvolil týdenní statistiku. Druhá by se spíše hodila pro aplikaci zaměřenou na mzdové účetnictví.

### Rozvrh třídy

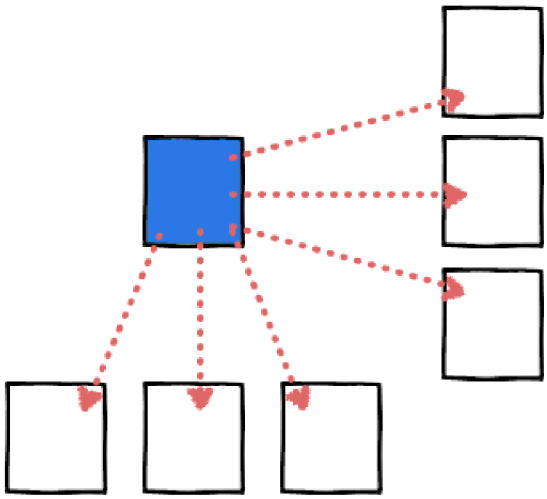
Tato část slouží k zobrazení rozvrhu třídy podle zvolené hodiny v rozvrhu učitele. V této části okna může uživatel zvolit přesun hodiny na volný termín. Také jsou zde zobrazeny varianty pro kruhovou výměnu hodin.

Hodiny tříd jsou zobrazeny po jednotlivých dnech. Dnů je zobrazeno 7 a začíná aktuálně zvoleným datem. Soboty a neděle jsou záměrně zobrazeny s menším rozsahem než ostatní dny, aby opticky oddělily jednotlivé týdny, ale na druhou stranu nerušily uživatele. V horní části jsou zobrazeny vyučovací hodiny obdobně jako u rozvrhu učitele.

Při zobrazení variant pro přesun hodin jsou zobrazeny volné místa, kam je možné hodinu přesunout. Tedy ta místa, kde má učitel i zvolená třída volno. U učitele jako volno beru, i pokud v databázi existuje lístek nějaké speciální hodiny (oběd, fixace, suplovací pohotovost, ...). V takovém případě je u volné hodiny zobrazena i informace o typu hodiny učitele. Uživatel má tak možnost zvolit v nouzových případech i hodiny, kdy má učitel například nastavenou fixaci.

Po zvolení přesunu hodin vykreslím šipku směřující z aktuální hodiny do místa přesunu. Začátek a konec šipky určím na základě relativní pozice obou míst. Pokud se místo začátku šipky nachází na vyšším řádku než místo konečné, pak se začátek šipky přesune k dolnímu

okraji místa a konec šipky k hornímu okraji. Pokud se oba řádky rovnají, pak jsou oba konce šipky přesunuty do středu místa. V případě, že počáteční místo bude ležet pod koncovým, pak bude začátek přesunut na horní okraj a konec na dolní okraj místa, do kterého směřuje. Obdobně postupuji v horizontálním směru. Tento postup je znázorněn na obrázku 6.4. Šipku tedy vykresluji tak, aby co nejméně překrývala aktuální hodnotu nebo místo přesunu.



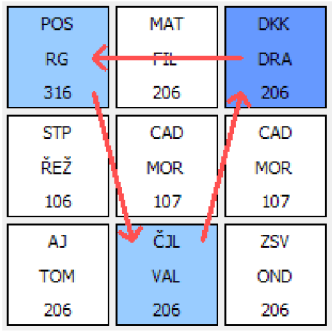
Obrázek 6.4: Konce šipky na základě relativní pozice míst

Tato optimalizace nejvíce pomáhá při kruhových výměnách, kde je přesouván vyšší počet hodin. Funkce kruhové výměny je po přesunu hodin druhá, která je zobrazena za pomoci rozvrhu tříd. Uživatel tuto funkci zobrazí přepnutím záložky u rozvrhu třídy.

Hlavní část výměny hodin je nalezení kruhů v rozvrhu hodin. Rozvrh třídy lze uvažovat jako graf, kde uzly tvoří jednotlivé hodiny a hrana z jednoho uzlu vede do druhého, pokud v databázi nemá učitel z první hodiny žádnou výuku v druhé hodině (v ROZVRH_LISTKY_TABLE platí typ<>0) a naopak třída v druhé hodině výuku má (tedy lze nalézt v tabulce ROZVRH_LISTKY_TABLE záznam, kde typ=0). Průchod grafu je pak proveden rekurzivní funkcí s omezeným zanořením tak, aby maximálně byly 4 hodiny pro výměnu. V kružnici jsou tak přesouváni pouze učitelé a předměty, zatímco třída a učebna zůstává neměnná. Zobrazení této kružnice lze pak provést pomocí dvou variant. Mezi variantami zobrazení kružnic lze přepínat v nastavení aplikace. První z nich je nazvána „Automatické generování variant výměn hodin“ druhou pak „Ruční generování variant výměn hodin“.

**Automatické generování variant výměn hodin** V tomto případě program podle dříve popsaného postupu, nalezne veškeré kružnice v grafu. Každá tato kružnice je pak zobrazena na vlastní kartě rozvrhu hodin. Kružnice je v programu zobrazena stejně jako na obrázku 6.5. Hlavní slabinou této varianty je velké množství kružnic. V některých případech jich v rozvrhu lze nalézt až 400. Proto jsem před zobrazením kružnice seřadil podle jejich délky. Uživatel tak první vidí ty jednodušší, které přesouvají menší počet hodin. Snížení počtu kružnic by se dalo dosáhnout implementací dalších omezení při hledání kružnic (preferované nasazení hodin, přesun i s učebnou, . . . ).

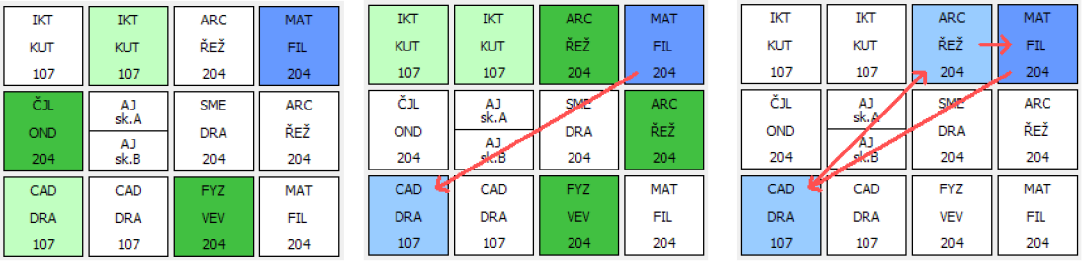
**Ruční generování variant výměn hodin** Tento postup těží z principu hledání kružnice, kdy na první úrovni je v kružnici pouze jeden uzel. V druhém kroku vznikne několik kružnic




Obrázek 6.5: Kružnice generovaná automaticky

vygenerovaných z prvního uzlu. Ve třetím kroku pak vznikne mnoho kružnic rozgenerováním každého uzlu z druhé úrovně. Počet kružnic tak roste exponenciálně. Rozhodl jsem se proto uživateli nabízet kružnice po částech a on by svým výběrem postupně konstruoval výslednou podobu kruhové výměny.

Algoritmus tedy postupuje obdobně jako v první variantě – naleznou se všechny kružnice. Kružnice jsou pak postupně procházeny a podle aktuální úrovně jsou uloženy indexy hodin (z indexu hodiny lze spočítat řádek a sloupec v rozvrhu), které lze zvolit pro pokračování kružnice. Je tedy nutné kromě aktuální úrovně znát i zvolenou hodinu z úrovně předchozí. Jelikož první hodinu kružnice zvolí uživatel už při výběru hodiny pro nahrazení, pokračuje tento algoritmu až od pomyslné druhé úrovně. Projde tedy všechny kružnice, a pokud se na předchozí úrovni kružnice nachází index hodiny zvolené uživatelem (u druhé úrovně tomuto požadavku vyhovují všechny kružnice), přidá index na aktuální úrovni kružnice do pole (pouze pokud nebyl index přidán dříve z jiné kružnice). Následně je pole těchto hodnot zvýrazněno v rozvrhu tříd.




(a) Výchozí stav (b) První krok (c) Druhý krok

Obrázek 6.6: Kružnice generovaná manuálně

Na obrázku 6.6 je naznačen postup uživatele při manuální volbě kružnice. Na obrázku 6.6a je zobrazen počáteční stav. V tomto stavu může uživatel zvolit jednu ze 4 variant, kam se modře označená hodina přesune. Světle zelená hodina nelze přesunout do původní hodiny a nelze tedy přímo uzavřít kružnici. Tmavě zelené naopak lze přesunout na místo původní hodiny a po jejich výběru se kružnice uzavře. Obrázek 6.6b naznačuje, že uživatel zvolil hodinu předmětu CAD učitele DRA. Zvolená hodina je označena modře a šipka vede z první hodiny na místo nově zvolené hodiny. Poslední krok se nachází na obrázku 6.6c, kde uživatel zvolil tmavě zelenou variantu ARC (ŘEŽ). Tím došlo k uzavření kružnice. Pokud se uživatel splete a označí špatnou hodinu, může se vrátit o krok zpět zvolením modré hodiny na požadované úrovni.

### Náhradní varianty

Poslední částí aplikace je oblast pro volbu náhradní varianty. Zde uživatel potvrdí změnu vybrané hodiny dle nabídnutých variant. Kromě volných učitelů se zde nachází ještě prvky pro potvrzení výměny či přesunu, prvek pro volbu odpadnutí hodiny, popřípadě prvek pro zrušení dříve nastaveného suplování.

Nejvíce položek tvoří volní učitelé a možné zástupy v hodinách. Tyto náhradní varianty se tvoří tak, že nejprve vytvořím seznam volných učitelů. U každého učitele tedy ověřím tři podmínky, které by ho mohly z množiny volných učitelů vyřadit:

- **Učí jinou třídu** – v aktuálním rozvrhu existuje lístek výuky (typ=0) v době zvolené hodiny. Pokud ano, tak ještě ověřím, zda je daná třída přítomná (není odpovídající záznam v tabulce CHYBEJICI_TRIDY_TABLE).
- **Supluje jinde** – v tabulce SUPLOVANI_LISTKY_TABLE je nastaveno na tuto hodinu jiné suplování, v této části je zároveň ověřeno, zda učiteli odpadla výuka na základě suplování.
- **Učitel chybí** – pro danou hodinu existuje záznam nepřítomnosti učitele v tabulce CHYBEJICI_UCITELE_TABLE.

Poté v seznamu volných učitelů zjistím zda, v daném roce vyučují předmět, který se v dané třídě má dle rozvrhu vyučovat. Podmínka „zda učí“ vyšla z požadavku uživatelů, kdy je pro ně spíše důležité, zda v daném roce učitel vyučuje předmět, než aprobace někdy v minulosti. Toto zjistím z tabulky ROZVRHY_LISTKY_TABLE (pokud se ní nachází záznam pro daného učitele s daným předmětem). Stejným způsobem zjistím, zda učitel vyučuje danou třídu.

Pomocí těchto dvou hodnot rozdělím učitele do čtyř kategorií. Učí třídu i předmět, učí jen předmět, učí jen třídu a ostatní učitelé. Zobrazení jednotlivých kategorií pak probíhá podle předpokládané kvality výsledného suplování. Nejprve jsou tedy zobrazeni učitelé, kteří učí třídu a předmět, poté učitelé kteří učí jen předmět, atd. Učitel je zobrazen pouze v nejvyšší kategorii, ve které se nachází (například učitel, který učí předmět i třídu, nebude zobrazen s učiteli, kteří učí jen předmět). Zvláštní skupinou jsou učitelé vyučující danou třídu, protože ti jsou zobrazení vícekrát pokaždé s jiným předmětem, který ve třídě vyučují. U každého předmětu jsou zobrazeny informace o rušení hodin předmětu v minulosti (plus/mínus statistika) U ostatních kategorií se předmět nemění a tak tato statistika není nutná. Pro lepší rozpoznání kategorií je každá podbarvena jinak. Na obrázku 6.7 jsou postupně zobrazeny všechny tyto čtyři kategorie plus speciální varianty popsané dříve.


Obrázek 6.7: Náhradní varianty pro změnu rozvrhu

U variant které vznikly ze seznamu volných učitelů jsou informace o předmětu, učiteli, úvazku učitele a typu volna v dané hodině. Jsou zde zobrazeny nastavené zkratky zvláštních hodin v rozvrhu. Pokud by zvolenou variantou došlo ke změně předmětu, je v pravém rohu předmětu uvedena i statistika předmětu.

Zvolením varianty dojde k jejímu uložení do databáze a změně hodiny v rozvrhu učitele.

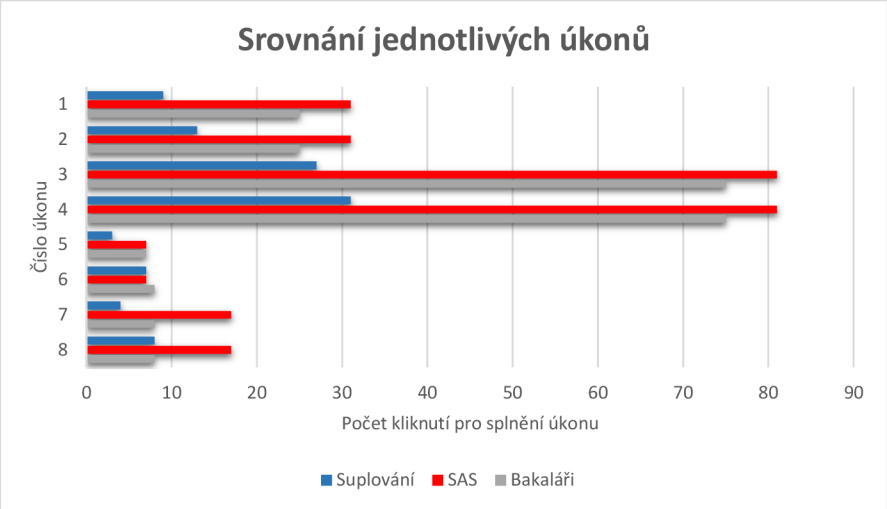
## 6.3 Srovnání s existujícím softwarem

Ke srovnání jsem využil základní úkony uživatele při tvorběnáhradního rozvrhu. Pro srovnání jsem vybral 2 nejčastější systémy pro školní agendu, a to aplikace SAS a Bakaláři. V porovnání například není zahrnuta kruhová výměna hodin, která v mé aplikaci zabrala největší část práce, avšak v ostatních systémech je kruhová výměna nemožná nebo nesmírně náročná a většinu podmínek musí hlídat sám uživatel.

| | | | Počet kliků pro splnění úkonu | | |
|---|---|---|---|---|---|
| # | Úkon | Platnost rozvrhu | Bakaláři | SAS | Suplování |
| 1 | Zástup 1 učitele | dnes | 25 | 31 | 9 |
| 2 | | za 2 dny | 25 | 31 | 13 |
| 3 | Zástup 3 učitelů | dnes | 75 | 81 | 27 |
| 4 | | za 2 dny | 75 | 81 | 31 |
| 5 | Změna nastaveného suplování | dnes | 7 | 7 | 3 |
| 6 | | za 2 dny | 8 | 7 | 7 |
| 7 | Přesun hodiny | dnes | 8 | 17 | 4 |
| 8 | | za 2 dny | 8 | 17 | 8 |

Tabulka 6.1: Srovnání jednotlivých aplikací

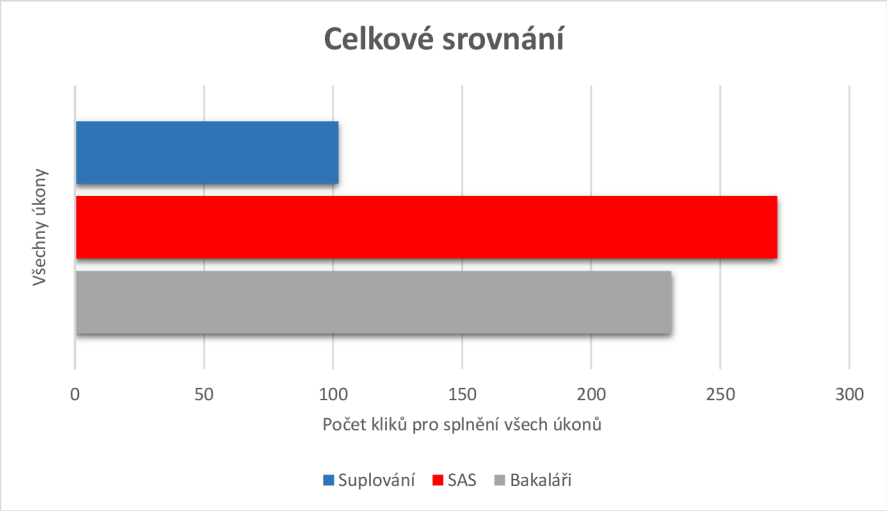
Jako testovací soubor úkonů jsem vybral tvorbu náhradního rozvrhu pro zástup jednoho nebo tří učitelů. Dále jsem u všech úkonů rozlišil dobu začátku platnosti náhradního rozvrhu. Nejčastěji učitelé plánují rozvrh dva dny dopředu. Druhou variantou je řešení krizové situace na začátku nebo v průběhu dne platnosti náhradního rozvrhu. Často těsně před hodinou. Dalším úkonem je změna již nastaveného suplování a přesun hodiny.




Obrázek 6.8: Srovnání jednotlivých úkonů

U úkonů 1 až 4 jsem pro porovnání uvažoval změnu 4 vyučovacích hodin. Právě čtyři hodiny jsou průměrný denní uvázek učitele při základní pedagogickém úvazku 21 hodin

přímé pedagogické činnosti týdně. Pro úkony „změna suplování“ a „přesunu hodiny“ (úkony 5 až 8) jsem počítal pouze s jednou hodinou, protože výkon těchto úkonů pro více hodin není běžný. Dále jsem neuvažoval četnost jednotlivých úkonů v souboru práce uživatele. Sice by se dalo předpokládat, že nejčastěji bude uživatel tvořit náhradní rozvrh dopředu. Respektive nejvíce úkonů vykoná právě s předstihem (úkony 2, 4, 6, 8). I když bude akutní plánování rozvrhu (úkony 1, 3, 5, 7) méně časté, uživatel bude pod časovým tlakem a bude vyžadovat rychlé vyřešení vzniklé situace. V aplikaci Suplování tedy kladu důraz na rychlost u akutních problémů spolu s velkou přehledností u úkonů plánovaných s předstihem.




Obrázek 6.9: Srovnání všech úkonů

Jednotlivé úkony a počet kliknutí, které musí uživatel vykonat, jsou znázorněny v tabulce 6.1. Hodnoty jsou zaneseny do grafu 6.8. Souhrnný přehled kliků je zobrazen na grafu 6.9. Zatímco u programů SAS a Bakaláři musí uživatel vykonat přibližně stejný počet kliků (rozdíl 41 kliků, cca 15 % ve prospěch systému Bakaláři), mnou navržená aplikace Suplování zjednodušší práci uživatele o 63 % (o 170 kliků) oproti aplikaci SAS a o 56 % (o 129 kliků) oproti aplikaci Bakaláři. Moje aplikace navíc zobrazuje informace, které uživatel využívá při rozhodování o zastupujícím učiteli (přespočetné hodiny, znalost prostředí či předmětu, speciální hodiny, možné změny předmětu, ...).

## 6.4 Možné pokračování práce

V této části kapitoly jsou popsány úpravy aplikace, které je potřeba ještě provést nebo implementovat pro nasazení softwaru ve školách. Některé z těchto činností jsou spíše teoretické a nemá praktický význam je implementovat, jiné jsou téměř nutné pro provoz aplikace u budoucích uživatelů.

V průběhu tvorby aplikace se ukázalo že výpočet přespočetných hodin je daleko náročnější, než jsem původně předpokládal. V této oblasti by se dalo vymyslet mnoho řešení, ať už pro zobrazení statistiky či rozdělení výpočtu do různých kategorií (souhrý měsíční počet přespočetných hodin, aktuální týdenní, ...). Takové řešení by však bylo pravděpodobně

stejně obtížné jako samotná kruhová výměna hodin a význam by mělo spíše v mzdovém softwaru.

Během prázdnin bych také chtěl začlenit software do školního prostředí a vyzkoušet aplikaci v testovacím provozu. Na takový zásah do běhu školy nebyl ve školním roce prostor. I když se při testování uživatelského rozhraní zdála práce s aplikací intuitivní a snadno pochopitelná, provedl jsem některé změny při implementaci aplikace, které je nutné znovu ověřit. Jedná se zejména o manuální generování kruhové výměny.

Co se týče kruhové výměny, tak je potřeba zjistit, která varianta bude uživatelům více vyhovovat (manuální nebo automatická). U automatického generování kruhů by stálo za zvážení, zda nezohlednit při výměně i přesun učebny. Zde je ale problém, že například u tělocviku či specializovaných předmětů je nutná určitá učebna, ale naopak pro český jazyk tato podmínka není tak podstatná. Na většině škol taktéž není přesně určeno, který předmět nutně musí být v dané učebně, u kterého je tato učebna pouze doporučená nebo který předmět může být vyučován kdekoliv. Například u tělocviku se často vyučuje i mimo učebnu (například na hřišti nebo běžecké dráze). Takže ve výsledku by tato podmínka, i přes nastavení, byla závislá na rozhodnutí uživatele.

V aplikaci není implementováno spojení a kontrolování výměn různých částí tříd. A to z důvodu neexistujícího rozdělení konkrétních žáků do částí tříd. U některých je takové rozdělení jasné (tělesná výchova chlapci-dívky), ale u některých se žáci mohou nacházet ve více zdánlivě disjunktních částech (cizí jazyky, semináře, ...). V první řadě by se tedy musel uvést vzájemný vztah daných částí tříd, v lepším případě pak zavést množinu žáků pro každou skupinu a následně tyto skupiny kontrolovat při výměnách lístků hodin, popřípadě při spojování. V pokročilejší fázi pak i rozdělení části třídy na skupiny a jejich připojení k jiným částem.

# Kapitola 7

# Závěr

Cílem práce bylo navrhnout a implementovat software pro tvorbu náhradních rozvrhů. Vytvořená aplikace svými funkcemi odpovídá požadavkům budoucích uživatelů. Výhoda oproti konkurenci spočívá v jednoduchosti uživatelského rozhraní a aplikace taktéž obsahuje funkce, které v konkurenčních systémech nejsou dostupné.

Snažil jsem se využít získaných znalostí ze školního prostředí a odstranit nedostatky dostupných aplikací s podobným zaměřením. Zvláštní důraz byl kladen na požadavky uživatelů, tedy pracovníků škol, kteří náhradní rozvrhy do současnosti tvořili ručně. Dle srovnání uvedeného v předposlední kapitole se podařilo zjednodušit práci s aplikací o 63 % oproti konkurenčním systémům.

Práci bych rozdělit do dvou fázi. První z nich tvoří návrh aplikace a testování tohoto návrhu s uživateli ve školách. Jako druhou fázi pak lze označit proces implementace této aplikace.

Při návrhu uživatelského rozhraní jsem nejprve získal informace o školském prostředí prostřednictví konzultací ve školách. Dále jsem se snažil lépe poznat, jak postupují pracovníci škol při tvorbě náhradního rozvrhu. Srovnal jsem existující aplikace pro tvorbu náhradních rozvrhů. Kromě těchto informací jsem při návrhu využil i zákon o zastupování pedagogických pracovníků. Poté jsem nastudoval moderní trendy v návrzích uživatelských rozhraních. Druhou oblastí, kterou jsem pro návrh aplikace prostudoval, je teorie grafů. Na základě těchto poznatků jsem navrhl uživatelské rozhraní aplikace a algoritmus, který vyhledá kruhové výměny hodin v rozvrhu třídy. Nakonec jsem implementoval výslednou aplikaci.

Během procesu tvorby aplikace se vyskytlo několik problémů, které je možné řešit jako další pokračování práce. Jedním z nich je například výpočet přespočetných hodin pedagogického pracovníka za kalendářní měsíc. Celou aplikaci bych chtěl během prázdnin podrobit testovacímu provozu, na který není během školního roku prostor.

# Literatura


[1] HOŘAVA, Michal a kol.: *Uživatelsky přívětivá rozhraní*. Horava and Associates, Praha, 2009, ISBN 978-80-254-5295-0.

[2] BENYON, David, TURNER, Phil and TURNER, Susan: *Designing iteractive systems: people, activities, contexts, technologies*. Essex, první vydání, Addison-Wesley, 2005, ISBN 0-321-11629-1, 789 s.

[3] WOOD, E. Larry: *User interface design*. CRC Press, New York, 1998, ISBN 0-8493-3125-0.

[4] KRUG, Steve: *Don't make me think*. New Riders, Berkeley, 2006, ISBN 0-321-34475-8.

[5] ZELENÝ, Jaroslav a MANNOVÁ, Božena: *Historie výpočetní techniky*. Scientia, Praha, 2007, ISBN 80-86960-04-8.

[6] ANDERSON, P. Stephen: *Přitažlivý interaktivní design: jak vytvářet uživatelsky přívětivé produkty*. Brno, první vydání, Computer Press, 2012, ISBN 978-80-251-3722-2, 240 s.

[7] Mac OS X Human Interface Guidelines [online]. Apple Inc. 2011, [cit. 2015-03-16]. URL http://developer.apple.com/library/mac/#documentation/UserExperience/Conceptual/AppleHIGuidelines/Intro/Intro.html

[8] MATOUŠEK, Jiří, NEŠETŘIL, Jaroslav: *Kapitoly z diskrétní matematiky*. Karolinum, první vydání, Praha, 2002, ISBN 80-246-0084-6, 381 s.

[9] HART, P.; NILSSON, N.; RAPHAEL, B.: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, ročník 4, č. 2, 1968: s. 100–107, ISSN 0536-1567.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4082128

[10] KOLÁŘ, Josef: *Teoretická informatika*. Česká informatická společnost, druhé vydání, Praha, 2004, ISBN 80-900853-8-5, 205 s.

[11] VIRIUS, Miroslav: *Základy algoritmizace*. České vysoké učení technické, první vydání, Praha, 1995, ISBN 80-010-1346-4.

[12] MŠMT ČR: Aktuální znění zákona o pedagogických pracovnících k 1. lednu 2015. 2015, [online],[cit. 2015-01-10].
URL http://www.msmt.cz/dokumenty/aktualni-zneni-zakona-o-pedagogickych-pracovnicich-k-1-lednu

[13] MŠMT ČR: Právní výklad k § 23 zákona o pedagogických pracovnících. 2015, [online],[cit. 2015-03-20].
URL http://www.msmt.cz/dokumenty/pravni-vyklad-k-23-zakona-o-pedagogickych-pracovnicich

[14] KLEŇHOVÁ, Michaela a kol.: *Krajská ročenka školství 2010*. Ústav pro informace ve vzdělání, Praha, 2011, ISBN 978-80-211-0618-5.

[15] W3Schools: Browser Display Statistics. 2015, [online],[cit. 2015-01-10].
URL http://www.w3schools.com/browsers/browsers_display.asp

# Příloha A

# Obsah DVD

- Prototyp uživatelského rozhraní
- Binární soubory aplikace Suplování
- Zdrojové soubory
- Návod pro zprovoznění aplikace Suplování
- Instalační soubory a popis instalace SAS
- Vzorová databáze
- Program pro úpravu databáze
- Text technické zprávy
- LaTeXové zdrojové kódy technické zprávy