Časový souhrn audiometrických vyšetření

The time summary of the audiometric investigations

Bc. Tomáš Barot

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš BAROT**
Osobní číslo:	**A07378**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Časový souhrn audiometrických vyšetření**

Zásady pro vypracování:

1. **Seznamte se s druhy vyšetření v ORL lékařství, způsoby jejich provedení a archivace.**
2. **Zpracujte návrh systému pro vizualizaci časových souhrnů provedených vyšetření.**
3. **Realizujte softwarovou aplikaci zajišťující vhodnou implementaci navrženého systému.**
4. **Ověřte řešení na rozsáhlejší databázi vyšetření a zhodnoťte možnosti jeho praktického nasazení.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **LEJSKA, M., et al. Základy praktické audiologie a audiometrie. 1. vyd. Brno: IDVPZ, 1994. 171 s. ISBN 80-7013-178-0.**
2. **UCHYTIL, B., et al. Vyšetřovací metody a základní diagnostika v otorinolaryngologii. 1. vyd. Praha: Triton, 2002. 254 s. ISBN 80-7254-190-0.**
3. **BINGHAM, B. J. G., HAWKE, M., KWOK, P. Atlas of Clinical Otolaryngology. 1st ed. St. Louis: Mosby-Year Book; illustrated edition, 1991. 206 s. ISBN 978-1556643156.**
4. **SMART, J., HOCK, K., CSOMOR, S. Cross-Platform GUI Programming with wxWidgets. 1st ed. Indiana: Prentice Hall, 2005. 700 s. ISBN 0-13-147381-6.**
5. **BLIŽŇÁK, Michal. Systémové programování. 1. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.**
6. **HAROLD, E. R., MEANS, W. S. XML v kostce. 1. vyd. Praha: Computer Press, 2002. 439 s. ISBN 80-7226-712-4.**
7. **SKONNARD, A., GUDGIN, M. XML: pohotová referenční příručka: referenční příručka programátora ke XML, XPath, XSLT, XML Schema, SOAP a dalším. 1. vyd. Praha: Grada, 2006. 342 s. ISBN 80-247-0972-4.**
8. **SCHMULLER, Joseph. Myslíme v jazyku UML: knihovna programátora. 1. vyd. Praha: Grada Publishing, 2001. 359 s. ISBN 80-247-0029-8.**

Vedoucí diplomové práce: **Ing. Viliam Dolinay**
Ústav automatizace a řídicí techniky

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **8. června 2010**

Ve Zlíně dne 19. února 2010

L.S.

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

prof. Ing. Vladimír Vašek, CSc.
děkan

prof. Ing. Vladimír Vašek, CSc.
ředitel ústavu

ABSTRAKT

Hlavním cílem diplomové práce bylo vytvoření systému pro statistické vyhodnocení časových souhrnů počtu provedených vyšetření pro ordinace ORL lékařství. Pro výsledky statistického vyhodnocení dat, podle definovatelných filtračních podmínek, je zajištěna vhodná forma prezentace výsledků a možnost volby způsobů jejich interpretace. V teoretické části jsou vysvětleny druhy a způsoby vyšetření sluchu v lékařském odvětví otorhinolaryngologie. Je zde popsána metodika tvorby aplikace a analýza způsobu archivace dat vyšetření v ORL. Praktická část dokumentuje etapu definice požadavků, návrhu architektury systému, realizaci a testování správnosti řešení. Následuje přehled funkcí, které systém pro statistickou analýzu provozu ORL ordinace poskytuje. Závěrem je uvedeno zhodnocení možností praktického nasazení realizovaného systému, a to ve formě statistického systému doplňujícího již existující programové vybavení v ordinacích ORL.

Klíčová slova: Otorhinolaryngologie, ORL, filtrace dat, vyhodnocení dat, testování, XML

ABSTRACT

The main aim of the thesis was to create the system, that evaluates the time-count summaries of the realized investigations in the ORL medical clinics. For the statistical evaluations of the examination data, was designed the appropriate visual output form. The outputs can be filtered according to the requirement conditions of the doctors. In the teoretically part of the thesis are explained the types and methods of the hearing investigations in the medical branches Otorhinolaryngology. Teoretical part describes the software creating methodology and the analysis of the data way archiving in the ORL. The practical part contains several parts depending on the system requirements, which are the system architecture design, the realization and verification of the solution. In the practical part are explained the functions of the implemented system, that are used for the ORL data statistic analyses and their results visualization. In the final part is mentioned the possible practical utilization of the implemented system as the statistical analytic tool, that can complement the current software equipment in the ORL clinics.

Keywords: Otorhinolaryngology, ORL, data filtration, data evaluation, testing, XML

Děkuji vedoucímu diplomové práce panu Ing. Viliamu Dolinayovi za odborné vedení, za konzultace a cenné připomínky při řešení práce, za možnost ověření výsledků na reálných datech z praxe. Rád bych poděkoval své rodině, že mi poskytla dobré podmínky pro studium, podporovala mě v době tvorby diplomové práce a v průběhu celého studia.

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, 31.5.2010 ..…….

podpis diplomanta

OBSAH

ÚVOD **9**
I TEORETICKÁ ČÁST **10**
1 VYŠETŘENÍ V ORL LÉKAŘSTVÍ **11**
1.1 DRUHY VYŠETŘENÍ A ZPŮSOBY JEJICH PROVEDENÍ 11
1.2 POPIS SYSTÉMU FOWLER POUŽÍVANÉHO V ORL LÉKAŘSTVÍ, ZPŮSOB ARCHIVACE DAT SYSTÉMEM 14
2 POSTUP TVORBY SOFTWAROVÝCH APLIKACÍ **17**
2.1 VOLBA METODY VÝVOJE SOFTWAROVÉHO PRODUKTU 17
2.2 FÁZE SPECIFIKACE A ANALÝZY POŽADAVKŮ 18
2.3 FÁZE NÁVRHU A VÝVOJE VÝSLEDNÉHO ŘEŠENÍ 18
2.4 FÁZE TESTOVÁNÍ 19
2.5 TESTOVÁNÍ VÝPOČETNÍ SLOŽITOST ŘEŠENÍ 20
2.6 VÝSLEDNÝ SOFTWAROVÝ PRODUKT 20
3 PROGRAMOVACÍ PROSTŘEDKY A NÁSTROJE **21**
3.1 VOLBA TECHNOLOGIE WXWIDGETS PRO TVORBU OKENNÍCH APLIKACÍ 21
3.2 VOLBA PROGRAMOVACÍCH NÁSTROJŮ, JEJICH INSTALACE A KONFIGURACE 22
4 PRVKY A TECHNIKY VÝVOJOVÉ KNIHOVNY WXWIDGETS **26**
4.1 TECHNIKA TVORBY APLIKACE, ZPRACOVÁNÍ UDÁLOSTÍ 26
4.2 OVLÁDACÍ VIZUÁLNÍ PRVKY GRAFICKÉHO ROZHRANÍ PROGRAMU 27
4.3 NEVIZUÁLNÍ PRVKY KNIHOVNY 33
4.4 ROZŠÍŘENÍ KNIHOVNY WXWIDGETS 35
5 TECHNOLOGIE XML **37**
5.1 ÚČEL A PRINCIPY JAZYKA XML 37
5.2 ANALÝZA ZPŮSOBU ARCHIVACE DAT SYSTÉMEM FOWLER Z POHLEDU XML 39
6 POSTUPY A TEORIE POUŽITÉ PŘI PROGRAMOVÁNÍ **41**
6.1 PRINCIP MEALLYHO KONEČNÉHO AUTOMATU 41
6.2 POSTUP ŘAZENÍ DAT 43
II PRAKTICKÁ ČÁST **44**
7 ZADÁNÍ SYSTÉMU SE SPECIFIKACÍ POŽADAVKŮ **45**
8 NÁVRH SYSTÉMU PRO VYHODNOCENÍ A VIZUALIZACI ČASOVÝCH SOUHRNŮ PROVEDENÝCH VYŠETŘENÍ **46**
8.1 NÁVRH ARCHITEKTURY SYSTÉMU 47
8.2 PRINCIPY JEDNOTLIVÝCH ČÁSTÍ ARCHITEKTURY SYSTÉMU 49
8.3 NÁVRH ŘEŠENÍ NAČÍTÁNÍ XML DAT OBLASTÍ DATA XML HIERARCHY 49
8.4 NÁVRH ŘEŠENÍ OBLASTI STATISTICKÉ ANALÝZY DAT 51
8.5 NÁVRH ZPŮSOBU VIZUALIZACE STATISTICKÝCH VÝSLEDKŮ 52
8.6 UKLÁDÁNÍ DAT V RÁMCI JEDNOTLIVÝCH ČÁSTÍ ARCHITEKTURY SYSTÉMU 53
8.7 NÁVRH TESTOVACÍ APLIKACE PRO OVĚŘENÍ SPRÁVNOSTI SYSTÉMU 56
9 REALIZACE NAVRŽENÉHO SYSTÉMU IMPLEMENTACÍ SOFTWAROVÉ APLIKACE **57**

9.1 PROGRAMOVÁNÍ NAVRŽENÉHO SYSTÉMU 57
9.2 IMPLEMENTACE OBLASTÍ ARCHITEKTURY NÁVRHU SYSTÉMU 59
9.3 TVORBA PROGRAMU TESTOVACÍHO GENERÁTORU DAT 64
10 PŘEHLED FUNKCÍ VYTVOŘENÉHO SYSTÉMU 65
11 TESTOVÁNÍ SPRÁVNOSTI VÝSLEDKŮ SYSTÉMU, MĚŘENÍ ČASOVÉ SLOŽITOSTI VYBRANÝCH POSTUPŮ ŘEŠENÍ 74
11.1 TESTOVÁNÍ SPRÁVNOSTI VÝSLEDKŮ 74
11.2 MĚŘENÍ ČASOVÉ SLOŽITOSTI APLIKACE 75
12 OVĚŘENÍ ŘEŠENÍ NA ROZSÁHLEJŠÍ DATABÁZI VYŠETŘENÍ 79
13 MOŽNOSTI PRAKTICKÉHO NASAZENÍ PROGRAMU 83
ZÁVĚR 84
ZÁVĚR V ANGLIČTINĚ 85
SEZNAM POUŽITÉ LITERATURY 86
SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK 88
SEZNAM OBRÁZKŮ 89
SEZNAM TABULEK 91
SEZNAM PŘÍLOH 92

ÚVOD

Diplomová práce se zabývá řešením problematiky statistického vyhodnocení časových souhrnů počtu provedených vyšetření v rámci provozu ordinace ORL lékařství. V praxi může být v ordinacích používán pro uchování a diagnostické zpracování dat vyšetření již existující systém *Fowler*. Zadání této práce vyjadřuje požadavek ze strany lékařské odborné veřejnosti, statisticky vyhodnocovat informace o provozu ordinace, a to z pohledu provedených druhů vyšetření. Požadavkem na systém je především univerzální, ekonomická a pro uživatele dostatečně obecná varianta řešení aplikovatelná do praxe. Systém má mít minimální podmínky na programové a technické vybavení počítačových stanic, na kterých bude zprovozněn. Neměly by být pro implementaci a výsledný provoz použity komerčně závislé nástroje ani techniky. Systém by měl fungovat v klasickém nastavení operačního systému o minimální verzi MS Windows XP a nepodmiňovat svůj běh podmínkou připojení k síti Internet. Pro snadnou instalaci a přenositelnost instalace systému by bylo vhodné nezapisovat konfiguraci nastavení do systémových registrů operačního systému. Podle všech těchto podmínek se návrh a následná implementace řešení systému opírá. Pro uživatele systém musí poskytovat univerzální možnost nastavení v podobě filtrace dat, volby druhů statistických vyhodnocení a formátu prezentace výsledků jednotlivých analýz. Systém má pro další případné zpracování dat zahrnovat také podporu jejich exportu. Přesné postupy promítnutí návrhů architektury řešení stanovených požadavků do finálního řešení systému, jsou definovány fázemi vývoje softwarové aplikace, podle kterých jsem vytvořil osnovu praktické části práce. V teoretické části se zabývám studií teorií a postupů, které jsem podrobně prostudoval pro následné využití při tvorbě výsledku praktické části práce. Tvořený systém se opírá o terminologii problematiky ORL lékařství, což je stěžejní oblast, které se implementace vlastního systému týká. Pro samotnou tvorbu řešení je třeba zvolit vhodné programovací nástroje, techniky a postupy. Konkrétní návrh architektury systému a jeho realizaci s následným testováním zahrnuje část praktická. Důležitou činností před samotným uvolněním navrhovaného systému do praxe je také provedení důkladného otestování jeho funkcí a volba vhodného způsobu ověření pravdivosti výsledků. Jedná se o výsledky statistických analýz provozu ORL ordinace v textových sestavách, tabulkách a grafech. Program s popsanými funkcemi ve formě nápovědy bude možno použít jako obecně orientovaný nástroj pro statistickou analýzu počtu provedených vyšetření ordinace ORL s načtením dat existujícího systému *Fowler*.

I. TEORETICKÁ ČÁST

1 VYŠETŘENÍ V ORL LÉKAŘSTVÍ

Praktická část diplomové práce se opírá o názvosloví z oboru lékařství ORL (otorhinolaryngologie). Proto je vysvětlena kategorizace základních druhů vyšetření a způsobů jejich provedení. Jedná se o zaměření oboru lékařství ORL, a to na oblast vyšetření funkcí sluchového ústrojí. Těmito záležitostmi se odborně zabývá audiologie, která je oborem péče o sluchové ústrojí. Audiologie má za úkol analyzovat sluchový orgán z hlediska funkčnosti a umožnit dále lékaři stanovit diagnózu případné poruchy sluchového aparátu. Audiologie používá pro analýzu funkcí sluchového ústrojí řadu metod, kterými mohou být jak metody orientačního charakteru, tak přesné exaktní metody. [1]

1.1 Druhy vyšetření a způsoby jejich provedení

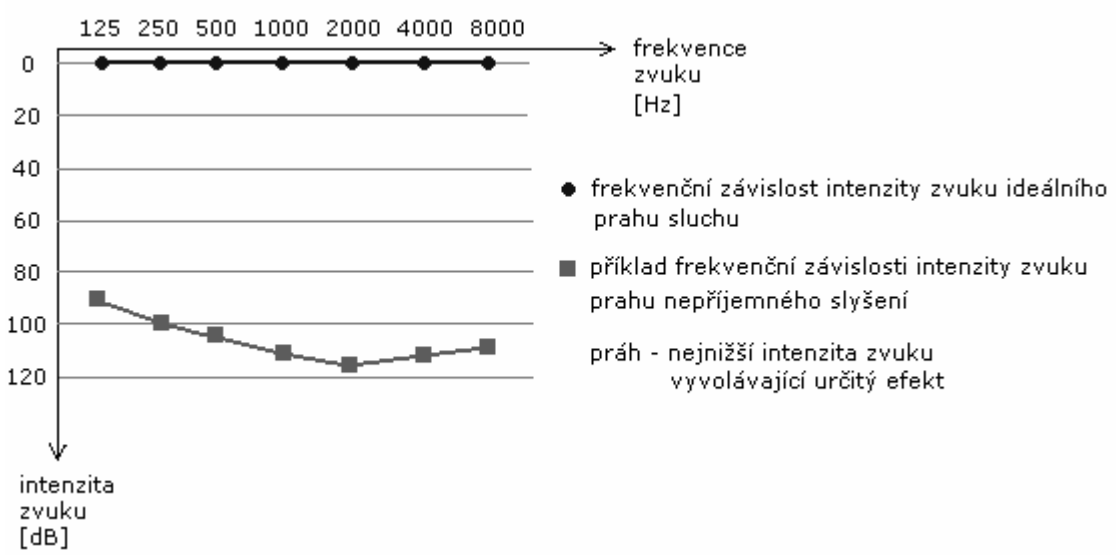
V audiologii je používán lékařský aparát ve formě nástrojů a přístrojů. Přístroje jsou založeny z pohledu fyzikálního principu na zákonech akustiky. Vstupy i výstupy přístrojů jsou konstruovány tak, aby podněty veličin akustické fyzikální podstaty dokázaly detekovat, a také generovat. Samotný zvuk je mechanickým vlněním nesoucí energii. Je charakterizován veličinami frekvence zvuku (jednotka *Hz*) a například také intenzita zvuku (jednotka $W.m^{-2}$, po vztažení k referenční hodnotě 10^{-12} $W.m^{-2}$ - relační jednotka *dB*). Přístroje navíc mohou počítat s akustickým tlakem, který souvisí s působením zvukové vlny na plochu překážky (jednotka *Pa*). Mezi významné přístroje patří audiometr a tympanometr. Audiometr je používán jako generátor akustických signálů o nastavené frekvenci a intenzitě. Dalšími součástmi a dílčími nástroji pro vyšetření jsou také příslušenství a součásti audiometru - sluchátka, kostní vibrátor, tlačítkový signalizátor, měnič intenzity signálu. Příkladem samostatného nástroje je ladička. Ladičky jsou základními zdroji akustického harmonického signálu. Pro měření charakteristik funkcí sluchu se používají dva způsoby, a to měření vzdušným vedením nebo kostním vedením. Na rozdíl od klasického přenosu zvuku vzduchem do oblasti vnitřního ucha, se kostní vedení vyznačuje tím, že je akustická energie přenášena pomocí rozechvění části lebeční kosti. Tato oblast instrumentace v audiologii v ORL lékařství je používána právě při provádění úkonů vyšetřování sluchového aparátu pacienta. Pro přesnější výsledky jednostranných vyšetření sluchového ústrojí se provádí šumové maskování protilehlého ucha. Aby bylo vyšetření provedeno s co nejmenšími vlivy vnějšího prostředí, vyšetřuje se pacient ve speciální audiometrické komoře. [1]

Hlavním cílem jednotlivých druhů metod je získání výsledků, které poskytnou podklady pro lékařské stanovení poruchy sluchu. Dále jsou uvedeny konkrétní druhy metod vyšetření a způsob jejich provedení v audiologii. Kromě dělení na orientační a exaktní se metody člení na objektivní a subjektivní. Získání výsledků u objektivní metody vyšetření spočívá pouze na práci lékařského přístroje. Výsledky objektivních vyšetření nemohou být nikterak ovlivněny vůlí pacienta. Subjektivní metody předpokládají účast vyšetřovaného pacienta v procesu získání výsledků. [1]

Hlavním zástupcem druhů vyšetření objektivních metod je tympanometrie. Metoda tympanometrická funguje na principu tlaku akustické energie působící silou na plochu ušního bubínku. Touto metodou se zjišťuje přenosová schopnost ušního bubínku. Tympanometrická křivka je pak výsledkem, a to ve formě grafu závislosti přenosové vlastnosti bubínku a tlaku ve vnější části pacientova ucha. Druhů vyšetření v rámci subjektivních metod vyšetření je celá řada. Jsou označovány jako metody subjektivní audiometrické. Většina praktických postupů v rámci jednotlivých metod má pojmenování po svém autorovi. Jedná se například o Weberovu, Rinného, Schwabachovu, Fowlerovu, Langenbeckovu, Kietzovu zkoušku sluchu. Postupy se různí technickými a praktickými předpoklady s podobnými nebo výrazně jinými postupy vyhodnocování výsledků. Subjektivní audiometrické metody jsou základní a speciální. [1]

Do subjektivních metod základních vyšetření patří klasická sluchová zkouška a prahová audiometrie. Klasickou zkoušku sluchu dělíme do dvou kategorií, a to podle způsobu vyšetření. Klasická zkouška může být buď zkouškou řečí (řeč hlasitá a šepot) nebo zkouškou ladičkami (členěnou dle způsobu provedení na Weberovu, Rinného, Schwabachovu metodu). Obecně se jedná o metodu orientační. Při klasické zkoušce řečí lékař pacientovi předříkává slova a zjišťuje mezní vzdálenost správného doslechu slov u pacienta, který slova opakuje. Klasická zkouška sluchu pomocí ladiček je použitelná pro vzdušné i kostní vedení akustického signálu. Prahová audiometrická metoda se vyznačuje svým cílem, a to určit individuální práh sluchu pacienta pro určité zvukové frekvence. Měření se začíná při frekvenci 1 kHz a pokračuje přesně definovanou posloupností hodnot frekvencí zdroje akustického vlnění. Určuje se hodnota, o kterou musí být intenzivnější zvuk dané frekvence, aby ho pacient se sluchátky zaslechl. [1] Pro vyšetření kostního

vedení je použit kostní vibrátor generující tón dané frekvence. Zjištěné individuální prahy sluchu pacienta (v dB) pro jednotlivé frekvence (v Hz) se vynáší do grafu - audiogramu. Grafické závislosti v podobě lomených úseček se konkrétněji dále člení podle strany vyšetřovaného ucha a způsobu provedení vyšetření - vzdušného nebo kostního. [1]



Obr. 1. Ukázka použití audiogramu pro grafické znázornění sluchových prahů [1]

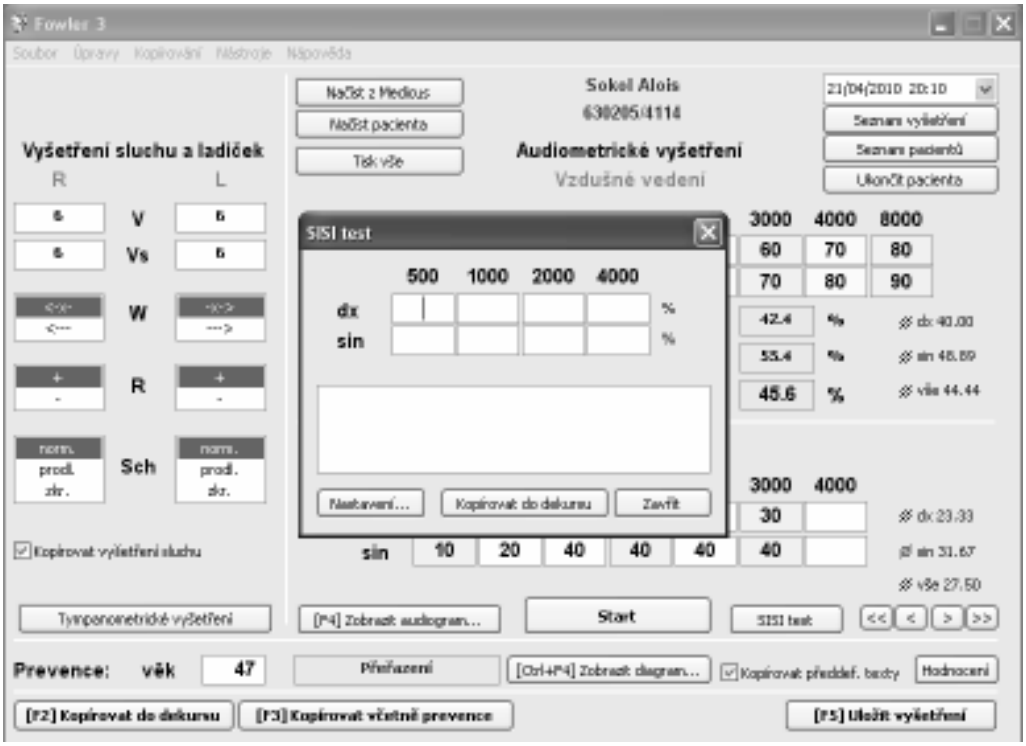
Do speciálních metod se řadí zkouška Fowlerova používaná pro určení jednostranné poruchy sluchu. Fowlerova zkouška vychází ze vzájemného porovnání hlasitostí tónů stejné frekvence u ucha zdravého a potenciálně postiženého. Speciální metodou je také SISI test, založený na procentuálním vyjádřením, a to počtu pacientem rozpoznaných impulsů v závislosti na přesně definované změně intenzity. [1]

Lékař výsledky provedených vyšetření interpretuje do závěrů o stavu sluchovém ústrojí vyšetřovaného pacienta. Podle záznamů předchozích vyšetření může také zjistit, zda-li se stav zlepšil nebo případně zhoršil. Obecně se typy sluchových vad a poruch, které lékař v ORL ordinaci diagnostikuje, člení na poruchy nedoslýchavosti a hluchoty. Kromě samotných poruch sluchového ústrojí může být závěrem lékaře také konstatování, že má pacient sluch v pořádku. V opačném případě lékař určí léčebný postup, například doporučí používání naslouchadla. [1]

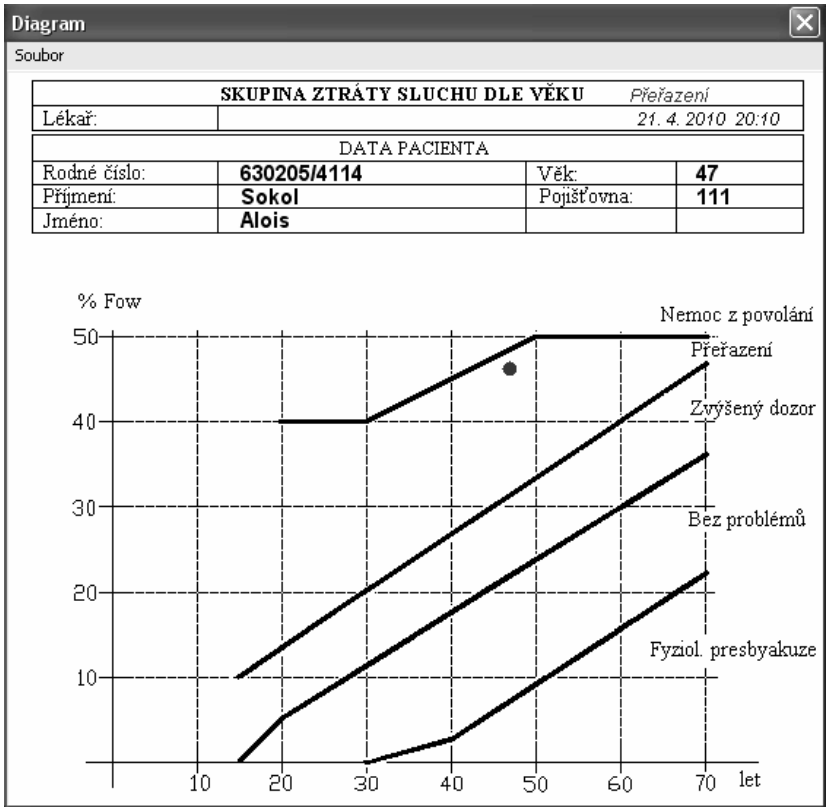
1.2 Popis systému Fowler používaného v ORL lékařství, způsob archivace dat systémem

Protože praktická část diplomové práce vyhodnocuje statistické souhrny počtů již provedených vyšetření, musí proto pracovat s již existujícími daty, které jsou již dávno předtím vytvořeny. Tato data mohou být v praxi ukládána existujícím lékařským programem Fowler, aplikovatelným v lékařských ordinacích specializace ORL. Tento softwarový program slouží lékařům jako komplexní systém pro vyšetření funkcí sluchového aparátu. Protože je pro tvorbu praktické části znalost tohoto systému a principy způsobu ukládání jeho dat důležité, provedl jsem dále zmapování významných funkcí tohoto v lékařské praxi používaného programu Fowler. [15]

Program Fowler má množinu funkcí určenou pro lékaře a pro zdravotní sestřičku. Ve většině funkcích se obě množiny shodují. Lékař i zdravotní sestřička mohou používat jeden společný počítač nebo dva různé počítače spojené počítačovou sítí. V programu se definuje databáze pacientů, kteří ordinaci navštíví. Každému pacientovi jsou v jeho záznamu v databázi přiřazeny identifikační údaje - rodné číslo nebo identifikační číslo v případě pacienta jiné národnosti než-li české, dále může být zadán číselný kód zdravotní pojišťovny. Pokud má v databázi pacient svůj osobní záznam, mohou být lékařem tomuto pacientovi přidány do databáze výsledky konkrétních audiologických vyšetření, kterými jsou tympanometrické, audiometrické, vyšetření sluchu klasické spolu se zkouškou sluchu pomocí ladiček a SISI test. Data umožňuje program načítat přímo z lékařských přístrojů nebo je možno některé typy výsledků zadávat do programu přímo. Program automaticky počítá lékařsky směrodatné výpočty, kterým je především procentuální hodnota ztráty sluchu podle Fowlera. Procentuální hodnoty jsou tři, a to pro oboustranná měření a jednostranná měření každého ucha zvlášť. Pro výsledky audiometrického vyšetření vytváří program v přehledné formě audiogram, znázorňující naměřené sluchové prahy pacienta. Pro tympanometrické vyšetření pacienta dokáže vyobrazit tympanogram. Program dále umí vytvořit přehledné okno s grafickým vyznačením zařazení pacienta do skupin podle poškozeného sluchu a věku. Samozřejmou funkcí programu je vestavěná spolupráce s řadou dalších ambulantních programů jako jsou Medicus a PC Doktor. Program Fowler je komplexní a specializovaný systém, který má spoustu dalších funkcí a detailních nastavení v rámci všech uvedených funkcí. [15]



Obr. 2. Ukázka systému Fowler se zkouškou vložení dat fiktivního pacienta [15]



Obr. 3. Ukázka diagramu ze systému Fowler pro fiktivního testovacího pacienta [15]

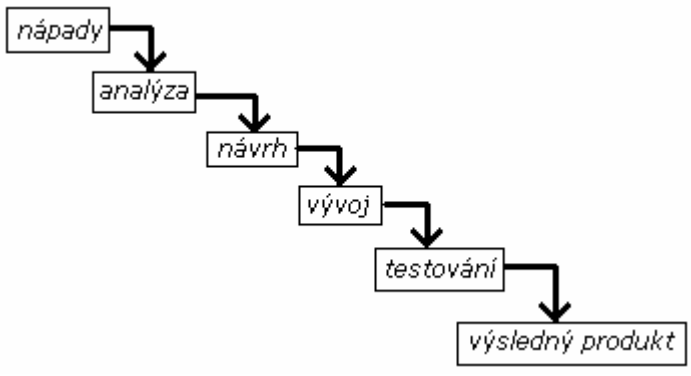
Nyní se zaměřím na způsob ukládání dat vyšetření systémem Fowler, tím provádím analýzu způsobu ukládání dat jednotlivých vyšetření. Pro praktickou část diplomové práce nebyly podstatné přímo hodnoty výsledků vyšetření, ale údaje statistického charakteru. Všechna data program Fowler ukládá ve formě textových souborů do v programu stanoveného adresáře. Tento adresář zahrnuje databázi pacientů, a to v souboru *patients.xml*. Každý pacient pak má dále svůj vlastní soubor s příponou *.xml*, ve kterém jsou vedeny konkrétní výsledky vyšetření a informace o jejich průběhu. Soubor *patients.xml* zahrnuje identifikační údaje u každého pacienta (klíčový identifikátor pacienta v databázi, jméno, příjmení, rodné číslo, číselný kód pojišťovny, případně datum narození). Konkrétní soubor dat vyšetření pacienta (s příponou *.xml*) je pak pojmenován shodně jako je hodnota jeho klíčového identifikátoru v databázi například *22213766-1996-4792-ABA3-D30E5C57C6AA.xml*. Obsahem tohoto souboru jsou záznamy o provedených vyšetření s konkrétními hodnotami. Při aktuálním datumu a času je přidané vyšetření přesně v souboru definováno a vyjádřeno, zahrnuje již konkrétní množinu záznamů vyšetření. Každý záznam je uvozen konkrétním názvem typu z pohledu audiologie, kterému odpovídá. Tímto způsobem lze z databáze zjistit, jak údaje o jednotlivých pacientech, tak i přehledné uspořádání dat konkrétních provedených vyšetření. Protože jsou data konkrétně ukládána do souborů pomocí principů jazyka XML, budu v rozboru archivace dat systémem Fowler navazovat v kapitole 5.2 této práce, až po vysvětlení obecných principů technologie XML. [15]

2 POSTUP TVORBY SOFTWAROVÝCH APLIKACÍ

Výsledkem diplomové práce je vytvoření vyhodnocovacího systému pro lékařské ORL pracoviště. Tvorba každého systému se dělí na definovatelné fáze, které by měl správně každý vývoj softwarové aplikace obecně zahrnovat. Dle těchto fází je řízen proces tvorby programu a také osnova jeho popisu v praktické části.

2.1 Volba metody vývoje softwarového produktu

Za účelem vnést do procesu tvorby softwarového produktu deterministická pravidla, byly definovány modely životního cyklu vývoje softwaru. Modelů životního cyklu existuje několik. Jednotlivé modely se přitom navzájem liší svou filozofií, avšak všechny zahrnují jednotný cíl - dostat se z fáze počátečního stanovení záměrů k výslednému produktu. Jsou definovány základní modely životních cyklů vývoje, případně se objevují jejich variace. Nejčastěji volenými modely bývají vodopádový model, spirálový model, model programování a oprav, model velkého třesku. Poslední uvedený model zahrnuje neorganizovaný vývoj bez jakéhokoliv detailního plánování s pouhým cílem dostat se k výsledkům, a proto jej zmiňuji jen pro úplnost, neboť není efektivní. Také model programování a oprav není zcela ideální. Připouští totiž jen hrubé zpracování produktu s neustálými opravami, pro případ nutnosti zahrnutí detailů, nedbá na důkladné stanovení požadavků v úvodu. Efektivními a nejčastěji používanými modely jsou vodopádový a spirálový model. Model vodopádový se vyznačuje především svou jednoduchou koncepcí, je přehledný, a proto jsem při tvorbě programu volil právě tento postup. Spirálový model je komplikovanější než-li předchozí modely. Vyznačuje se tím, že cyklicky opakuje činnosti, které se vyskytují v ostatních modelech.[2][3] Na schématu (Obr. 4) uvádím postup návrhu programu podle vodopádového modelu.



Obr. 4. Postup vývoje softwarového produktu dle vodopádového modelu [2]

2.2 Fáze specifikace a analýzy požadavků

Úvodní fází tvorby softwarového produktu je specifikace a analýza požadavků, ujasnění nejdůležitějších záměrů. Podle specifikovaných požadavků si lze již vytvořit konkrétní představu o podobě výsledného produktu, naplánovat postup tvorby, vybrat správnou metodiku a volit také implementační prostředky. Cílem je mít představu o všech klíčových vlastnostech softwaru a stanovit směr práce na projektu. Důležitými otázkami, nad kterými by se měl autor zamyslet, jsou důvody vzniku systému, oblasti nasazení systému, časový horizont jeho přibližného dokončení a spektrum koncových uživatelů systému. Proto, aby produkt splnil požadavky zadavatele, je třeba zmapovat, v této rané fázi tvorby projektu, veškeré specifikace vztažené na výsledný systém. Specifikace požadavků je převážně obecného charakteru. Požadavky potvrzují výčet funkcí, které by měly být v produktu zahrnuty. Není podmínkou, aby byly specifikace detailní. Jedná se o vize, myšlenky vzniku systému, stanovení oblasti nasazení systému a také určení směru jeho rozvoje. Obecně se však jedná o seznam funkcí, které zadavatel po systému požaduje. Nároky na přesný formát specifikací souvisí s plánovaným využitím systému. Detailnější formát bude vyžadován pro aplikace v kritických oblastech, kde je nepřípustné příliš zobecněné definování požadavků. Jedná se především o lékařství, finanční, státní, vojenské, letecké sektory. U tohoto typu požadavků obvykle, v průběhu řešení projektu, minimálně dochází k jejich redefinici, například pouze v případech mimořádných situací. [2][3]

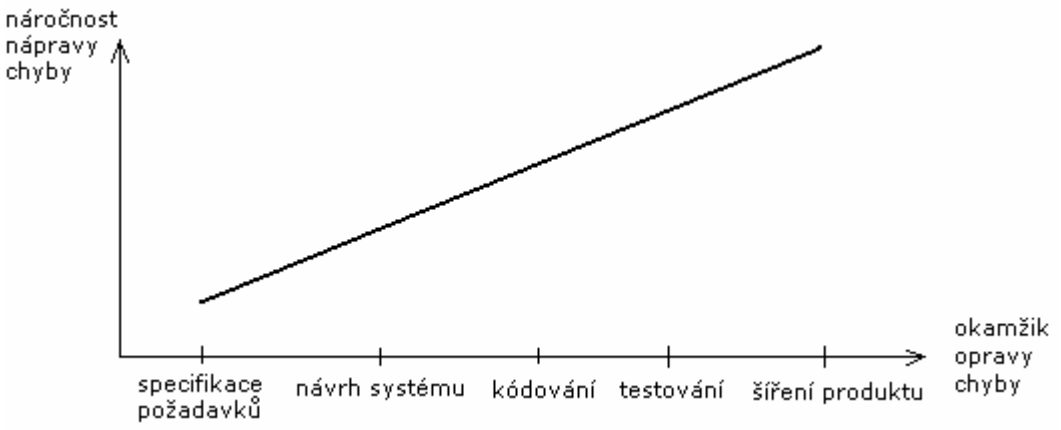
2.3 Fáze návrhu a vývoje výsledného řešení

Pokud jsou shromážděny požadavky na výsledný produkt, může nastat fáze návrhu systému. K etapě návrhu lze dospět analýzou specifikovaných požadavků a zvážením představ o finální podobě produktu. Po úspěšném dokončení návrhu následuje fáze vývoje, při níž je, vhodně zvolenými programovými prostředky a technikami, implementováno výsledné řešení projektu. Při implementaci je doporučeno detailně komentovat programový kód. [2][3]

Samotná etapa návrhu obvykle zahrnuje stanovení architektury a návrhové diagramy (diagram toku dat, diagram stavů a přechodů, diagram toku řízení). Podstatnou činností je stanovení architektury produktu. Architektura specifikuje celkovou strukturu softwarové aplikace s popisem jejích významných součástí, včetně vzájemné komunikace mezi nimi. Činností při definici architektury může být volba typu databáze, formát uložení dat, definice struktur aplikace a návrh objektů v nich obsažených. [2][3]

2.4 Fáze testování

Obecně je cílem testování odhalovat chyby v systému, lokalizovat jejich zdroj a zajistit nápravu. S rostoucí složitostí systémů teoreticky nepřímo úměrně klesá pravděpodobnost detekce veškerých chyb. Tuto skutečnost může také umocnit existence chybových faktorů, které se v systému vyskytují vzácně a je velmi těžké je objevit, jejich možnou přítomnost předpovídat a následně lokalizovat. Důvodem snížení pravděpodobnosti detekce všech chyb může být také příliš velká a rozsáhlá testovací množina vstupních hodnot a počet jejich kombinací pro řešený problém. Důležitým pravidlem pro reálné testování je zúžení sady variant testů do zvládnutelného počtu s přihlédnutím na míru rizika zanedbání konkrétních variant. Místa a oblasti, v nichž se mohou chyby objevit, jsou programový kód, návrh systému a specifikace požadavků. Právě oblast specifikace a návrhu systému je podle statistik nejkritičtějším místem výskytů chyb. Chyby v těchto dvou oblastech jsou formy jiných očekávání od výsledné podoby produktu. Spektrum uživatelů produktu bude mít vždy rozličné názory nad podobou systému. Tento typ chyb lze již ve výsledné etapě tvorby produktu obtížněji eliminovat, navíc za větších ekonomických nákladů. Proto je oblast návrhu systému a také oblast specifikace požadavků natolik důležitá a významná. Náklady na opravu chyb a náročnost jejich nápravy, s plynoucím časem významně roste dle následujícího grafického vyjádření (Obr.5). Závěrem plynoucím z této skutečnosti je kompletní dodržování činností v rámci jednotlivých fází tvorby systému - přesná definice požadavků a uváženě provedený návrh systému. Existuje řada metod testování softwarových aplikací. Společným průnikem všech těchto testovacích postupů je kontrola správnosti, přesnosti, kvality, spolehlivosti a monitorování chování systému při nadměrné zátěži. [2]



Obr. 5. Náročnost nápravy chyby v různých fázích tvorby produktu [2]

2.5 Testování výpočetní složitost řešení

Aby byla fáze testování kompletní, je doporučené pro rozsáhlejší systémy ověřit výpočetní složitost. Otestovat, zda-li je při větších nárocích na program dosažitelné konečné řešení v přiměřeném čase. Výpočetní složitost především souvisí s nároky na potřebný čas a kapacitu paměti. Podstatnou skutečností je, že časová i paměťová složitost závisí na volbě vybraného postupu řešení. Neefektivní algoritmy mohou ve výsledku značně ovlivnit vlastnosti softwarového produktu, a to konkrétně při zvýšení objemu požadavků na systém. Při malém množství vstupních požadavků se negativní změna charakteristik programu nemusí vůbec projevit. Řešením zlepšení kvality systému je volba co nejlepších dostupných metod a postupů. V některých případech se může stát, že zcela ideální a efektivní způsob najít nelze. K určení časové složitosti je určen specifický matematický aparát, který vyjadřuje závislost výpočetní náročnosti na objemu vstupních dat. Pokud vyznačíme tuto závislost ve formě posloupnosti hodnot do grafu a vhodně aproximujeme regresní funkcí například n^2+4n+3, tak bereme v úvahu pouze část polynomu nejvyššího stupně a navíc bez číselných multiplikativních konstant. Zbytek polynomu zanedbáme nebo-li vynecháme zbývající aditivní části. Výslednou podobou upravené regresní funkce je tvar n^2. Pokud výsledkem testování bude řešení y ve srovnání horší nebo stejné jako reálná funkce f (pro uvedený příklad platí, že $f \sim n^2$), použijeme pro formální popis asymptotické složitosti řešení zápis $y \in O(f)$. V tomto případě může y řádově růst stejně rychle jako funkce f nebo pomaleji. Pokud pouze pomaleji, platí zápis $y \in o(f)$. Pro srovnatelný asymptotický růst funkcí y a f platí vyjádření $y \in \Theta(f)$. Funkce y tak nebude růst rychleji ani pomaleji než-li funkce f. Běžnými srovnáními jsou $\Theta(1)$, kdy je složitost neovlivněna objemem vstupních dat, dalšími jsou $\Theta(\log n)$, $\Theta(n)$, $\Theta(n.\log n)$, $\Theta(n^k)$ $k \in N$, faktoriální růst $\Theta(n!)$. [7]

2.6 Výsledný softwarový produkt

Po dokončení prací na softwarovém produktu nastává okamžik, kdy je důležité se zamyslet nad formou a způsoby jeho distribuce. Mezi součásti, které do výsledného produktu spadají, se řadí také nápověda nebo tutoriál s ukázkovými příklady, návody na instalaci, informace o podpoře produktu, reklamy, chybové zprávy a instalační spustitelné soubory. Nápověda k programu by měla zahrnovat návod na instalaci programu a jeho součástí, manuál obsahující přehledný popis funkcí programu nebo také multimediální tutoriál s ukázkami práce s programem. [2][3]

3 PROGRAMOVACÍ PROSTŘEDKY A NÁSTROJE

Programovacích prostředků a nástrojů pro tvorbu počítačových okenních programů je celá řada. Při tvorbě praktické části diplomové práce jsem se zaměřil na volbu nástrojů, které jsou volně přístupné, se kterými jsem i podrobněji pracoval během studia.

3.1 Volba technologie wxWidgets pro tvorbu okenních aplikací

Pro vytvoření okenní programové aplikace je možné využít technologii wxWidgets. Jedná se o programovou knihovnu vytvořenou jejími tvůrci v jazyce C++. Výsledný program je nutné z tohoto důvodu psát také v jazyce C++. Standardními technikami objektově orientovaného programování lze využít ve vyvíjeném programu prvky a principy této programovací knihovny. Pomocí knihovny wxWidgets můžeme naprogramovat okenní aplikace, které vycházejí z komplexních požadavků, kladených na současné programy. Okenní aplikace, vyvinuté touto technologií, jsou určeny pro v dnešní době nejčastěji instalované operační systémy. Proto je funkcionalita aplikací, kterou jim knihovna wxWidgets vnáší, založena především na principech zpracování událostí operačního systému. Vytvořený program se pak chová v operačním systému přirozeně a může mít definovanou indikaci a obsluhu jeho událostí (např. stisknutí klávesy, pohyb myši nad programem). Program může být přeložen, do spustitelné podoby, vývojovými nástroji pro řadu různých operačních systémů, kterými jsou MS Windows, Linux a jeho distribuce, Macintosh a široké spektrum dalších. Kód programu je pro různé operační systémy stejný. Touto vlastností se wxWidgets vyznačuje. Pokud při přenesení programu na různé operační systémy zůstává kód nezměněn, hovoří se, že technologie, jíž je program vytvořen, je multiplatformní. Výhodou při programování s wxWidgets je také samotné psaní zdrojových kódů programu v jazyce C++. Jazyk nabízí možnost modifikovat již předdefinované prvky knihovny wxWidgets dle svého zvážení nebo odvozovat od existujících prvků knihovny nové vlastní prvky, ať už vizuální grafické nebo nevizuální logické. Kromě vytvoření základu aplikace, s přirozeným fungováním pod operačním systémem a zpracováním událostí uživatele, je možnost využít celé řady dalších prvků a technik, které knihovna wxWidgets nabízí. Z pohledu programování se techniky a řešení knihovny wxWidgets nachází v hierarchii tříd, ze kterých lze tvořit objekty vizuálního či logického charakteru a začlenit je do programu (např. dialogové okno, tlačítka, logické úložné kolekce pro data). Pro vývoj okenních aplikací je použití knihovny wxWidgets výhodným řešením. [4][5][6]

3.2 Volba programovacích nástrojů, jejich instalace a konfigurace

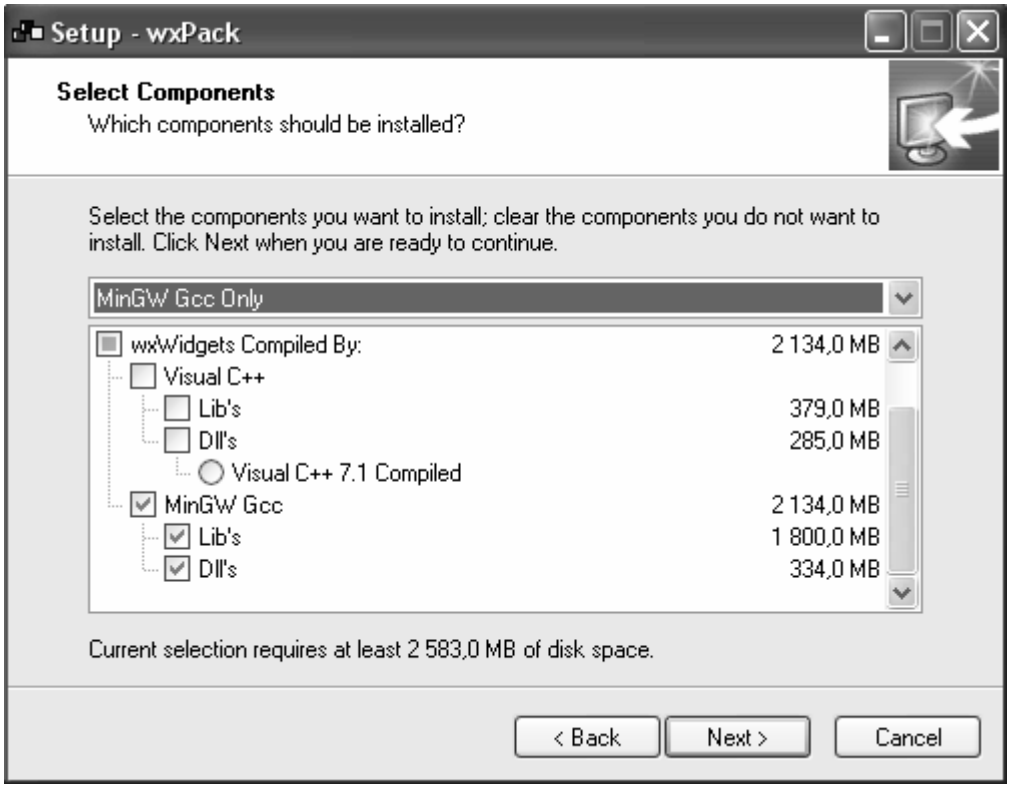
Pro programování okenních aplikací pomocí knihovny wxWidgets je třeba nejprve nainstalovat tři základní celky, kterými jsou samotná knihovna wxWidgets, editor kódu a překladač. Po instalaci je třeba programovací nástroje správně nakonfigurovat pro jejich správné fungování. [4]

Protože je programovacích nástrojů celá řada a zmíněné tři celky musí být navzájem slučitelné po funkční stránce, je vybrána pro vývoj praktické části práce kombinace tří nástrojů, jejichž instalaci nyní vysvětlím. Jedná se o knihovnu wxWidgets verze 2.8.5, editor CodeBlocks a překladač MinGW. Protože není vyhledání správné kombinace vývojových nástrojů nejjednodušší, umístil jsem na CD diplomové práce (Příloha P I) instalační soubory těchto nástrojů, které jsou volně dostupné. [6][19][20]

Nejprve je nutné nainstalovat na počítač editor kódu a překladač. Editor kódu CodeBlocks lze získat ze zdroje [19], překladač kódu ze zdroje [20]. Na přenositelném CD příloze P I jsem instalační soubory editoru a překladače umístil do adresáře *zdrojove_kody /programovaci_nastroje*. Soubory jsou archivační a mají pojmenování *mingw.zip* a *CodeBlocks.zip*. Instalace probíhá jejich dekomprimací do adresářů *mingw* a *CodeBlocks*, které je doporučeno vložit například do umístění *C:*. Záleží na struktuře disků v počítači. [19][20]

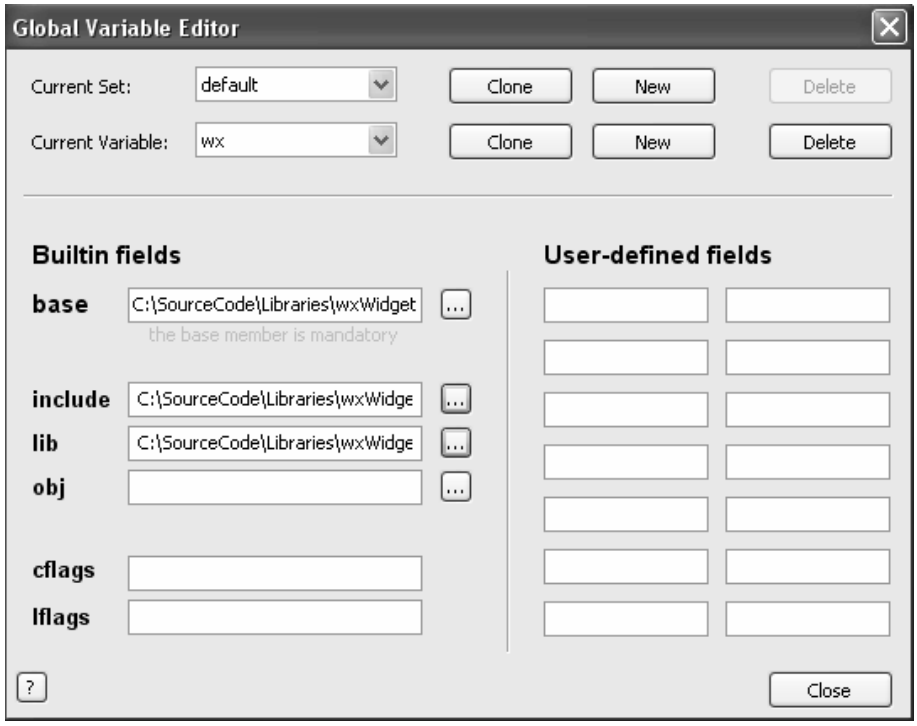
Konfigurace editoru kódu CodeBlocks a integrace překladače MinGW probíhá po prvním spuštění programu CodeBlocks. Editor automaticky detekuje přítomnost MinGW a chce po uživateli tuto automatickou detekci potvrdit. Po potvrzení se editor dotáže, zda-li používat CodeBlocks jako výchozí editor úpravy souborů typu *.h a .cpp*. Tato možnost asociace je volitelná. V tomto okamžiku můžeme CodeBlocks zavřít a věnovat se instalaci knihovny wxWidgets na počítač. [19][20]

Knihovna wxWidgets je také volně dostupná, tak jako editor a překladač kódu. Pro zvolenou kombinaci verze překladače a editoru jsem použil verzi knihovny 2.8.5. Instalace knihovny je usnadněna instalačním balíčkem *wxPack*, který jsem umístil na přenositelné CD přílohu P I do adresáře *zdrojove_kody /progra*movaci_nastroje s názvem *wxPack_v2.8.5.01.exe*. Instalační balíček lze získat ze zdroje [16], je volně dostupný. Instalace knihovny wxWidgets probíhá po spuštění instalačního souboru *wxPack_v2.8.5.01.exe*. Po uživateli je požadován výběr adresáře a součástí knihovny. [16]

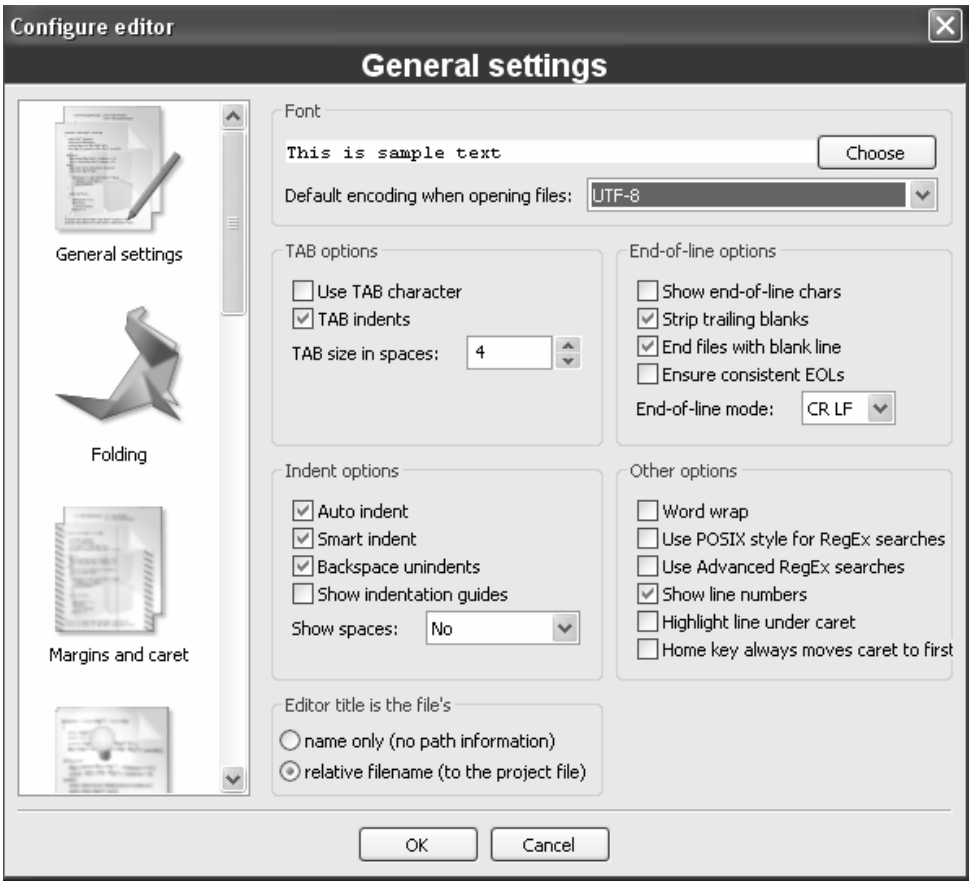


Obr. 6. Ukázka kroku instalace knihovny wxWidgets instalačním balíčkem wxPack [16]

Instalace *wxPack* balíčku trvá přibližně 30 minut. Knihovna wxWidgets je poté nainstalována, ale pro její použití je nutné provést v editoru CodeBlocks nastavení, a to nadefinování cesty ke knihovním adresářům. Nastavení je nejjednodušší provést tvorbou nového projektu přes menu *File > New >Project, wxWidgets projekt*. Průvodce po vašem zadání základních informací o projektu (jméno, umístění) sám zajistí vyobrazení konfiguračního okna (Obr.7). V dialogu je požadováno nastavení cest k souborům knihovny. Položka *base* specifikuje základní adresář knihovny po instalaci (např. *C:\SourceCode\Libraries\wxWidgets* nebo jiné Vaše umístění). Položky *include* a *lib* vyjadřují adresáře v rámci *base* položky. Po určení těchto tří parametrů je třeba potvrdit zadání konfigurace tlačítkem *Close*. Někdy se může zobrazit po stisknutí tlačítka *Close* chybové oznámení, které je doporučeno potvrdit, přejít a tvorbu projektu průvodcem dokončit. V editoru lze dále přes menu *Settings>Editor* nastavit znakové kódování např. UTF-8 (Obr.8). V novém projektu editoru CodeBlocks nyní může psát programy využívající prostředků knihovny wxWidgets. Skrze nabídku *Start* systému MS Windows lze dále v podnabídce *wxPack* najít nástroj pro tvorbu designu programů wxFormBuilder [21] a kompletní nápovědu knihovny wxWidgets [6]. [4][16][19]

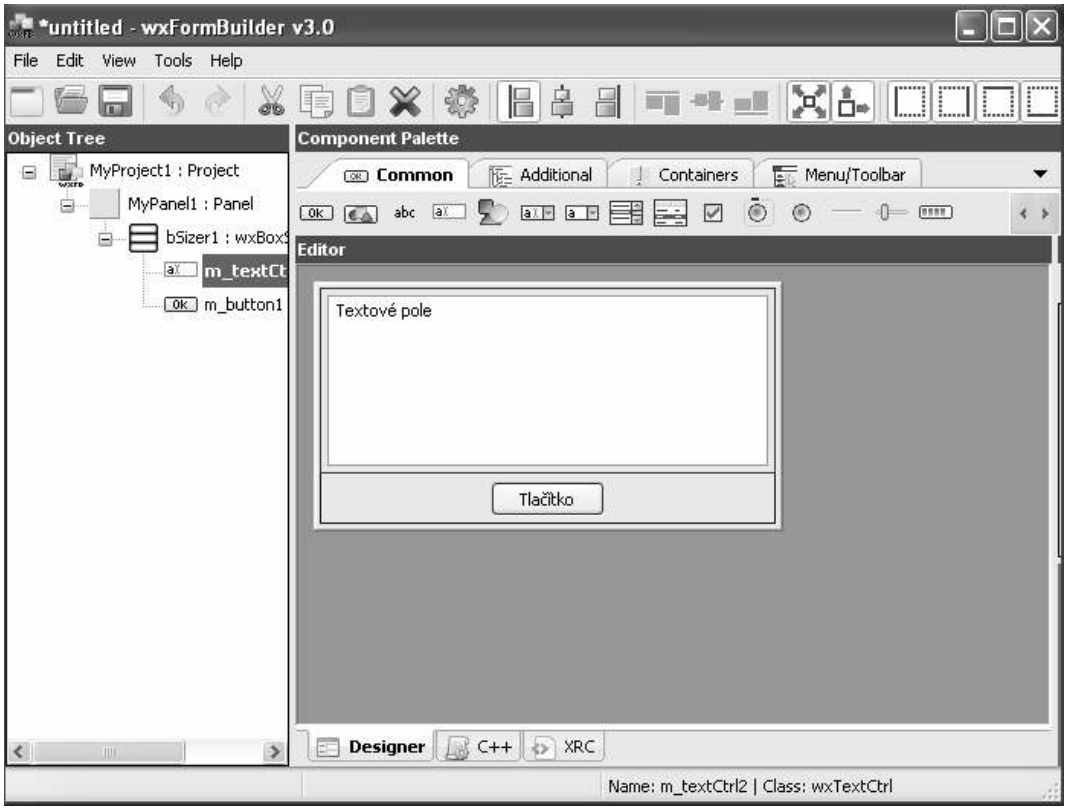


Obr. 7. Konfigurace knihovny wxWidgets v editoru CodeBlocks [19]



Obr. 8. Nastavení kódování znaků na UTF-8 pro editor CodeBlocks [19]

Překlad kódů programu probíhá v editoru CodeBlocks přes menu *Build>Build and run* nebo přes klávesovou zkratku *F9*. Spustitelný program lze najít v adresáři projektu programu */bin/Release*. Pro návrh designu obsahu oken námi tvořeného programu, tedy pro určení skladby ovládacích prvků, lze použít editoru designu wxFormBuilder (Obr.9) dostupného z nabídky *Start* pod položkou *wxPack/wxFormBuilder/*. Pomocí programu wxFormBuilder lze jednoduše provést návrh designu např. dialogového okna s ovládacími prvky a do kódu v editoru CodeBlocks pouze vložit kódový výstup z programu wxFormBuilder. Jedná se o návrhový program, který vytvoří kódem kostru návrhu. Je dále nutné detaily prvků návrhu dále upravit a dodefinovat, a to ručním nastavením ve wxFormBuilderu nebo přímo v kódu v editoru CodeBlocks. [19][21]



Obr. 9. Ukázka návrhu designu okna programu nástrojem wxFormBuilder [21]

Pokud má knihovna využívat ovládacích prvků, které standardně nezahrnuje, pak je potřeba učinit jejich manuální doinstalování. Trend přidávání nových prvků do knihovny wxWidgets přináší rozšíření možností této knihovny o nové techniky a funkce. [4][17]

4 PRVKY A TECHNIKY VÝVOJOVÉ KNIHOVNY WXWIDGETS

V kapitole o programovacích prostředcích a nástrojích jsou uvedeny základní charakteristiky knihovny wxWidgets pro vývoj okenních aplikací. Knihovna nabízí velkou množinu tříd s funkcemi, které mohou ve svých programech programátoři využít a dále přetvářet do konkrétních podob [4]. Nyní bude uveden teoretický základ k těm prvkům a technikám knihovny wxWidgets, které jsem nejčastěji použil při tvorbě praktické části. Výklad je zaměřen na významnější prvky a techniky s uvedením informací v terminologii objektově orientovaného programování. Komplexní vysvětlení principů programování s knihovnou wxWidgets mohou programátoři čerpat z programové dokumentace knihovny wxWidgets [6] s detailními popisy knihovních tříd, popřípadě z odborných knih [4] a [5].

4.1 Technika tvorby aplikace, zpracování událostí

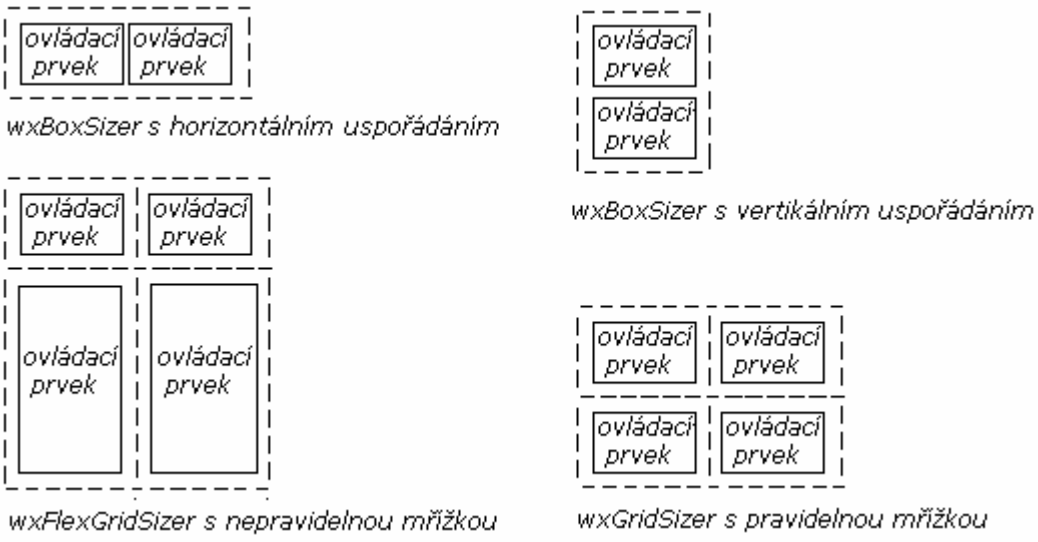
Pro tvorbu programu je na počátku nutné vytvořit logický základ nazývaný aplikace. Pro definici tohoto stavebního základu slouží makro *IMPLEMENT_APP*. Třída knihovny wxWidgets zapouzdřující aplikaci se nazývá *wxApp*. Je nutné odvodit vlastní třídu např. *CAplikace*, odvozenou právě od třídy *wxApp*, a to klasickými technikami objektového programování. Třída umožňuje definovat v rámci metody *OnInit* kód, který je proveden v okamžiku prvotního spuštění aplikace. Zde se přidává například kód vytvoření a vizualizace hlavního grafického okna programu nebo obecně libovolný kód, který se má provést ihned po startu. Univerzálně platí, že pro vlastní použití jakékoliv předdefinované třídy knihovny wxWidgets je nutné vždy třídu odvodit děděním od knihovní třídy a poté provádět vlastní zásahy a změny pouze do této vlastní třídy. Prostřednictvím operace dědění odvozená třída zahrnuje veškerou funkčnost, kterou má třída děděná. [5][6]

Třída objektu odvozená od knihovní třídy wxWidgets může zahrnovat tabulku událostí, definovatelnou příkazem *DECLARE_EVENT_TABLE*. Tabulka událostí je součástí mechanismu zpracování událostí, které vzniknou v operačním systému nad tímto konkrétním objektem, v němž je tabulka definována. V tabulce se definuje typ události (např. *EVT_BUTTON)*, identifikátor prvku a jméno metody, která se má spustit jako reakce na vzniklou událost. Samotná definice tabulky je uvozena pomocí *BEGIN_EVENT_TABLE* a *END_EVENT_TABLE*. [5][6]

Konkrétních druhů událostí je mnoho a někdy vychází právě z potřeb funkčnosti objektů specifických tříd. Dokonce je možné definovat svůj vlastní typ události, a to pokud definujeme například svoje vlastní třídy objektů. Základními typy událostí v knihovně wxWidgets jsou události změny velikosti prvku, rolování posuvníku, událost pohybu myši, kliknutí myši, stisku klávesy, výběru, událost uzavření, aktivace a celá řada dalších. [5][6]

4.2 Ovládací vizuální prvky grafického rozhraní programu

Ovládací vizuální prvky slouží ke zobrazování informací a k interakci s uživatelem programu. Tyto prvky nazývám jako vizuální, protože jsou součástí grafického rozhraní programu a jsou vykreslovány operačním systémem. Operační systém vykresluje základní typy prvků knihovny wxWidgets automaticky, ale můžeme i tvořit své grafické ovládací prvky a volit způsob jejich podoby a zobrazení. Velká řada prvků dokáže zahrnovat v sobě množinu dalších prvků. Takové objekty mají označení kontejnerové objekty, neboť do jejich obsahu může patřit jakýkoliv další objekt. Základními kontejnerovými objekty jsou pro oblast grafické interpretace dialogová okna, rámcová okna a panely. Z důvodu, aby vkládané grafické ovládací prvky byly určitým způsobem na ploše kontejnerového objektu symetricky uspořádány, existuje v knihovně wxWidgets mechanismus zvaný sizery. Jedná se o nevizuální systém vodících čar tvořících podobu mřížky, tabulky. Do každé kolonky takovéto tabulky lze přiřadit umístění konkrétních prvků. Typů sizerů nabízí knihovna wxWidgets celou řadu. Mezi základní používané patří *wxBoxSizer* (obdélníková mřížka), *wxGridSizer* (mřížka s kolonkami konstantních rozměrů), *wxFlexGridSizer* (mřížka s kolonkami různých rozměrů). Každý sizer má nastavitelné vlastnosti, kterými jsou u vícekolonkových mřížek počet řádků a sloupců, u jednokolonkové mřížky volba směru vkládání prvků za sebou v rámci kolonky (vertikální nebo horizontální směr). U *wxFlexGridSizeru* je využitelná možnost definovat řádky a sloupce, v jejichž průniku budou rozměry mřížky měnitelné a velikost prvku uvnitř obsaženého bude maximalizována. Pro tento případ je nutné provést nastavení samotného prvku z pohledu chování vzhledem k sizeru. Prvek musí být nastaven jako expanzivní, aby mohl svůj obsah přizpůsobit právě nadřazené kolonce mřížky sizeru. Sizery mohou být strukturálně vnořovány a ve výsledku splňovat i složité návrhy rozložení prvků na ploše objektu, který mřížku v podobě sizeru obsahuje. [4][5]



Obr. 10. Přehled nečastěji používaných typů mřížek polohového uspořádání prvků [4]

Ať už se jedná o prvky kontejnerové, které dokáží obsahově přijmout prvky jiné, nebo o prvky samostatné, v případě pokud se řadí do oblasti grafických vizuálních prvků, pak je dostupná možnost nastavení jejich vzhledu. Pro konkrétně definované prvky knihovny wxWidgets existuje vždy konkrétní sada nastavení. Nastavení vzhledu prvku lze provést v určitých případech rovnou při vytvoření objektu prvku konstruktorem jeho třídy nebo dodatečně pro již existující objekt, a to funkcemi, které jsou tomu přímo určeny. Jedná se například o funkci nastavení zobrazovaného textu na ploše prvku, povolení ohraničení, barva podkladové plochy, formát písma vyobrazovaného textu a spousta dalších vlastností, v řadě případů specifických právě pro daný typ prvku. [4][5]

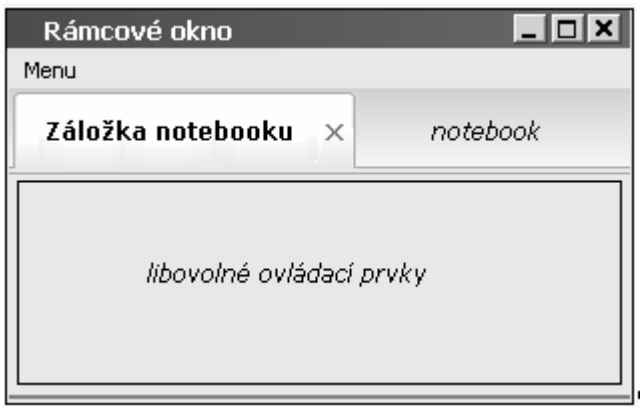
Protože jsem uvedl označení ovládací grafické vizuální prvky, tak bych vysvětlil důvod přidání přívlastku ovládací. Důvodem je skutečnost, že prvek dokáže uživateli nejen poskytovat informace svou grafickou podobou, ale také slouží pro interakci s ním. Pomocí ovládacích prvků uživatel může zadávat do aplikace informace nebo volit různá nastavení. Záleží na podobě prvku. Ovládací prvky mají schopnost generovat události, které mohou být odesílány mechanismu zpracování událostí, buď prvku samotného nebo nadřazeného. Podle definice tabulky událostí pak dojde případně ke spuštění konkrétního kódu, jako reakce na vznik dané události. [4][5]

Každý ovládací prvek musí mít, pro správné zpracování událostí, při svém vzniku definován identifikátor. Jedná se o unikátní číselné označení, které v případě vzniku události vypovídá o zdroji vzniku této události (např. událost stisknutí tlačítka myši). Některé třídy prvků knihovny wxWidgets mají své definované typy událostí. Ovládací prvky můžeme definovat také své vlastní. Máme také možnost svým prvkům nadefinovat typy událostí, aby splnily svůj účel, být ovládacími a interaktivními pro potřeby uživatele. [4][5]

Základem grafického rozhraní aplikace bývá úvodní okno v podobě dialogu nebo rámcového okna. Rámcové okno má zabudovanou funkčnost ve třídě knihovny wxWidgets *wxFrame*. Pro použití rámcového okna v aplikaci je třeba nejprve odvodit svou vlastní třídu od této existující knihovní třídy. Odvozená třída může zahrnovat data a prvky již podle uvážení. Pro existenci rámcového okna je třeba z třídy rámcového okna vytvořit objekt, její instanci. Objekt lze dále nechat vyobrazit uživateli. Třída rámcového okna poskytuje mnoho metod pro nastavení grafické podoby, kterými jsou například nastavení barvy podkladu, nastavení ikony programu, nastavení textu titulkového pruhu okna. Rámcové okno se vyznačuje tím, že může obsahovat programové menu a panel nástrojů. Menu tvoří hierarchie nabídek vnořeně členěných na dílčí nabídky. Položky menu, s přiřazenými číselných identifikátory v kódu programu, dokáží generovat zprávy události typu stisknutí položky menu myší. Tyto události lze dále obsloužit skrze mechanismus tabulky událostí. Na stejném principu funguje také panel nástrojů. Rozdílem oproti menu je obrázková podoba položek namísto podoby textové. [4][5]

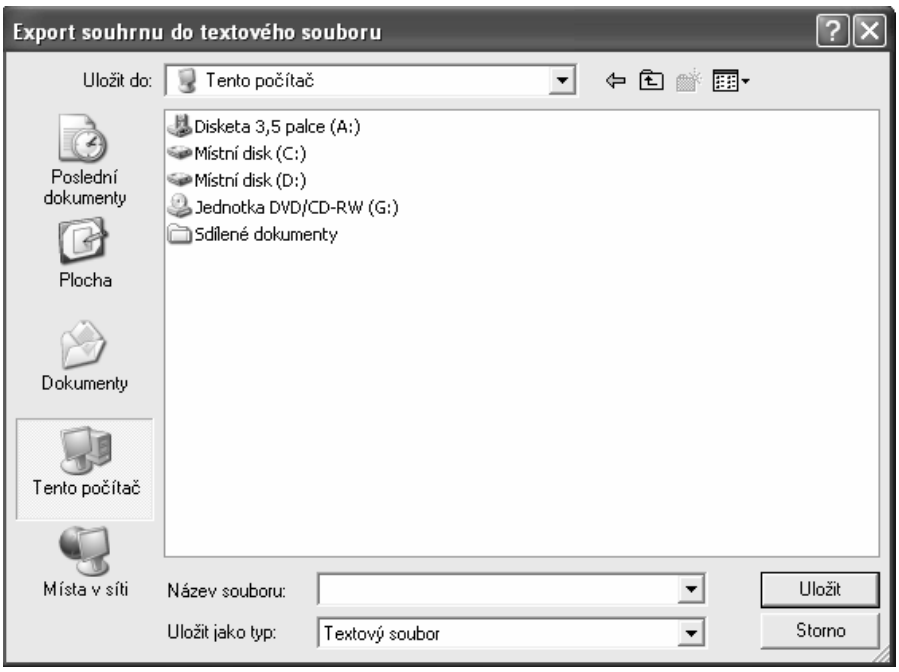
Obr. 11. Ukázka rámcového okna s programovým menu a panelem nástrojů

Rámcové okno bývá používáno především pro spouštění dalších dialogových oken. Výhodným ovládacím prvkem, který je možné na plochu rámcového okna vložit, je prvek notebook třídy knihovny wxWidgets *wxAuiNotebook* nebo *wxNotebook*. Notebook je systém záložek, které mohou obsahovat například panely s dalšími ovládacími prvky. Notebook slouží také jako správce vnořených záložek, které lze přidávat, přepínat a uzavírat. Lze nadefinovat různý styl zobrazení záložek uvnitř notebooku a také přidat každé záložce obrázek před textový popis názvu záložky. Notebook existuje ve dvou variantách *wxAuiNotebook a wxNotebook*. Varianta AUI znamená, že je prvek notebook použitelný v systému AUI. Systém AUI je význačný tím, že spravuje označené prvky, (například uvnitř rámcového okna) a umožňuje uživateli využívat nadstavbové funkce pro práci s ovládacími prvky, kterými je jejich libovolné posouvání a přeskládávání pomocí pohybu stisknuté myši. Systém AUI má za cíl nabídnout uživateli možnost vlastního uspořádání grafické podoby programu. Prvky přidané do režie tzv. AUI manažeru lze jako bloky přesouvat libovolně po ploše okna a tím měnit jeho strukturu. Varianta notebooku *wxAuiNotebook* tak umožňuje oproti základní verzi *wxNotebook* být spravována právě systémem AUI. Konkrétně lze například jednotlivé záložky notebooku libovolně seskupovat v rámcovém okně. Na následujícím obrázku je uvedena ukázka rámcového okna s prvkem třídy knihovny wxWidgets *wxAuiNotebook*, jehož záložky lze myší libovolně v rámci okna uspořádat. Přidávání záložek je dáno kódem programu, ve kterém se například určí, že se záložka s určitým obsahem přidá až po kliknutí na položku menu. Prvek notebook lze také přidávat na obyčejný panel. Panel třídy *wxPanel* je tak jako rámcové okno kontejnerovým typem vizuálního grafického objektu, který dokáže zahrnovat prvky další. Na rozdíl od rámcového okna panel neobsahuje menu, panel nástrojů a okenní titulek [4][5][6]



Obr. 12. Ukázka a schématický popis rámcového okna s prvkem AUI notebook

Kromě panelu a rámcového okna, které je spíše charakteru úvodního okna aplikace, lze ovládací a vizuální prvky vkládat také na dialogové okno. Vkládání prvků probíhá do sizeru, který představuje zmíněnou tabulkovou nevizuální mřížku. Dialog také jako obecný panel neobsahuje programové menu ani panely nástrojů. Na rozdíl od klasického panelu obsahuje titulkový okenní pruh. Dialogy jsou objekty vytvořené ze třídy odvozené z knihovní třídy *wxDialog*. Kromě tohoto typu dialogu existují již předpřipravené kompletní dialogy, které jsou často používané v programových aplikacích. Příkladem jsou dialogy možnosti volby umístění pro uložení souboru, dialog pro výběr souboru, který má být otevřen, dialog změny formátu písma, nastavení barvy z barevné palety. Standardní předpřipravené dialogy mají vestavěné funkce, které umožňují uživatelem zadané nastavení v kódu programu získat. Například standardní dialog volby umístění při uložení souboru dokáže vrátit pomocí definované funkce hodnotu ve formě textu, který obsahuje zadanou cestu k souboru. Velkou výhodou rámcových oken a dialogů obecně je možnost jejich zobrazení ve dvou režimech. Prvním režimem je režim modální, kdy po vlastním zobrazení se dané okno stává výhradním a nelze z něj přejít mimo něj. Druhý režim se nazývá nemodální a při tomto režimu lze ze zobrazeného okna libovolně přejít k práci na jiných oknech, což u modálního zobrazení není povoleno až do okamžiku zavření daného okna. [4][5]



Obr. 13. Příklad standardního dialogu reprezentovaného třídou wxFileDialog

Ovládacích a vizuálních prvků, které lze umístit do kontejnerových objektů, přímým vložením nebo pomocí uspořádávajících mřížek sizerů, je velká řada. Knihovna wxWidgets nabízí sadu tříd pro klasické, z operačních systémů známé, ovládací prvky, ale podporuje i strukturou komplikovanější prvky. S ohledem na složitost struktury prvku se odvíjí počet funkcí a metod uvnitř tříd, které tyto prvky reprezentují. Při vkládání prvku do mřížky kontejnerového objektu je nutné vytvořit opět konkrétní objekt od třídy knihovny wxWidgets. Umístění prvku lze zadat absolutními souřadnicemi nebo relativně použitím mřížky uspořádání. Každý prvek má dostupnou sadu metod pro ovlivnění svého vzhledu zobrazení. Pro prvky obsahující nastavovanou hodnotu (číselnou nebo textovou) v různé formě, existují metody pro její nastavení prvku a také pro její zpětné získání programovým kódem. Některé prvky mohou zahrnovat pouze jednu hodnotu, ale existují i variace ovládacích prvků, které dokáží uchovávat mnoho hodnot. Samozřejmostí tříd prvků je přizpůsobení se mechanismu generování a zpracování událostí. Pro různé konkrétní druhy prvků existují různé typy událostí. Čím je ovládací prvek komplikovanější, tím nabízí více metod a možností pro práci s ním, jak z pohledu programátora, tak i z pohledu uživatele programu. [4][5]

Konkrétním příkladem pouze vizuálních prvků jsou textový popisek *wxStaticText,* indikátor průběhu *wxGauge* a obrázek *wxStaticBitmap* Těmto prvkům se obvykle nepřidává obsluha událostí, jsou používány jako informativní prvky. Mezi prvky vizuální ovládací, které již dokáží generovat zprávy při vzniklých událostech, patří tlačítko *wxButton*, zaškrtávací políčko *wxCheckBox*, přepínací políčko *wxRadioButton,* stromová struktura *wxTreeCtrl,* výběrové pole možností *wxChoice*. Ovládací prvky mohou sloužit pro zadávání hodnot, a to přímo hodnotou nebo jiným speciálním způsobem. Jedná se o prvky textové pole *wxTextCtrl*, tabulku *wxGrid*, tabulku vlastností *wxPropertyGrid,* posuvník s hodnotou *wxSlider*. Na obrázku (Obr.14) jsou vyobrazeny ukázky ovládacích prvků - tlačítko, popisek, zaškrtávací pole, přepínací pole, posuvník s hodnotou, indikátor průběhu, výběrové pole a textové pole.[4][5]

Obr. 14. Dialog s ukázkou klasických ovládacích a vizuálních prvků [4]

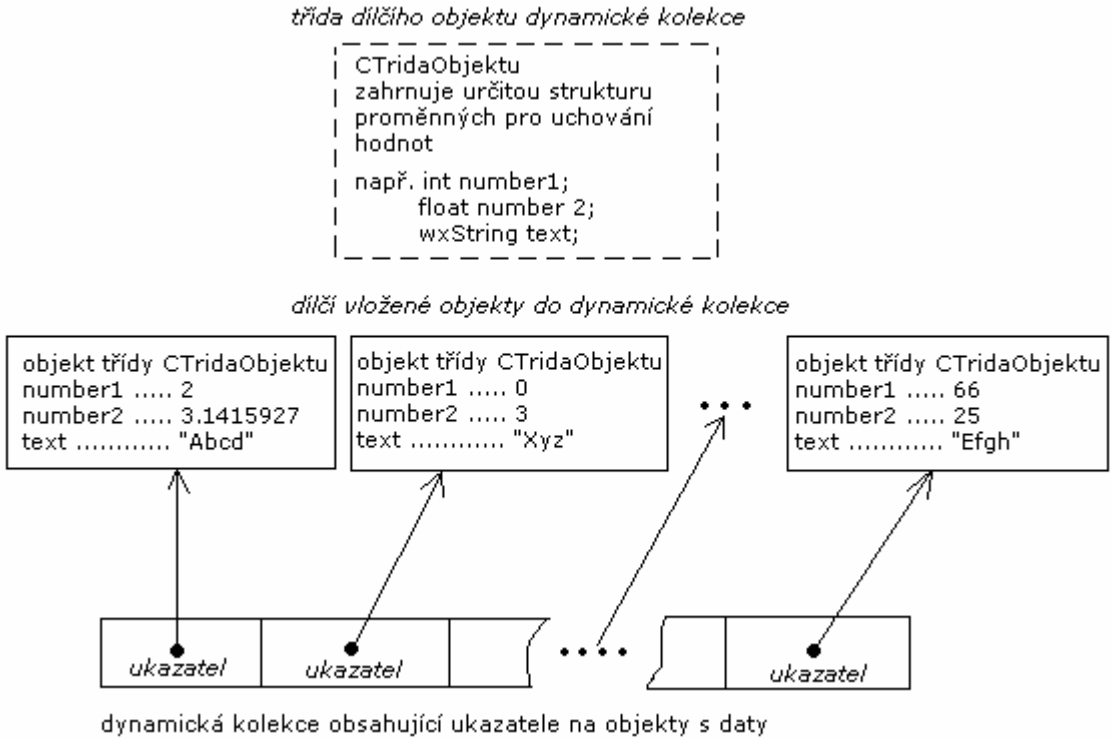
Kromě klasických ovládacích prvků jsem ze složitějších prvků v praktické části použil tabulku vlastností *wxPropertyGrid*, která uchovává hodnoty v rámci rozsáhlejší datové struktury a prvek tabulka *wxGrid*. Pro tyto vizuální ovládací prvky, tak jako pro všechny ostatní, dává knihovna wxWidgets k dispozici sadu funkcí, které jsou pro práci s prvkem z pohledu vývoje velice propracované a detailní. Na obrázku (Obr.15) uvádím příklad vyobrazení tabulky klasické a tabulky vlastností. [4][6]

Obr. 15. Dialog s ukázkou ovládacích prvků klasická tabulka a tabulka vlastností [4]

4.3 Nevizuální prvky knihovny

Hierarchie tříd knihovny wxWidgets zahrnuje třídy pro ovládací prvky vizuální, ale zahrnuje také podmnožinu tříd, které sloužící pro režijní účely nebo se starají o určité mechanismy. Například zajišťují princip, technologii, kterou knihovna nabízí. Důležitým využitím tříd nevizuálního charakteru je uplatnění pro ukládání dat. Nejčastěji používaným nevizuální třídou je dynamická kolekce. Jedná se o strukturu, která dokáže ukládat data. Nyní budu popisovat složitější variantu dynamické kolekce – dynamickou kolekci objektů, kterou je možné využít pro jakoukoliv zvolenou strategii dynamického ukládání údajů. Samotná kolekce se tváří jako obyčejné pole, které má vlastnost, že je svou kapacitou dynamické. Před vznikem kolekce v kódu programu není vyžadováno určení přesné maximální velikosti kapacity kolekce, což je velkou výhodou oproti klasickému poli v jazyce C++.[4][5][6]

Typem prvků dynamické kolekce je ukazatel na objekt, a to té třídy, kterou pro tento účel určíme. Objekt třídy, který vytvoříme může zahrnovat jakkoliv složitě uložené informace. Pro systém kolekce stačí znát pouze ukazatel na objekt této třídy. Dynamická kolekce je tedy polem ukazatelů na objekty třídy, kterou v kódu určíme. Přidání nového záznamu do kolekce probíhá tak, že nejprve vytvoříme dílčí objekt třídy pro uložení konkrétních informací a jako novou položku kolekce uložíme pouze ukazatel na nový objekt. Pro přidávání nových záznamů má kolekce k dispozici vestavěnou podporu ve formě široké škály funkcí, které významným způsobem práci s kolekcí usnadňují. Položky kolekce lze libovolně mazat, skrze libovolný ukazatel kolekce přistupovat k informacím v konkrétním objektu s daty. Tento způsob ukládání dat se zdá být hodně komplikovaný, ale umožňuje velmi dynamické a náročné ukládání informací, kdy práce se samotnými daty a kolekcí je ve výsledku otázkou vepsání několika příkazů v kódu programu. Tento účinný mechanismus pro ukládání dat, je tvořen objekty tříd, které patří do skupiny nevizuálních. Dynamickou kolekci jsem zmínil proto, že je příkladem nevizuálního prvku knihovny a také z důvodu, že jsem ji používal pro ukládání dat v praktické části práce. Na schématu (Obr.16) je struktura dynamické kolekce vysvětlena. V tabulce (Tab.1) jsou uvedeny základní funkce pro práci s dynamickou kolekcí z pohledu programování. [4][5][6]



Obr. 16. Princip a struktura dynamické kolekce objektů [4][5]

Příklady kódu pro práci s dynamickou kolekcí	*Vysvětlení ukázky kódu*
WX_DEFINE_OBJARRAY(CKolekce);	definice kolekce
WX_DECLARE_OBJARRAY(CKolekce,CTridaObjektu);	deklarace kolekce s určením třídy dílčího objektu
CKolekce Kolekce;	deklarace objektu dynamické kolekce
class CTridaObjektu { int cislo; };	třída dílčího objektu kolekce

CTridaObjektu* ukazatel;
ukazatel= new CTridaObjektu;
ukazatel->cislo=2;
Kolekce.Add(ukazatel);
	přidání nového objektu s daty do kolekce

int precteneCislo; precteneCislo=Kolekce.Item(0).cislo;	získání hodnoty první položky kolekce
Kolekce.Count()	funkce získání počtu položek kolekce
Kolekce.RemoveAt(0);	smazání prvního záznamu kolekce
0, 1, 2, ….. , Kolekce.Count() -1	indexování položek kolekce ve volaných metodách

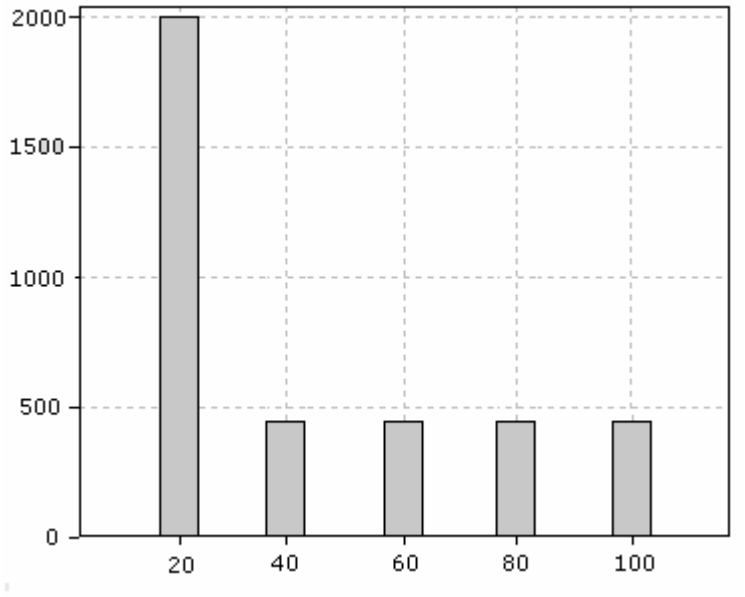
Tab. 1. Ukázky zdrojového kódu pro práci s dynamickou kolekcí objektů [4][5]

4.4 Rozšíření knihovny wxWidgets

Nabídka ovládacích prvků v knihovně wxWidgets je široká, přesto pro speciální řešení určitých problémů ve standardní verzi knihovny programátor nemusí najít vše, co právě potřebuje. Řešením je vytvoření vlastních ovládacích nebo logických tříd prvků nebo použití tříd prvků, které někdo předpřipravil k použití. Doplňkové ovládací a logické prvky lze přidat ke standardní knihovně instalací ze stránky uživatelsky vytvořených doplňků knihovny wxWidgets na internetových stránkách zdroje [17]. V archivačním ZIP souboru bývá zpravidla obsažena nápověda, zdrojové a hlavičkové soubory tohoto ovládacího prvku. Hlavičkové soubory je možné ponechat v takové podobě v jaké jsou uloženy. Výjimku tvoří zdrojové kódy, kde bývá pravidlem vytvořit v programovacím nástroji nový projekt typu knihovna a všechny zdrojové soubory s příponou .cpp sjednotit právě do jednoho knihovního souboru. Při programování našeho programu, který využívá tento nově přidaný prvek stačí nastavit cestu k umístění hlavičkových souborů ovládacího prvku a cestu ke knihovnímu souboru, popřípadě přidat jméno tohoto souboru do výčtu knihoven. Knihovní soubor se liší tím, jaká verze překladače je používána vývojovým nástrojem. Knihovními soubory překladače MinGW jsou soubory s příponou *.a*. Překladače ostatní mohou umožnit překlad do dynamické DLL nebo statické LIB knihovny. [4][17][20]

Pokud by nebylo z technických důvodů možné převést všechny zdrojové *.cpp* soubory do jediného knihovního souboru (*.a, .DLL, .LIB*) existuje možnost přiřadit zdrojové *.cpp* kódy nového prvku ke kódům svého programu a přeložit je tak společně s programem, ale jedná se o neideální postup, protože zdrojové kódy prvku, který vytvořil někdo jiný by měly být vždy oddělené od našeho programu. Řešením je jejich převod do knihovního souboru, který již k našemu programu přidat můžeme. [4][17][20]

V praktické části práce bylo potřeba zobrazit výsledky analýzy dat ve formě sloupcových grafů. Tento typ ovládacího prvku standardní knihovna wxWidgets nezahrnuje, proto jsem jej vyhledal v databázi uživatelsky vytvořených ovládacích prvků knihovny wxWidgets na zdroji [18]. Jednalo se o řešení zobrazení grafů ovládacím prvkem wxFreeChart. Provedl jsem instalaci souborů ovládacího prvku a překlad zdrojových souborů do jediného knihovního souboru. Tento ovládací prvek, tak jako veškeré standardní ovládací prvky, nabízí sadu funkcí , jako jsou například funkce pro definici vzhledu a datového obsahu. Charakteristikou prvku je automatické přidání systému posuvníku s rolováním obsahu, pokud je údajů v grafu více než pojme okno grafu. [17][18]



Obr. 17. Ukázka ovládacího prvku wxFreeChart sloužícího pro zobrazení grafů [18]

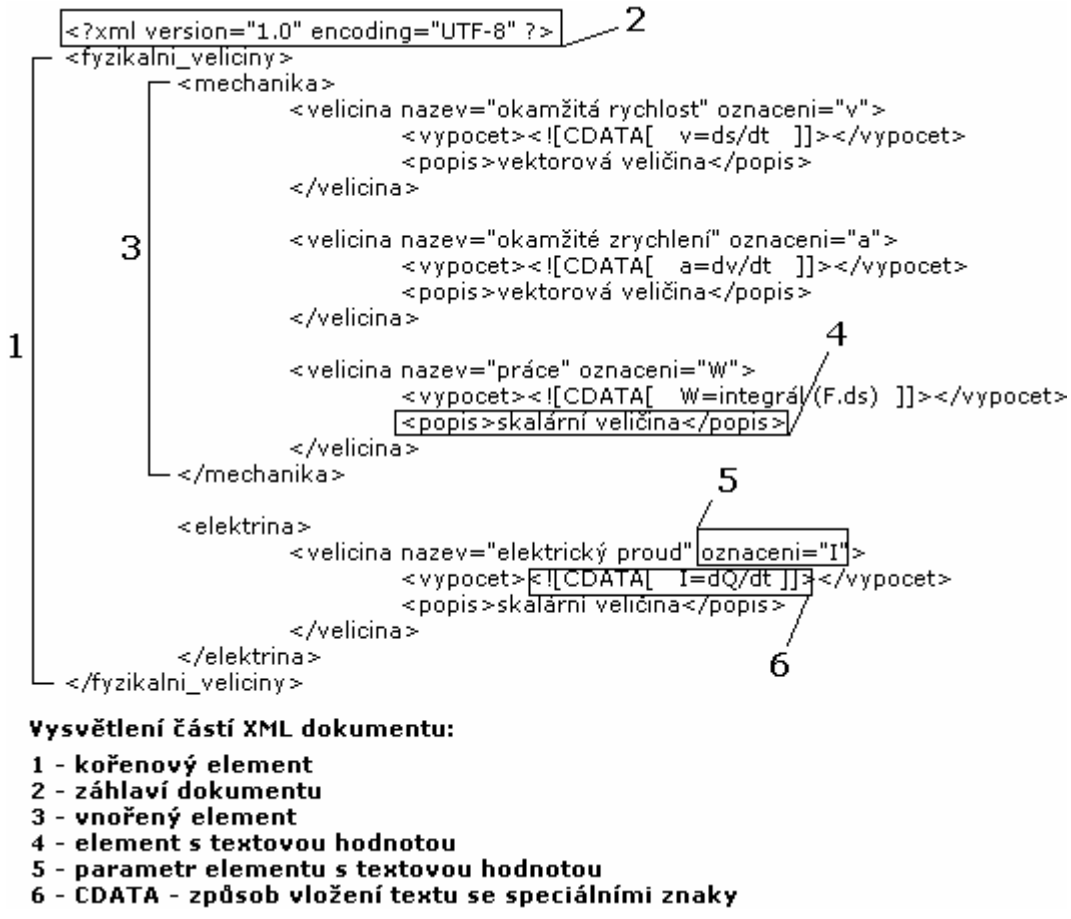
5 TECHNOLOGIE XML

5.1 Účel a principy jazyka XML

Jazyk XML je význačný tím, že umožňuje uchovávat informace v jednoduché textové podobě, a to podle přesně definovaných pravidel. Data jakkoliv strukturálně členěná lze podle pravidel XML uložit v takovém tvaru, který zachovává možnou podporu dalšího datového zpracování. Formát uložení dat totiž není nahodilý, ale je přesně určen specifikacemi jazyka XML. Proto lze také s výhodou použít dokumenty XML při výměně dat mezi více systémy, kdy systémy počítají s předem známou podobou uložení informací v XML souboru. Samotný dokument je pouze nositelem informací zapsaných dle platných pravidel. Dokument je prakticky soubor s textem, proto jiné funkce než-li obsahové, informativní poskytovat ze své podstaty nemůže. Upravovat obsah lze v klasickém textovém editoru. Častou tendencí programátorů internetových stránek je ukládat do XML dokumentů pouze obsah a v souboru konkrétní stránky nadefinovat jen způsob a styl zobrazení těchto dat. Na tomto příkladu je patrný smysl používání jazyka XML, a to pro tvorbu dokumentů, jejichž hlavním cílem je přehledně a korektně ukládat data pro jejich uchování, další možnou správu a zobrazování. [8][9][10]

Základním pravidlem pro tvorbu XML souborů je dodržování specifikací jazyka, které vyjadřují myšlenku správně tvořeného obsahu dokumentu. Na úvodním řádku souboru je uveden zápis verze XML a případně způsob kódování znaků (*windows-1250*, *UTF-8* a jiné) ve tvaru *<?xml version="1.0" encoding="UTF-8" ?>*. Kódování je důležité například pro přenos dokumentu z operačního systému MS Windows na distribuce operačního systému Linux. Psaní dokumentů je založeno na principu elementů, které jsou vyjádřeny počáteční, obsahovou a ukončovací částí např. *<jmeno>Novák</jmeno>*. Slovo *jmeno* znamená název elementu a text *Novák* je obsahem elementu, jeho hodnotou. Pojmenování elementů se řídí přesným pravidlem, které zakazuje přítomnost speciálních znaků (mezera, uvozovky, apostrof, středník a další speciální) a číslice v prvním znaku názvu. Pokud jsou používána písmena, číslice (ve druhém až posledním znaku názvu) nebo znak podtržítka, bude název elementu vždy korektní. Textovým obsahem elementu může být textová hodnota např. *Novák*. Pokud jako hodnotu uvádíme řetězec se speciálními nestandardními znaky je vhodné je uvozovat sekcí CDATA například takto: *<specialni> <![CDATA[text se speciálními < ; % ^ znaky]]></specialni >*. [8] [9]

Tvorba struktury celého XML dokumentu vychází ze skutečnosti, že obsahem elementů, kromě textové hodnoty nebo speciálního textu v sekci CDATA, mohou být další vnořené elementy. Jazyk XML definuje přesná pravidla tvorby této struktury, neboť vzájemné vnořování elementů musí být provedeno korektně. Struktura je uvozena kořenovým elementem, který se musí v každém dokumentu povinně vyskytovat – úvodní část <*element_dokumentu*>, koncová část </*element_dokumentu*>. Uvnitř bývá vnořen alespoň jeden další element nebo žádný element. V případě žádného výskytu elementu dalšího uvnitř kořenového se jedná o prázdný XML dokument. Uvnitř kořenového elementu lze umístit elementy s textovou hodnotou, elementy s textem uvozeným sekcí CDATA. Samozřejmostí je vnořování dalších elementů do obsahu jiného. Takový element místo hodnoty textové agreguje jiný element. Element nemůže současně obsahovat textovou hodnotu a zahrnovat přitom další elementy. Proto existují parametry XML, jejichž ukázkou zápisu je parametr *version* nebo *encoding* se svými hodnotami *"1.0"* a *"UTF-8"*. Pomocí vnořování elementů lze vytvořit složité struktury, jejichž příkladem může následný XML dokument pro uložení informací o fyzikálních veličinách (Obr. 18). [8][9][10]

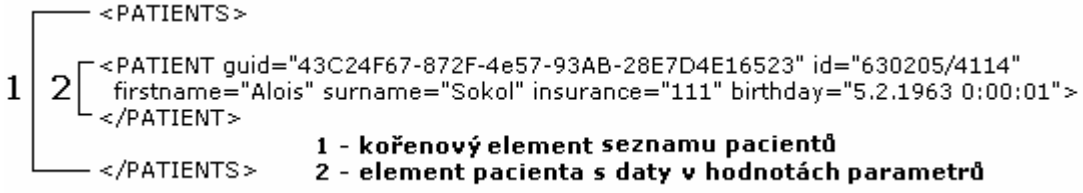


Obr. 18. Znázornění principu tvorby dokumentů XML na příkladu z oblasti fyziky

5.2 Analýza způsobu archivace dat systémem Fowler z pohledu XML

V kapitole 1.2 je analyzován způsob archivace dat vyšetření pacientů z pohledu obecného postupu archivace, i když určité techničtější detaily byly zmíněny, a to způsob obecného ukládání dat systémem Fowler do souborů s příponou *.xml*. Protože jsou vysvětleny principy XML až v kapitole 5.1, je uveden rozbor archivace dat systémem Fowler nyní již z pohledu jazyka XML.

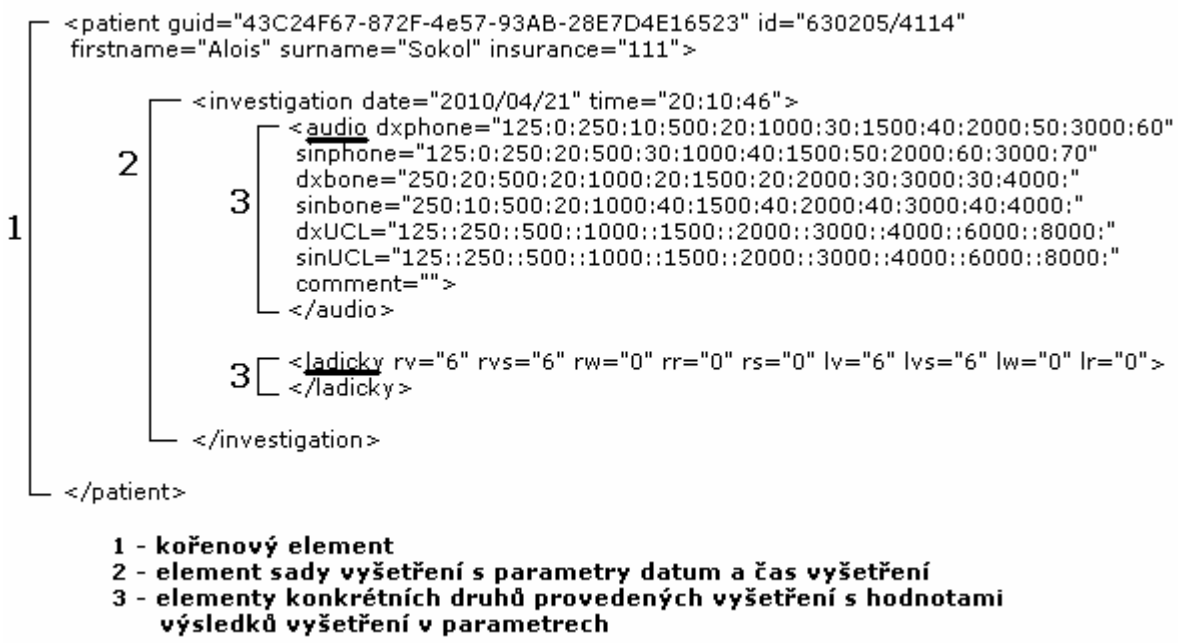
Formát ukládání dat pacientů a hodnot jejich provedených vyšetření se řídí pravidly jazyka XML. Program Fowler velmi přesně dodržuje standardizovaná doporučení pro ukládání dat pomocí technologie XML. Soubor se seznamem pacientů *patients.xml* zahrnuje klíčová data o pacientech zadaná lékařem a databázový klíčový identifikátor generovaný automaticky systémem Fowler. Dokument má kořenový adresář pojmenován *PATIENTS* a zahrnuje uvnitř pro každého pacienta jeden element s pojmenováním *PATIENT*. Každý element *PATIENT* má v podobě parametrů s textovými hodnotami uloženy všechny potřebné údaje o pacientovi. Parametry mají pojmenování *guid* (klíčová identifikátor pacienta v rámci databáze pacientů), *id* (rodné číslo nebo identifikátor v případě cizince), *firstname* (křestní jméno pacienta), *surname* (příjmení pacienta), *insurance* (číselný kód pojišťovny), *birthday* (datum narození pacienta). [15]



Obr. 19. Ukázka seznamu pacientů v systému Fowler s daty fiktivního pacienta [15]

Každý pacient má svá data vyšetření ukládána do souboru s názvem souvisejícím s jeho hodnotou parametru *guid* v souboru *patients.xml*. Pro pacienta z obrázku (Obr.19) je soubor s daty vyšetření pojmenován podle jeho *guid*, a to *43C24F67-872F-4e57-93AB-28E7D4E16523.xml*. Protože pro tvorbu praktické části bylo třeba znát ze souborů vyšetření pouze informace provozního charakteru (datum a čas vyšetření) a údaj o druhu provedeného typu ORL vyšetření, zaměřím se při popisu schématu pouze na tato fakta. [15]

Soubor XML dat vyšetření konkrétního pacienta má kořenový element s názvem *patient*. Uvnitř se nacházejí vnořené elementy sad vyšetření s jménem *investigation*. Element *investigation* má důležité dva parametry *date* (datum provedené sady vyšetření) a *time* (čas provedené sady vyšetření). Hovořím o sadě vyšetření, protože uvnitř elementu *investigation* se může nacházet více elementů, a to elementů reprezentujících již konkrétní provedená vyšetření. Záznam konkrétního vyšetření již nemá parametry časového charakteru, protože jsou uvedeny právě v nadřazeném elementu *investigation*. Element konkrétního typu vyšetření má přesně stanovené pojmenování, a to podle druhu ORL vyšetření. Jednotlivé výsledky jsou ukládány jakožto hodnoty parametrů těchto konkrétních elementů vyšetření. Názvy elementů jednotlivých druhů vyšetření jsou následující: *audio* (audiometrické vyšetření), *ladicky* (klasická zkouška sluchu a vyšetření ladičkami), *tympano* (tympanometrické vyšetření), *sisi* (SISI test). [15]



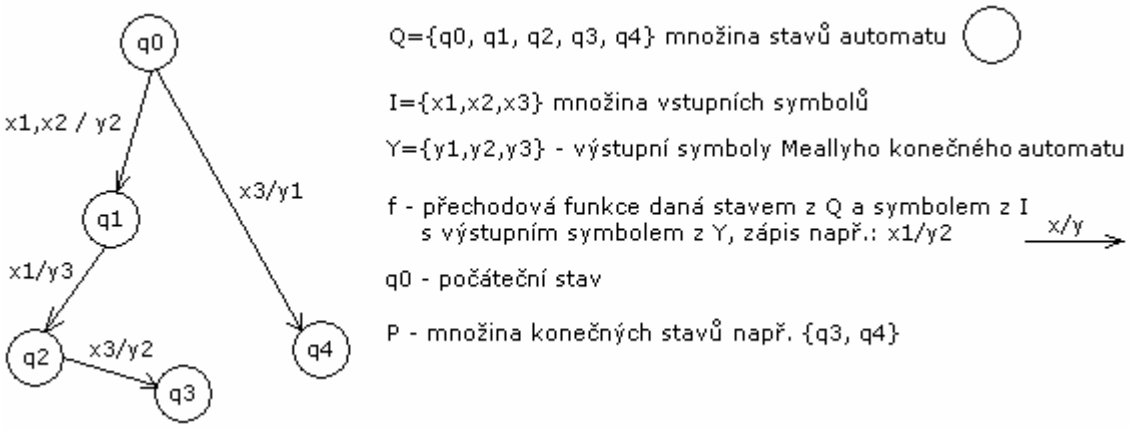
Obr. 20. Ukázka souboru výsledků vyšetření fiktivního pacienta v systému Fowler [15]

6 POSTUPY A TEORIE POUŽITÉ PŘI PROGRAMOVÁNÍ

6.1 Princip Meallyho konečného automatu

Praktická část se v části popisu použitých řešení při programování také opírá o teorii konečných automatů, a proto bude nyní vysvětlena se všemi náležitostmi. Svou podstatou a principem fungování je Meallyho konečný automat vhodně použitelný pro zpracování a analýzu znaků textového celku. Pro konečné automaty existuje řada modifikací. Meallyho automat je právě jednou z nich. Obecně je konečný automat považován za formální model výpočetního stroje. Množinou vstupů automatu je řada symbolů abecedy konečného rozsahu. Každý automat se může nacházet v jednom ze své množiny stavů. Podle hodnot konkrétního vstupního symbolu, který je automatem načten, se případně mění stav automatu z daného na jiný. Tomu odpovídají podmínky přechodů mezi stavy dle stanovených hodnot symbolů na vstupu automatu. Na počátku činnosti je automat v základním počátečním stavu a pokud má definovány stavy další a podmínky přechodů mezi nimi, může při zpracování vstupního symbolu vždy dle předepsaných podmínek činnosti cyklicky nabývat různých stavů. Stavy na sebe navazují právě jednotlivými přechody, takže je přechod z původního stavu na nový, v případě splnění podmínky přechodu učiněn deterministicky. Automat může přejít do dalšího stavu vyznačeného pouze přechodem. Nemůže obecně přecházet z libovolných stavů na jiné. Vše se řídí dle pravidel určených při návrhu. Protože konečný automat nemá klasickou paměť pro ukládání dat, pamětí jsou ve své podstatě samotné stavy, kterých nabývá a ve kterých setrvává po dobu nesplnění podmínek přechodu do některého ze sousedních stavů. Aby popis mechanismu konečného automatu byl kompletní, je nutné vysvětlit ještě výstupní funkce. Různé varianty automatu generují své výstupy svým specifickým způsobem. Výhodným použitím konečného automatu je Meallyho varianta. Meallyho konečný automat totiž generuje výstupy při přechodu mezi jednotlivými stavy a výstup závisí na právě vyhodnocovaném vstupním symbolu. Proto je použití Meallyho varianty natolik výhodné, protože se při návrhu automatu kromě podmínek přechodů mezi jednotlivými stavy dá dynamicky dodefinovat, jaké výstupní symboly bude při těchto přechodech generovat. Dalším typem konečného automatu je Moorův automat, který se od Meallyho liší ve skutečnosti, že jeho výstupy vyjadřují pouze stav automatu, ve kterém se nachází a nejsou ovlivněny právě vyhodnocením vstupního symbolu. Meallyho automat lze graficky znázornit stavovým diagramem a je možný také jeho matematický zápis. [7][11]

Matematicky lze vyjádřit navržený Meallyho automat jako uspořádanou *n-tici*, v tomto případě pětici $\{Q, I, f, q_0, P\}$. Množina Q znamená posloupnost stavů automatu$\{ q_0, \ldots, q_n\}$, ve kterých se může nacházet. Množina I zahrnuje posloupnost znaků vstupní abecedy. Vyjádření f znamená matematický kartézský součin množin Q a I. Vyjádření q_0 je základním stavem automatu. P je množina stavů konečného automatu, ve kterých pokud se automat při přečtení posledního znaku nachází, tak říkáme, že automat vstupní slovo přijal. Kromě matematického popisu a stavového diagramu existuje dále například vyjádření navrženého automatu pomocí tabulky nebo stavového stromu. Na obrázku (Obr.21) je uvedena ukázka stavového diagramu pro vyjádření návrhu Meallyho konečného automatu. Toto navržení lze vyjádřit také tabulkou (Tab.2) [7][11][12]



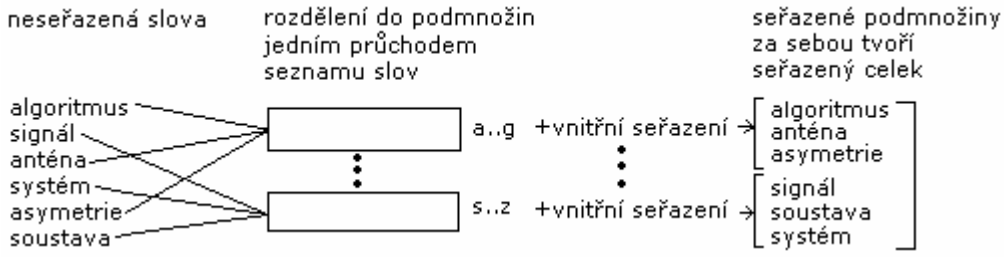
Obr. 21. Vyjádření návrhu Meallyho konečného automatu stavovým diagramem [11]

q - stav automatu, x - vstupní symbol, y - výstupní symbol (x/y) - přechod	**x1**	**x2**	**x3**
q0	(1/y2)	(1/y2)	(4/y1)
q1	(2/y3)	0	0
q2	0	0	(3/y2)
q3	0	0	0
q4	0	0	0

Tab. 2. Vyjádření návrhu Meallyho konečného automatu ve formě tabulky [11][12]

6.2 Postup řazení dat

Výklad třídicích postupů je zaměřen na v praktické části využitou nerekurzivní třídicí metodu Bubble-sort použitou jako dílčí metoda v rámci nadřazeného postupu třídění. Metoda spočívá v opakovaném průchodu množiny prvků do té doby, dokud nejsou všechny seřazeny. Řazení dvou po sobě následujících prvků spočívá v jejich porovnání a případné výměně. Protože se jedná o elementární metodu, je její použití vhodné při začlenění do dalších třídicích postupů a řešení. Příkladem je její použití pro metodu třídění spojováním. Třídění spojováním vychází z rozdělení množiny prvků, které je třeba setřídit, do více podmnožin. Na každou tuto menší podmnožinu se aplikuje právě elementární metoda třídění jako je například uvedený Bubble-sort. Protože je počet prvků v jednotlivých podmnožinách menší, než-li by byl u sjednoceného celku, bude časové provedení seřazení elementární metodou výpočetně méně náročné. Po seřazení prvků v rámci jednotlivých podmnožin elementární metodou, se podmnožiny systematicky sloučí v jeden celek, v němž budou již všechny prvky korektně seřazeny. V odborné literatuře je uvedeno, že při metodě třídění spojováním se množina všech prvků dělí na dvě podmnožiny. To je důležité pro číselné porovnávání. Vhodnou modifikací je pro případ abecedního řazení textových hodnot rozdělení do více podmnožin např. dle charakteristického prvního písmene slova. Nemusí být provedeno rozdělení na tolik podmnožin, kolik je písmen abecedy, ale může být pro podmnožinu charakteristická skupina určitých písmen např. *a* až *f*. V těchto dílčích podmnožinách se provede řazení elementární metodou. Pro uvedený případ je vhodné použít pro jednotlivé podmnožiny nový paměťový prostor, do kterého jsou prvky nejprve rozřazeny podle příslušnosti jednotlivých podmnožin. Po seřazení se paměťové prostory spojí v jeden celek, který tvoří již abecedně seřazené textové hodnoty. Pro řazení číselných hodnot je výhodné použít rychlejší rekurzivní algoritmy, ale pro řazení textových hodnot je postačující uvedené řešení. [13][14]



Obr. 22. Schéma postupu řazení textových dat metodou třídění spojováním

II. PRAKTICKÁ ČÁST

7 ZADÁNÍ SYSTÉMU SE SPECIFIKACÍ POŽADAVKŮ

Praktická část diplomové práce pojednává o postupu tvorby zadaného systému, který je dále implementován, a to do podoby softwarové aplikace. Při její tvorbě jsou dodržovány fáze procesu vývoje systémů, konkrétně vodopádového modelu. Před samotným popisem návrhu, implementace a funkcí hotového řešení, je nejprve uvedena množina požadavků a specifikací, podle kterých se samotná tvorba programu odvíjí. Fáze určení požadavků je stavebním základem pro další fáze tvorby programu.

Obecně formulovatelným zadáním praktické části je vytvoření systému pro vyhodnocení a vizualizaci časových souhrnů počtu provedených vyšetření v ORL lékařství. Konkrétní implementace systému má vycházet ze způsobu archivace dat vyšetření ORL systémem Fowler [15], řídit se terminologií ORL lékařství a má být otevřeným systémem pro případné budoucí modifikace. Výsledky navrženého systému mají být prověřeny z pohledu jejich správnosti. Program je nutné otestovat na množině testovacích a reálných dat, aby byla jeho funkčnost prověřena před jeho nasazením do praxe. Počty vyšetření budou vyhodnocovány za stanovené časové období a z pohledu různých diagnóz vyšetření. Implementace aplikace bude založena na práci se soubory typu XML. Systémovým požadavkem na program je fungování na systému minimálně verze MS Windows XP a vyšší. Systém může sloužit jako doplněk k programu Fowler [15], ale díky principu otevřenosti jej bude možno v budoucnu pro jiné účely programově přizpůsobit. Funkce programu budou popsány v nápovědě programu.

Přesnějšími specifikacemi na systém jsou druhy často používaných statistických souhrnů. Kromě uvedeného vyhodnocení počtu vyšetření v rámci stanoveného časového období (roky, měsíce, dny) a pro určené typy vyšetření, se jedná o počet pacientů a vyšetření pro vybrané pojišťovny, statistiky pouze pro vybrané pacienty. Systém by měl být otevřený v případě vyobrazení dostupných typů vyšetření s ohledem na to, že mohou přibýt v budoucnu typy nové. Volba řešení analýzy dat a jejich interpretace ve formě abecedně řazených výsledků by měla být co nejméně časově náročná. Statistické výsledky programu bude možno exportovat do formátu souborů, které dokáží být dále tisknutelné nebo importovatelné jinými aplikacemi určenými pro jejich případnou další analýzu.

8 NÁVRH SYSTÉMU PRO VYHODNOCENÍ A VIZUALIZACI ČASOVÝCH SOUHRNŮ PROVEDENÝCH VYŠETŘENÍ

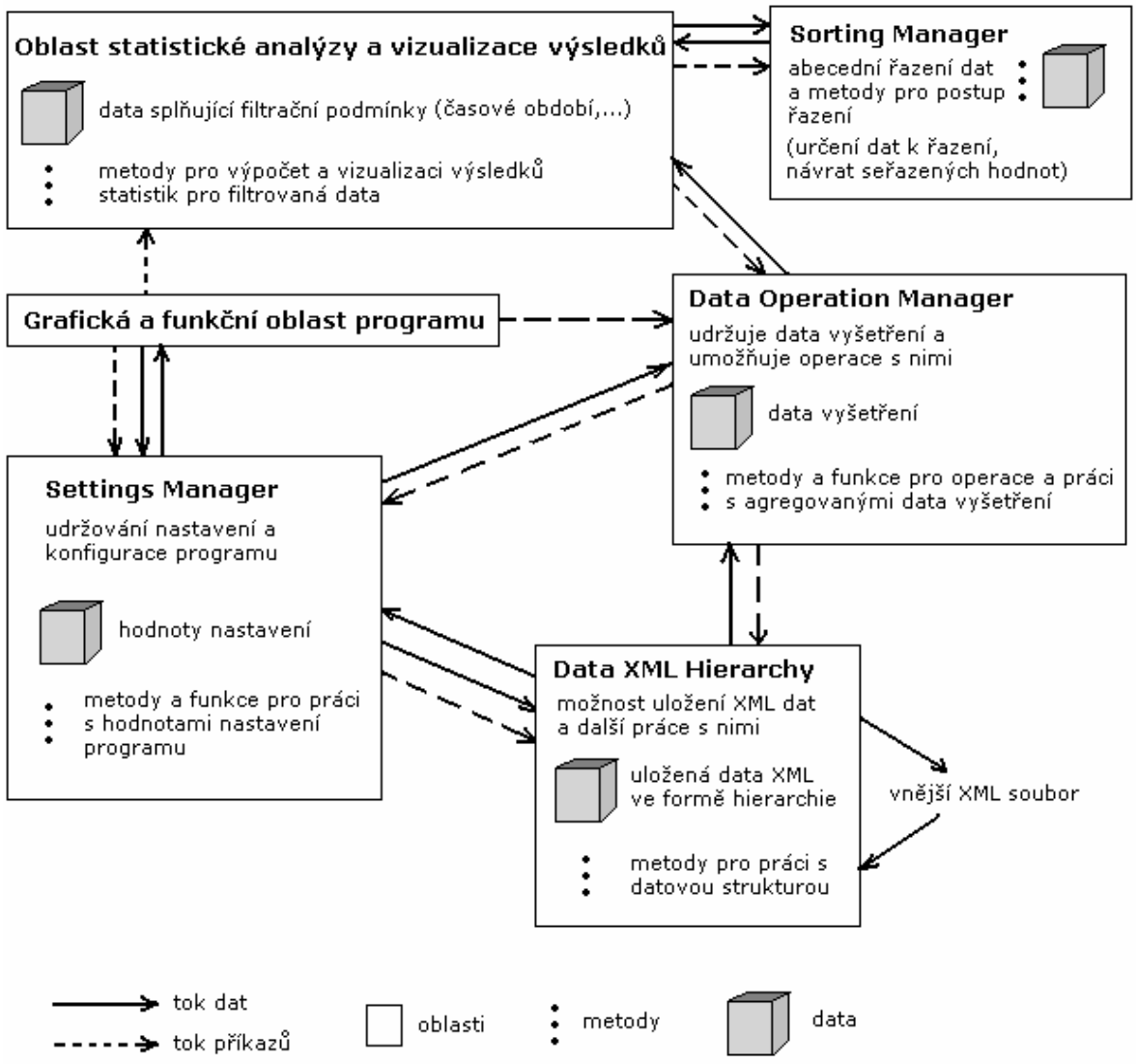
Z fáze provedení specifikací obecných i konkrétnějších požadavků nyní přejdu k fázi návrhu systému. Budu vycházet ze skutečností, které se týkají logické podstaty řešení, ale také z technického hlediska. Souhrnně návrh tvoří představu o podobě a plánu tvorby výsledného systému, který splňuje všechny definované vstupní požadavky. Z fáze návrhu vychází pak samotná implementace řešení.

Podle požadavků zadání a doporučené strategie směrování postupu tvorby programu jsem do výběru programovacích prostředků zařadil nástroje a technologie, se kterými jsem již při studiu na vysoké škole konkrétně pracoval a mám s nimi zkušenosti. Významným důvodem výběru se stal také ekonomický aspekt a skutečnost, že program nevyžaduje síť Internet a nevyžaduje speciální jiné komerční nástroje – je jednoduše spustitelný bez jakýchkoliv přídavných systémových součástí. Všechny programovací prostředky jsem také vybíral po stránce jejich bezplatné dostupnosti a možnosti jejich dalšího použití. Protože má být dle zadání systém otevřený, vyřadil jsem možnost programovat systém pomocí nástrojů určených pro jeden operační systém a umožnil jsem svým výběrem konkrétní programovací techniky snadný překlad programu pro ostatní operační systémy (např. pro distribuce Linuxu, MacOS). Zvolenými prostředky jsou multiplatformní programovací knihovna okenních aplikací wxWidgets, překladač MinGW, editor kódu CodeBlocks s integrací překladače MinGW, návrhový program grafického designu aplikace wxFormBuilder. Zvoleným jazykem pro psaní zdrojových kódů je C++. Nástroje a jejich příslušné jazyky, které nebyly bezplatně dostupné (Delphi, Visual Basic, VS C#), nebyly multiplatformní, potřebovaly dodatečnou instalaci běhových prostředí (C# .NET, Java), nebo vyžadovaly Internet (PHP, ASP, JSP), jsem všechny z výběru vhodných prostředků vyřadil. Vybrané prostředky: knihovna wxWidgets a nástroje CodeBlocks, MinGW překladač, wxFormBuilder a jazyk C++ splnily všechny specifikované požadavky na zadání a také byly vhodné z ekonomického pohledu na volbu řešení a předpokladu splnění podmínek pro budoucí běh programu. Vytvořený program bude potřebovat pouze minimální požadavky pro svůj běh (OS MS Windows XP), nebude potřebovat ke své činnosti žádné jiné komerční prostředky a bude spustitelný na počítačových stanicích nepotřebujících připojení k síti Internet.

Aby bylo možné výsledky, které systém vyhodnotí, uživatelsky dále analyzovat a zpracovávat, navrhl jsem způsob podpory, a to v podobě exportu výsledků do CSV a XML textových souborů. Jedná se o technické řešení, které je použitelné v případě, pokud chceme data dále importovat některým z komerčních kancelářských tabulkových nebo databázových programů. Pro snadný tisk a práci s výsledky jsem zvolil princip existence textových forem výsledků a export do zobrazitelných souborů – webová stránka s textem nebo tabulkou, popřípadě samotný textový soubor. Exporty výstupů programu zde zmiňuji jako další způsob ekonomicky výhodného řešení, protože se nepředpokládá standardní export např. přímo do MS Excelu a jeho existence na disku, ale je zvolena prostřední cesta ve formě exportovaných souborů dále možnými k otevření komerčními programy. Exportní návrh řešení umožňuje univerzální export do jakýchkoliv dalších analytických programů, které dokáží pracovat s CSV a XML technologiemi. V případě nepodporování importu XML a CSV souborů jinými aplikacemi lze s daty pracovat ve variantě jednoduchého textu nebo webové stránky. Návrh principu exportů řeší otázku otevřenosti systému do budoucna po stránce práci s výsledky programu.

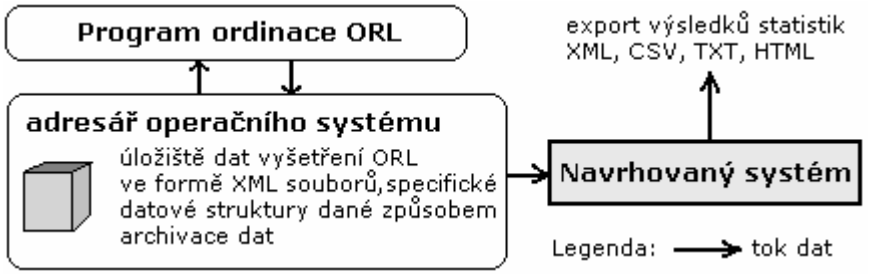
8.1 Návrh architektury systému

Nejdůležitější částí fáze návrhu je určení architektury. Aplikaci je nutné rozčlenit do funkčních celků a oblastí, které se starají již o přesné úkoly. Oblasti navzájem spolupracují nebo se strukturálně překrývají, mohou být i libovolně vnořovány, ale mají společnou vlastnost, že jsou podmnožinami celé aplikace. V programovém kódu poté tyto menší celky označuji a pojmenovávám podle jejich úlohy jako manažery nebo jako ostatní celky starající se nebo zahrnující vizualizační logiku, fungování nebo správu dat. Na následujícím obrázku uvádím návrh vnitřní architektury navrhovaného systému. Systém člením na manažer správy nastavení (Settings Manager), manažer správy a operací nad daty vyšetření (Data Operation Manager), oblast pro práci s daty formátu XML (Data XML Hierarchy), manažer abecedního řazení dat (Sorting Manager). Ostatními součástmi jsou oblasti starající se o grafickou a funkční podobu aplikace, oblast statistické analýzy a vizualizace výsledků. Oblasti, které jsem označil pojmem manažer, jsem v návrhu systému vymezil jako objekty starající se funkčně o své agregované datové úložiště. Manažer daného druhu nabízí jiným oblastem programu přes své rozhraní možnost uložit mu příkaz nebo požadovat od něj hodnotu nebo hodnoty, a to jako výsledky jeho určité specifické činnosti.



Obr. 23. Schéma architektury vnitřní části navrženého systému

Návrh architektury pro začlenění navrhovaného systému do prostoru vnějšího jsem koncipoval podle pravidel způsobu archivace dat vyšetření z kapitoly teoretické části práce 5.2. Na obrázku (Obr.24) uvádím schéma interakce systému s vnějším okolím.



Obr. 24. Schéma začlenění navrhovaného systému do vnějšího okolí

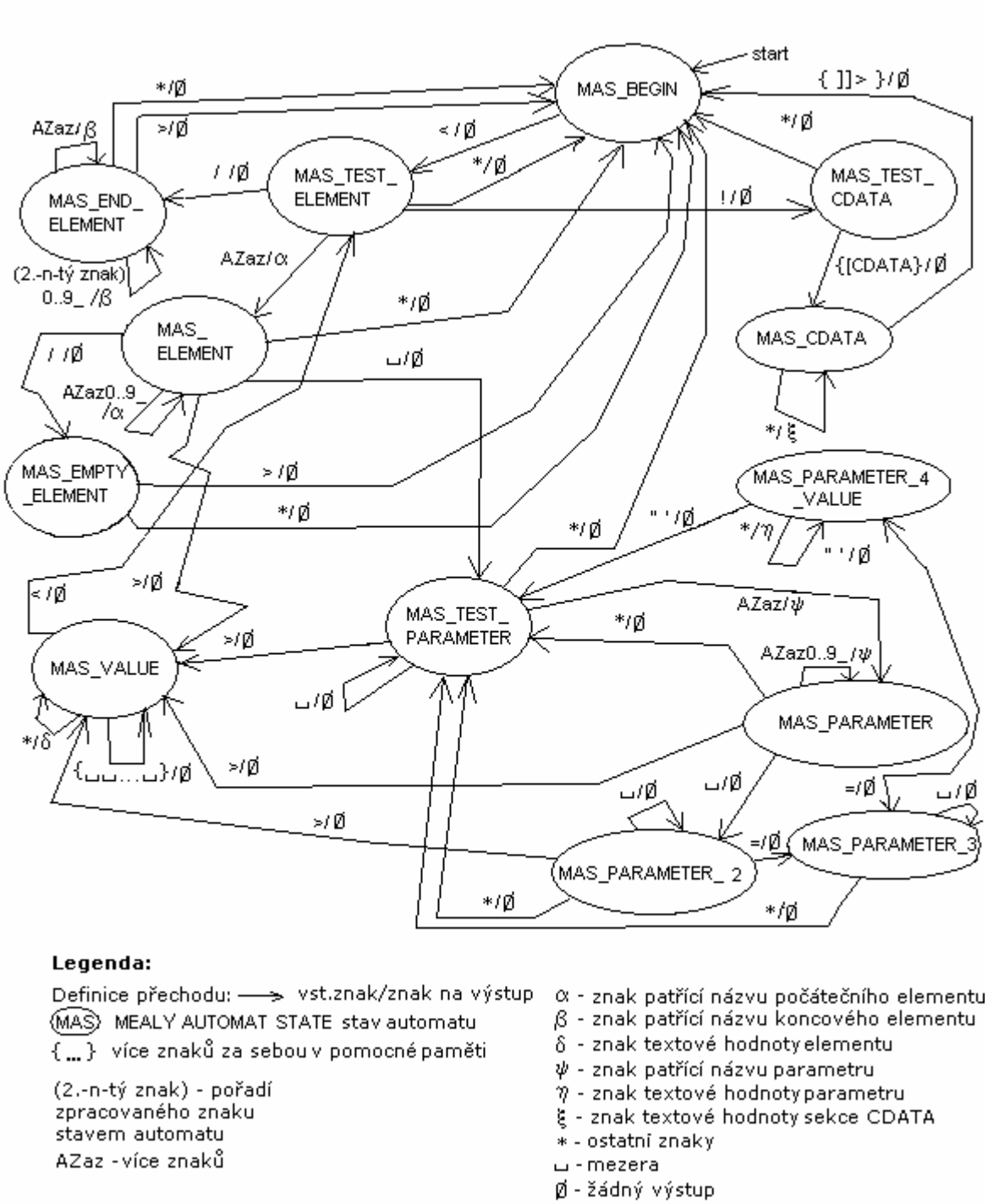
8.2 Principy jednotlivých částí architektury systému

Nyní se více podrobněji zaměřím na popis jednotlivých částí architektury navrhovaného systému. Tok příkazů a dat jsem znázornil na obrázku (Obr.23). Vyznačené návrhy toků jsem volil z pohledu funkčního, a to dílčích částí architektury. Program je reprezentován oblastí *grafická a funkční oblast programu*. Oblast zahrnuje logiku aplikace a formu komunikace s uživatelem v podobě dialogů nastavení. Nastavení a konfigurace programu je za běhu programu udržována v datové struktuře správce *Settings Manager*. Jakákoliv další oblast může tyto hodnoty používat pro svůj chod nebo je případně modifikovat. Práci se souborem XML, v případě načítání i ukládání, zajišťuje oblast *Data XML Hierarchy* uchovávající data XML v interní datové struktuře. Oblast *Data XML Hierarchy* kromě správce *Settings Manager* využívá, ale jen pouze pro načítání dat, oblast *Data Operation Manager*. Tento manažer má za úkol načítat data vyšetření do struktur oblasti *Data XML Hierarchy* a dále je uchovávat pro další potřeby programu. Příkazy k načtení dat do *Data Operation Manageru* dává samotný program v případě, kdy je potřeba data načíst. Manažer komunikuje se správcem nastavení pro případ, že chce získat nastavenou cestu k adresáři s daty, které má načíst. Programová oblast zajišťuje dle požadavků uživatele spuštění činnosti oblasti *Oblast statistické analýzy a vizualizace výsledků.* Zde se nejprve všechna data za pomoci *Data Operation Manageru* přefiltrují podle uživatelem daných podmínek (např. časové období, jen vybrané typy vyšetření) a uloží do oblasti statistické analýzy pro statistické vyhodnocení a vizualizaci jeho výsledků v různých formách. Oblast statistické analýzy dále bude poskytovat možnost výsledky exportovat. Tato oblast úzce spolupracuje se správcem abecedního řazení dat *Sorting Manager*, který data abecedně nebo číselně setřídí a vydá seřazené zpět k dalšímu případnému vyhodnocení nebo vizualizaci.

8.3 Návrh řešení načítání XML dat oblastí Data Xml Hierarchy

V teoretické části práce v kapitole 6.1 je uvedena teorie Meallyho konečného automatu, kterou je výhodné použít pro proces zpracování znaků textového souboru. Na této teorii je založen princip oblasti *Data XML Hierarchy*, a to přesněji funkcionalita načítání dat z XML souborů do její interní datové struktury. Data souborů XML by bylo možné načítat pomocí jednořádkových příkazů knihovny wxWidgets, ale z důvodu, že se diplomová práce týká práce s XML technologií, jsem vypracoval vlastní návrh řešení načítání XML dat, a to na základě v teorii práce uvedeném principu zpracování textových dat Meallyho

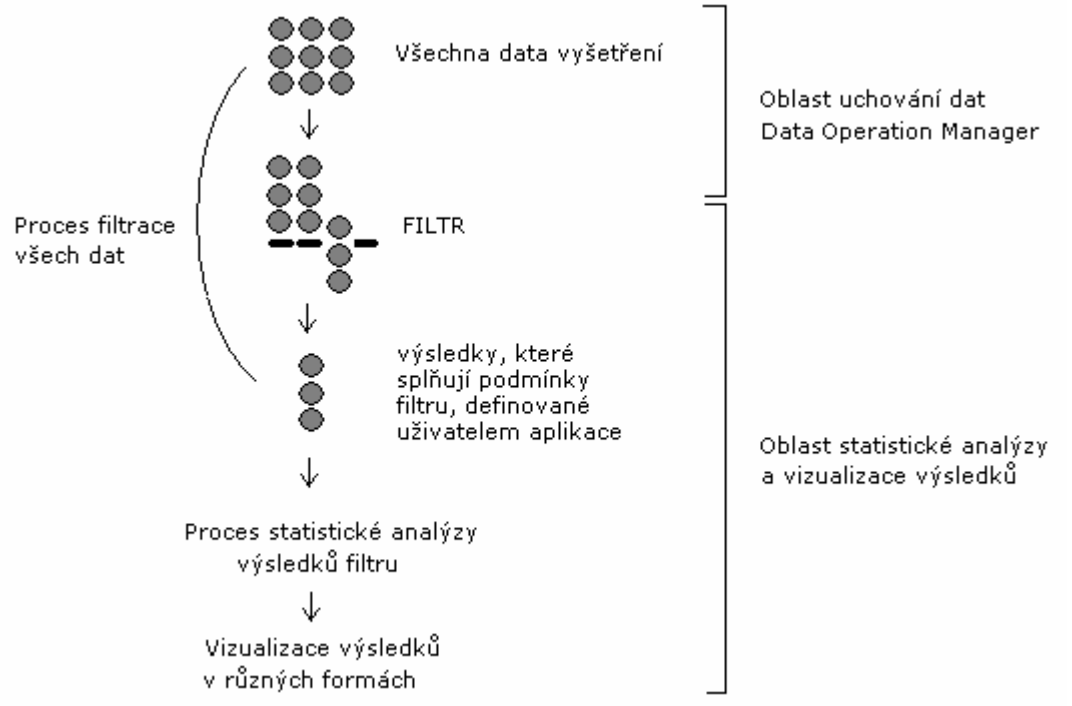
konečným automatem. Na obrázku (Obr.25) uvádím vlastní řešení načítání dat XML ze souborů. Zpracování funguje načítáním znak po znaku a jejich analýzou a vyhodnocením Načítání zahrnuje identifikaci symbolů elementů, jejich parametrů, hodnot, CDATA sekcí.



Obr. 25. Návrhu řešení načítání XML dokumentů v oblasti Data XML Hierarchy

8.4 Návrh řešení oblasti statistické analýzy dat

Statistická analýza dat probíhá v navrženém systému podle zásadního postupu. Data se po načtení ihned neanalyzují. Je nutno počítat s omezujícími filtračními podmínkami, které uživatel programu stanoví. Filtrem projdou pouze ta data, která budou odpovídat přesně definovaným filtračním podmínkám. Filtr bude univerzální, dynamický a pro průchod dat musí být splněny všechny podmínky současně. Pro případ, že by náhodou uživatel některou podmínku filtrace nedefinoval, umožní mu navrhovaná část systému zadat zobecnění ve formě „všichni pacienti“, „všechna období“ a další. Podmínky filtrace vychází striktně ze zjištěného způsobu archivace systémem ordinací ORL z kapitoly práce 5.2. Na obrázku (Obr.26) uvádím pevně stanovený postup filtrace, který je jádrem principu fungování oblasti systému, určené pro statistickou analýzu dat. Až po provedení filtrace jsou statistiky počítány.



Obr. 26. Obecně navržený postup filtrace dat dle podmínek uživatele

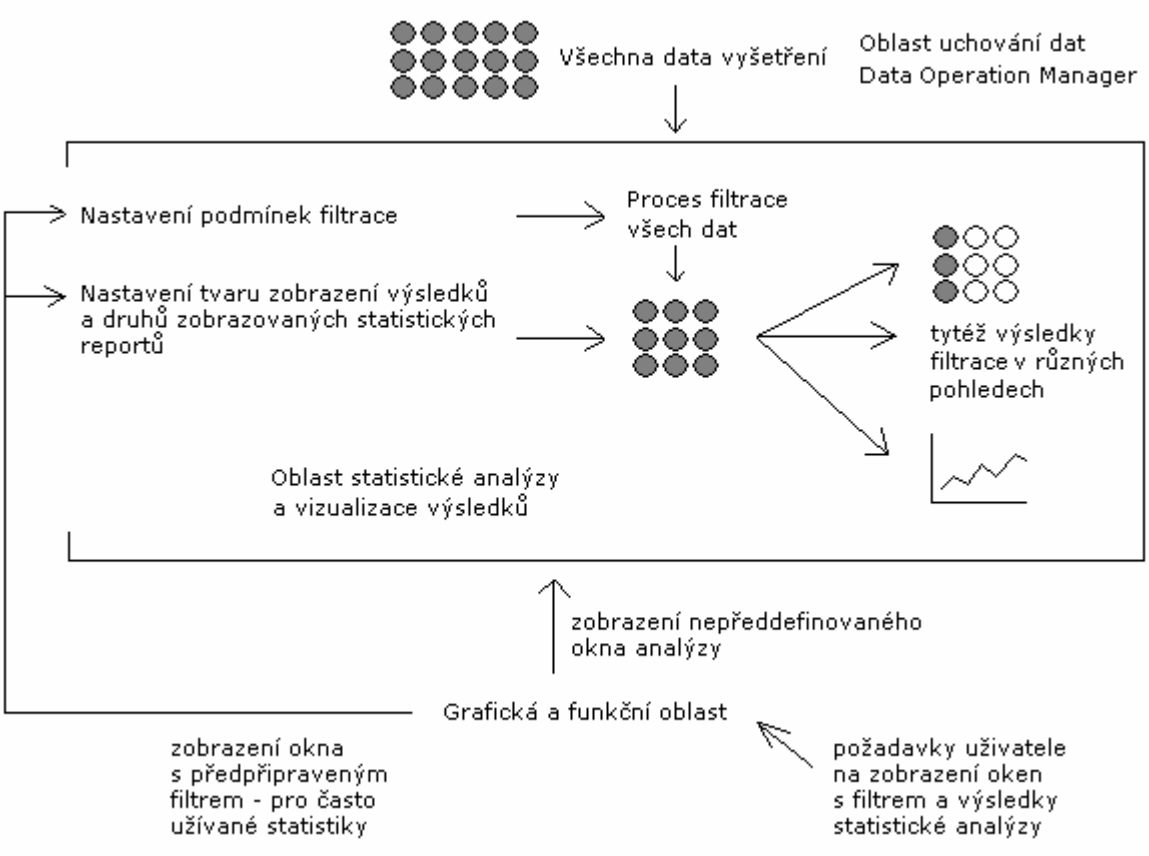
Filtr navrženého systému musí uživateli poskytovat univerzální nástroj, kde lze nadefinovat jakkoliv náročné a komplexní omezení průchodu dat. Dle rozboru způsobu archivace dat vychází ihned veškeré varianty možné filtrace. Podmínky musí být možno navzájem zkombinovat do formy složitého nebo jednoduchého tvaru. Z podrobného

rozboru v kapitole 5.2 vyplývá, že existuje seznam pacientů a přehled jejich vyšetření bude nutno filtrovat tak, aby byly obsaženy co nejvíce všechny možné varianty a kombinace omezení při pohledu na data. Ze samotného názvu diplomové práce vychází první kritérium filtrace dat, kterým je definice časového období provedených vyšetření. Filtr musí zahrnovat možnost ponechat nulové omezení - *celé období* a dále připustit varianty konkrétního zadání datumu. Navržená varianta počítá i s omezením ve tvaru zadání *od-do, do data, od data*. Musí být zajištěna kontrola zadaných datumů z pohledu jejich tvaru a časové následnosti. Druhou podmínkou filtru je navržena definice výběru konkrétních typů provedených vyšetření pro zúžení výběru dat pouze pro konkrétní druh metody stanovení diagnózy. Třetí podmínkou je možnost výběru konkrétního okruhu pacientů z abecedně řazeného seznamu, dle jmen, rodných čísel nebo označení kompletní množiny pacientů. Předposledním a posledním kritériem filtrace je výběr konkrétních pojišťoven ze seznamu a určení intervalu ročníků narození pacientů. Všechna uvedená filtrační kritéria budou aplikována na data a projdou pouze ta, která budou podmínkám filtru plně odpovídat. Pro šest navržených omezení v rámci filtru lze vytvořit natolik kombinací filtračních podmínek, že si v nich každý uživatel najde své specifické zadání konkrétní filtrace dat, která se mají dále statisticky vyhodnotit. Pro podmínky, kterými neprojdou žádná data bude v systému vyobrazeno upozornění na počet nula výsledků filtrace. Dále je navrženo zajištění aktualizace hodnot podmínek v případě přidání nového typu vyšetření v budoucnu, přidání nových dat vyšetření nebo nového pacienta.

8.5 Návrh způsobu vizualizace statistických výsledků

Statisticky budou vyhodnocena data, která projdou filtrem z různých hledisek a aspektů, tak aby byla splněna očekávání uživatelů. Samozřejmostí je výpis samotných filtrovaných dat, které splnily všechny podmínky filtru a vyhodnocené statistiky pro data. Statistiky zohledňují časový průběh dat a jejich náležitost ke stanovenému filtračnímu kritériu omezení časového intervalu. Vyhodnocují se počty všech součástí evidence (počty pacientů pojišťoven, počet vyšetření pacientů, počet vyšetření pojišťoven). Jsou vykombinována všechna možná stanovení počtů pro jednotlivé entity záznamů vyšetření ORL. Výsledky filtrace budou v systému zobrazeny v tabulkové i textové formě, bude možnost jejich vyobrazení ve smyslu volby prvního sloupce, zobrazovaných kategorií dat a strukturálně vnořené abecední řazení. Výsledky vyhodnocení počtů budou vyobrazeny formou grafů s histogramy četnosti , časovými histogramy, tabulkou a textovou sestavou.

S exportem výsledků se v navrženém systému od počátku návrhu počítá. Návrh oblasti statistické analýzy a vizualizace výsledků počítá s variabilitou úpravy zobrazení tvaru výsledků z pohledu uživatele, a to ve specifickém uživatelském nastavení, které poskytne konkrétní pohled na statistické výsledky filtrovaných dat. Grafická a funkční oblast návrhu systému by měla také zajistit umožnění zobrazení více oken a pohledů se statistikami a zobrazovat již předefinované často používané souhrny.

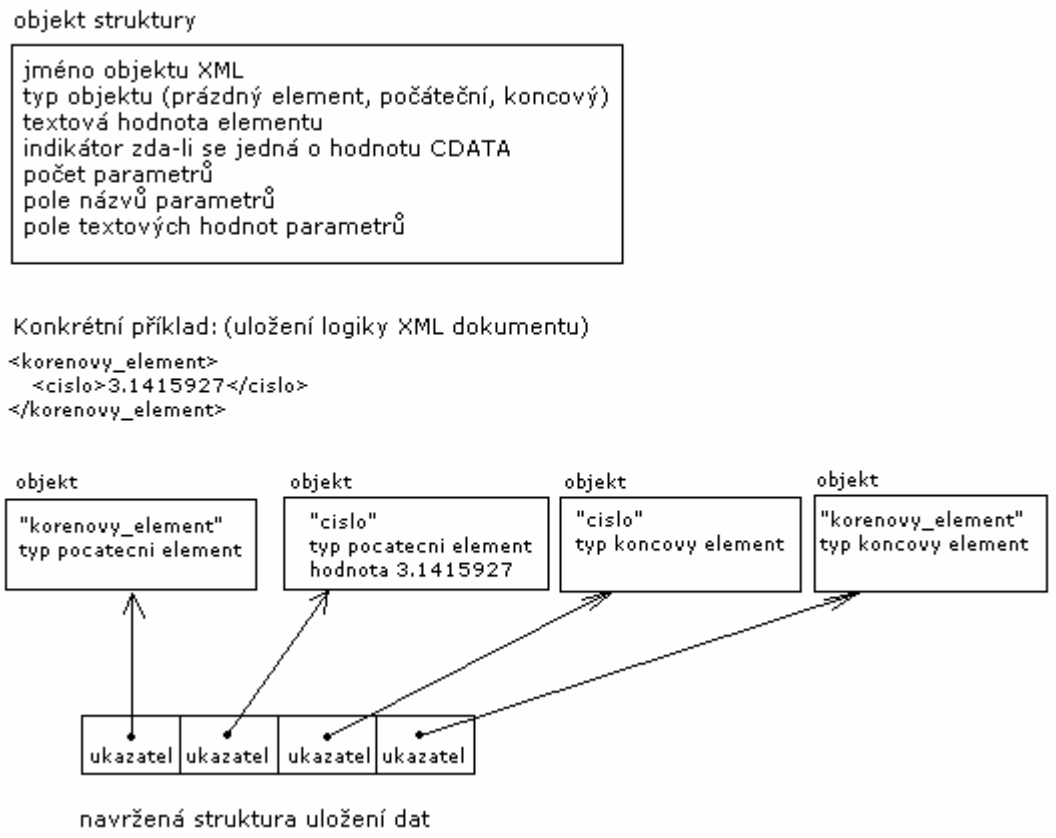


Obr. 27. Schéma principu statistického vyhodnocení dat a vizualizace výsledků

8.6 Ukládání dat v rámci jednotlivých částí architektury systému

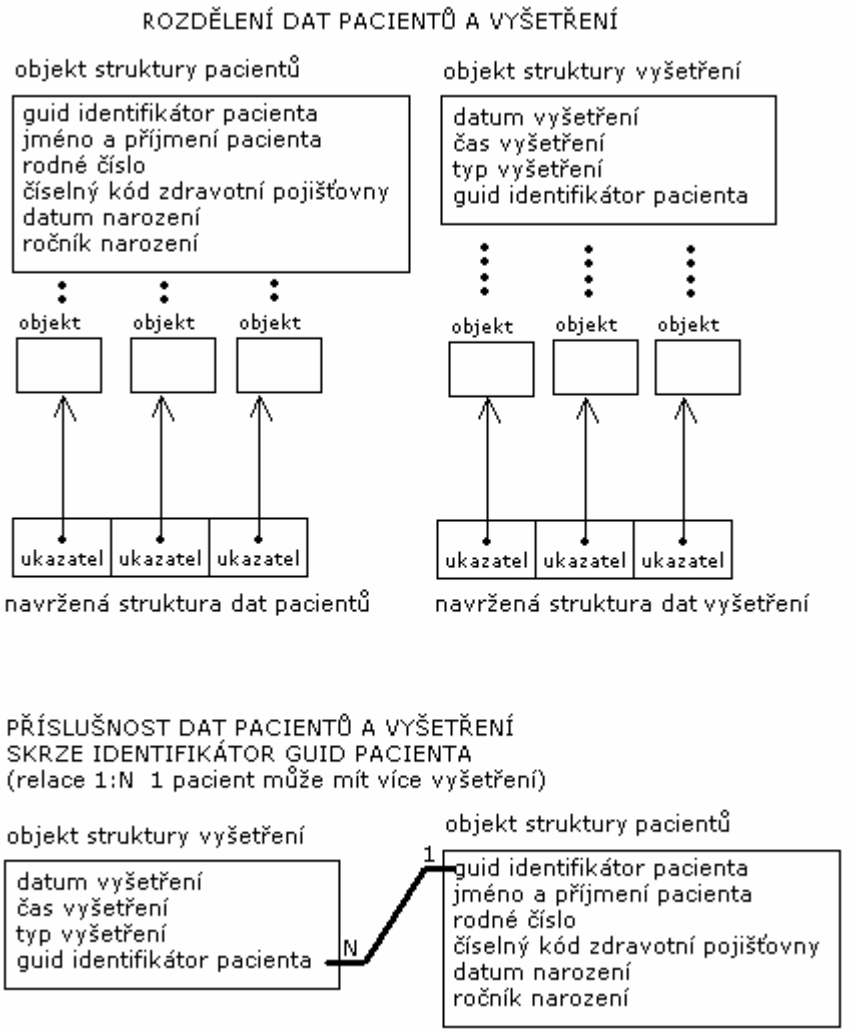
V návrhu architektury systému je pro jednotlivé části uvedeno, že zahrnují ukládání dat a údajů pro účely interní i celého systému. Návrhy struktur pro uložení dat se opírají o rozbor archivace vyšetření ORL z kapitoly 5.2, ale jedná se také o interní ukládání informací pro potřeby fungování programu. Návrh bude realizován nevizuálními prvky ukládání dat uvedenými v kapitole 4.3, kterými jsou obecné dynamické kolekce objektů.

Čistě obecného charakteru je struktura dat navržené oblasti Data XML Hierarchy. Ta slouží pro uchování údajů o XML dokumentu a jeho částech. Struktura dat musí být schopna uchovávat jak hodnoty, tak i identifikaci částí XML dokumentů a informaci o jejich vzájemném vnořování. Struktura je dynamickou kolekcí objektů, které reprezentují samotné části logiky XML dokumentu. Každý objekt pak udržuje informaci o svém typu XML objektu (počáteční element, koncový element, prázdný element) a také případně svou textovou hodnotu - klasickou nebo definovanou v sekci CDATA. Součástí informací v objektech jsou také pole parametrů s jejich jmény a textovými hodnotami. Struktura je naplněna při načtení XML souboru pomocí popsaného navrženého způsobu použití Meallyho konečného automatu. Na obrázku (Obr.28) je návrh struktury uložení dat oblasti Data XML Hierarchy. S údaji ve struktuře oblast dále nakládá a pracuje. Např. umožní takto uložená data nabídnout ke zpracování jiným oblastem návrhu.



Obr. 28. Návrh datové struktury oblasti Data XML Hierarchy pro práci s XML daty

Podle způsobu archivace dat se již řídím při návrhu struktury pro uchování dat vyšetření. Tato struktura uložení dat patří oblasti Data Operation Manager. Data jsou ve struktuře uchovávána pro jejich následnou filtraci. Data nejsou obsahově měněna, pouze dále čtena.

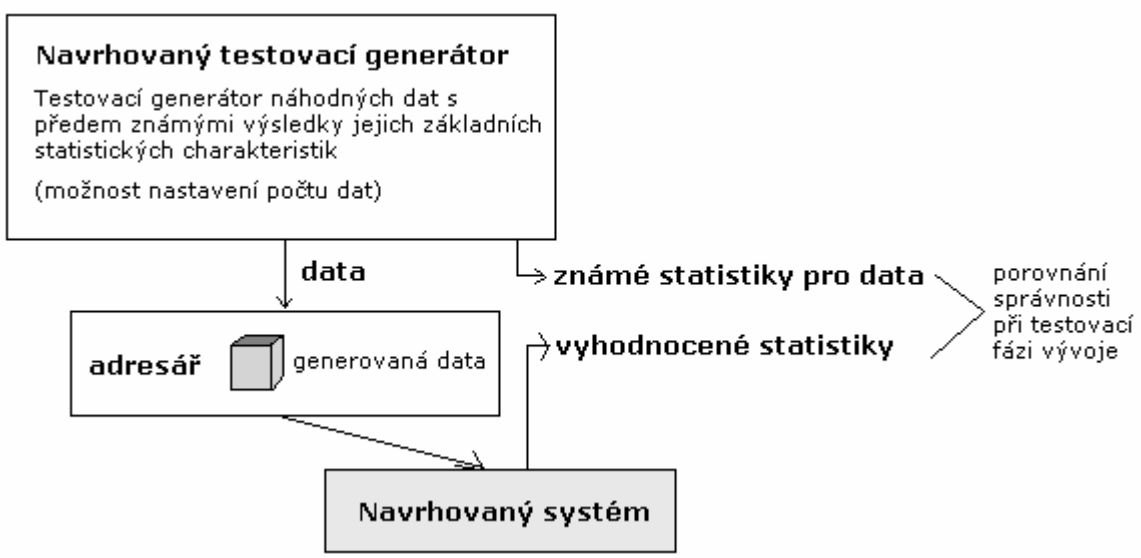


Obr. 29. Návrh datové struktury oblasti návrhu Data Operation Manager

Z důvodu, že statistických pohledů na filtrovaná data může být nezávisle mnoho, dle potřeb uživatele, neměním obsahovou strukturu Data Operation Manageru při operaci filtrace, ale data použiji pro tvorbu nové datové struktury, která bude zahrnovat již přefiltrovaná data. Každá samostatná oblast statistické analýzy a vizualizace výsledků pak bude mít svou strukturu filtrovaných dat a původní množina dat z Data Operation Manageru bude nezměněna a k dispozici pro další možné operace filtrace. Oblast statistické analýzy má tři kolekce dat – filtrovaní pacienti, filtrovaný vyšetření a třetí kolekcí je sjednocení záznamů obou kolekcí do jediné, a to skrze hodnotu *identifikátor guid pacienta*. Počet záznamů třetí kolekce odpovídá proto počtu záznamů první násobeného počtem záznamů druhé filtrované kolekce. Toto jednoznačné přiřazení záznamů pacientů a jeho vyšetření je dále používáno pro statistické vyhodnocení.

8.7 Návrh testovací aplikace pro ověření správnosti systému

Protože musí být správnost výsledků a činnost navrženého systému potvrzena, navrhl jsem přídavný testovací program, který bude generovat náhodná data pacientů a vyšetření. Testovací generátor dat náhodně vytvoří sadu dat, které dodržují způsob archivace ORL vyšetření a bude pomocí něho předem známo jak mají základní statistiky správně vyjít. Pro malý objem testovaných dat se účel testovacího programu na první pohled neprojeví. Výhoda testovacího programu je patrná až při velkém objemu dat např. 1000 pacientů a 2000 vyšetření. Pak je pro kontrolu práce navrženého systému v uvedeném případě výhodné generátor použít. Testováním se předejde nesprávnostem ve výpočtech statistik a lze je včas napravit, a to ještě v době, než-li bude program nasazen do praxe.



Obr. 30. Začlenění navrhovaného testovacího generátoru do procesu testování

9 REALIZACE NAVRŽENÉHO SYSTÉMU IMPLEMENTACÍ SOFTWAROVÉ APLIKACE

Ve fázi realizace se vychází z etapy definice požadavků a z fáze návrhu a implementuje se vše stanovené do tvaru výsledného softwarového produktu. Realizace zahrnuje naprogramování navrženého systému, architektury, funkcí a všech potřebných součástí. Všechny uvedené principy z předchozí kapitoly jsou nyní formovány do podoby spustitelného programu. Nyní se zaměřím na popis tvorby navrhovaného systému z pohledu programování. Přehled a konkrétní vysvětlení všech funkcí výsledného hotového řešení uvádím podrobněji v kapitole desáté.

9.1 Programování navrženého systému

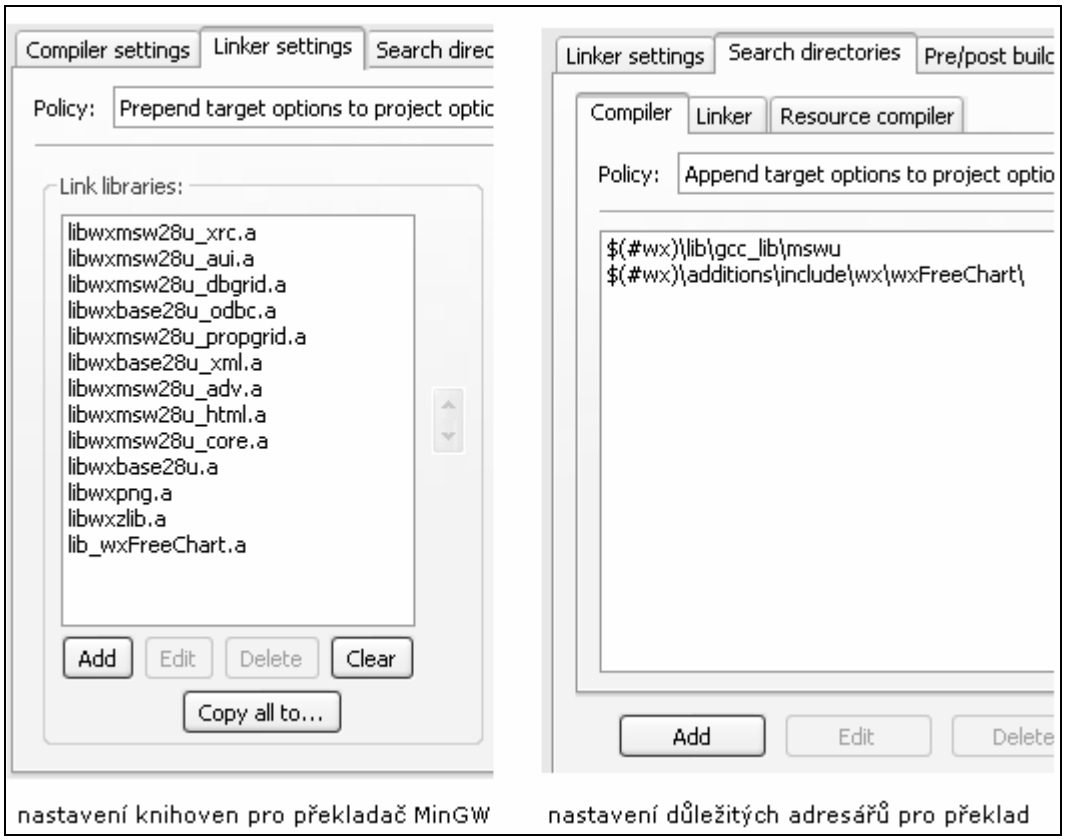
Pro programování výsledného systému jsem použil knihovnu tvorby okenních aplikací wxWidgets, editor kódu CodeBlocks, editor návrhu grafického designu formulářů wxFormBuilder a překladač MinGW. Nástroje jsem nainstaloval a nakonfiguroval dle postupu uvedeného v teoretické části práce. V editoru jsem vytvořil nový projekt a patřičně provedl jeho nastavení. Na následném přehledu uvádím již finální podobu všech součástí projektu programu a přesné nastavení projektové konfigurace. Doplňuji dále údaje o instalaci a začlenění části knihovny - ovládacího prvku wxFreeChart do projektu.

Protože se nastavení konfigurace projektu programu opírá také o existenci již nainstalovaného ovládacího prvku wxFreeChart, dostupného ze zdroje [18], uvedu nejprve postup instalace, kterou jsem také předpřipravil na přiložené CD přílohu P I. V adresáři přílohy P I - *zdrojove_kody /programovaci_nastroje /* se nachází archivační soubor *doplnkyWxWidgets.zip*. Po jeho dekomprimaci se objeví složky *wxFreeChart* a soubor *lib_wxFreeChart.a*. Adresář je třeba nakopírovat do umístění *C:\SourceCode \Libraries\wxWidgets2.8\additions\include\wx* (nebo případně do umístění Vaši instalace wxWidgets). Soubor *lib_wxFreeChart.a* vložit do umístění *C:\SourceCode\Libraries\ wxWidgets2.8\additions\lib\gcc_lib* (nebo případně z pohledu umístění Vaši instalace wxWidgets). Tím je ovládací prvek, který není běžnou součástí knihovny nainstalován a připraven k použití v rámci projektu programu. Další používané ovládací prvky knihovny wxWidgets jsou již běžnou součástí instalace knihovny.

Projekt zahrnuje následující soubory, důležité pro tvorbu programové aplikace. Výsledná spustitelná podoba programu je umístěna po překladu programu v adresáři projektu */bin/Release*. Uvedené soubory, jsou uloženy na přiloženém CD příloze P I. v následujícím umístění *zdrojove_kody /kody / kody_programu/*. Jedná se o adresář projektu tvorby programu.

- Projektové soubory: *program_dp.cbp, program_dp.layout, program_dp.depend, sources_platform_mswindows.rc*
- Zdrojové implementační soubory: *data_operation_manager.cpp, data_xml_hierarchy.cpp, dialog_directory_setting.cpp, dialog_insuranceList.cpp, dialog_waiting.cpp, dialog_investigation_types_setting.cpp, frame_window.cpp, program.cpp, settings_manager.cpp, sorting_manager.cpp, statistics_visualization.cpp*
- Zdrojové hlavičkové soubory: *data_operation_manager.h, data_xml_hierarchy.h, dialog_directory_setting.h, dialog_insuranceList.h, dialog_waiting.h, dialog_investigation_types_setting.h, frame_window.h, identifiers.h, program.h, settings_manager.h, sorting_manager.h, statistics_visualization.h*
- Zdrojové soubory obrázků v podadresáři */icons_pictures*: *program_icon.ico, program_icon.xpm*
- Ostatní soubory: *design.fbp, prikazovy_radek.bat, prikazovy_radek.doc,* soubory adresáře *patients*/ (testovací sada dat vyšetření pacientů), soubor *genTestovDat .exe* adresáře *generovanaData/* (spustitelný testovací generátor dat), soubory nápovědy programu umístěné v adresáři *napoveda/*.
- Soubory generované samotným programem*: insuranceList.txt* (číselník pojišťoven), *program_settings.xml* (nastavení konfigurace programu při jeho používání)

Podoba konkrétního nastavení projektového souboru je významná tím, že bez její přesné definice by nemohl být projekt přeložen do tvaru spustitelného souboru. V editoru CodeBlocks se pro otevřený soubor projektu nadefinuje konfigurace prostřednictvím programového menu editoru *Project>Build options*. Na kartách dialogu nastavení je nutné uvést přesné definice dle obrázku (Obr. 31). Důležité je přidání definice cesty pro překladač zejména pro doinstalovávaný ovládací prvek wxFreeChart.



Obr. 31. Nastavení podmínek překladu projektu - v editoru CodeBlocks [19]

Nápovědu k funkcím programu s praktickými ukázkami jsem vytvořil v klasickém editoru webových stránek. Programovou ikonu jsem vytvořil ve verzi klasické i v provedení pro jiné operační systémy (soubor s příponou *.xpm*), a to pro případ budoucího rozšíření systému na jiné operační systémy (například distribuce systému Linux). Ostatní operační systémy totiž mívají odlišnou podporu typů obrázků.

9.2 Implementace oblastí architektury návrhu systému

Základní částí systému, která není jednorázově naprogramovatelná, ale je třeba ji komplexně doplňovat s postupně přidávanými funkcemi programu, je grafická a funkční oblast. Zahrnuje složitou logiku provázanosti jednotlivých zobrazovacích, nastavovacích dialogů, manažerů a jejich sjednocení v rámci celého navrženého systému. Oblast zahrnuje implementaci dialogů nastavení cesty k adresáři s daty vyšetření pacientů *dialog_ directory_setting.cpp,* dialog nastavení podporovaných typů vyšetření programem *dialog_investigation_types_setting.cpp*, dialog s informací o průběhu analýz a operací

dialog_waiting.cpp a dialog zobrazení číselníku kódů pojišťoven *dialog_ insuranceList.cpp*. Patří zde rámcové okno programu *frame_window.cpp*, logika programu *program.cpp*. Implementační soubory *.cpp* mají své shodně pojmenované deklarační soubory s příponou *.h*. Vzhled dialogů je definovatelný ručně zápisem kódu nebo je možné použít designér, který vygeneruje kostru samotného návrhu uspořádání ovládacích prvků na ploše dialogového okna. Soubor koster struktury návrhu vzhledu jednotlivých prvků navržené oblasti grafické a funkční je pojmenován *design.fbp*. V souboru jsem provedl návrh grafického rozhraní programu a vygenerovanou kostru v kódu, na míru doplnil a konkrétně programově zkompletoval. Oblast grafická a funkční je neoddělitelnou součástí od všech ostatních oblastí navržené architektury systému, zahrnuje objekty správců a manažerů patřících do oblastí sousedních. Hlavní rámcové okno jsem naprogramoval jako systém AUI s ovládacím prvkem třídy wxAuiNotebook, v jehož záložkách jsou vstupy a výstupy oblasti statistické analýzy dat (filtr, nastavení formátu výstupů, výsledky analýzy).

Oblast návrhu *Settings Manager* má na starosti udržování, ukládání a načítání programové konfigurace. Při prvním spuštění je konfigurace vytvořena v základním provedení a pro další spuštění programu tímto správcem uchovávána do doby ručních změn v nastavení programu uživatelem. Základním médiem pro uchování nastavení programu je soubor *program_settings.xml*, který program sám vytváří při jeho neexistenci nebo neúmyslném smazání v adresáři spustitelného programu. Nastavení zahrnuje definici cesty k adresáři dat vyšetření a seznam typů vyšetření, které program dokáže rozpoznávat v archivačním datovém úložišti. Správce nastavení jsem implementoval v souboru *settings_manager.cpp* společně s jeho datovým úložištěm v podobě hodnot ve formě proměnných. Správce dále používá funkcí oblasti *Data XML Hierarchy*, a to pro operaci ukládání a načítání dat konfigurace. Příkladem uložení nastavení je následující ukázka kódu v jazyce XML:

```
<?xml version="1.0" encoding="windows-1250"?>
<program_settings>
<investigationTypeElementName>audio</investigationTypeElementName>
<investigationTypeName><![CDATA[audio vyšetření]]></investigationTypeName>
<investigationTypeElementName>tympano</investigationTypeElementName>
<investigationTypeName><![CDATA[tympano vyšetření]]></investigationTypeName>
<investigationTypeElementName>ladicky</investigationTypeElementName>
<investigationTypeName><![CDATA[vyšetření ladičky]]></investigationTypeName>
<workingXmlDirectory><![CDATA[C:\patients]]></workingXmlDirectory>
</program_settings>
```

Oblast *Data XML Hierarchy*, implementovaná ve stejnojmenném souboru, se navenek tváří jako samostatná datová struktura. Ve skutečnosti se jedná o správce, který udržuje hodnoty XML dokumentu v jejich logické členitosti a pořadí. Navržený postup načítání dat pomocí zpracování znaků Meallyho konečným automatem je realizován prostřednictvím paměti aktuálního stavu Meallyho automatu a větvením typu *switch-case*. Zpracovávané znaky jsou analyzovány z textového souborového streamu, vyhodnoceny a uloženy podle procesu rozhodování do příslušného objektu dynamické kolekce datové struktury této oblasti ve formě popsané ve fázi návrhu. Objekt dynamické kolekce reprezentuje daný prvek XML logiky a je pomocí funkcí umožněna práce s objekty samotnými a jejich hodnotami. Vyhledávání již uložených prvků XML je v kolekci usnadněno formou vyhledávacích funkcí, ve kterých se udá název nadřazeného elementu XML a případně pořadí umístění v rámci něho, a to pro vícenásobný výskyt hledaného elementu.

V souboru *data_operation_manager.cpp* je implementována oblast pro práci se samotnými daty pacientů a vyšetření. Pro načítání dat využívám funkcí struktury *Data XML Hierarchy* a dotazuji se jí pouze již na logicky členěné hodnoty. Oblast manažeru poskytuje primárně data konkrétní oblasti statistické analýzy. *Data Operation Manager* také aktualizuje hodnoty filtrových kritérií jako jsou konkrétní výčet seznamu pacientů z dat, výčet konkrétních pojišťoven. Zajišťuje tak dynamické přizpůsobení podoby samotného filtru oblasti analýzy. Pro zmíněnou statistickou oblast plní manažer nejvíce podstatnou funkci, a to provedení filtrace dle jejích filtračních podmínek. Výsledkem jsou nové hodnoty, které manažer poskytuje oblasti analýzy k uložení, a to z důvodu, aby konkrétní data vyšetření uchovávaná v manažeru byla nepozměněna. *Data Operation Manager* zahrnuje mechanismus filtrace s ověřením současného splnění všech podmínek. Ověření příslušnosti konkrétního objektu vyšetření do filtračního časového vymezení a také soulad se všemi dalšími podmínkami (typy vyšetření, pojišťovny, výčet pacientů, ročníky narození). Takto jsou vyhodnocena všechna data a následně nabídnuta oblasti analýzy k uložení. Implementovaný *Data Operation Manager* má logickou podobu. Nemá žádné grafické výstupy pro systém. Stará se o logické načtení dat a o operace nad nimi, a to využitím funkcí a metod knihovny wxWidgets pro dynamické kolekce objektů. Pro tvorbu objektů dynamické kolekce je s výhodou použito konstruktoru jazyka C++. Parametry předané ve volání konstruktoru ukládám přímo do datových složek daného objektu, který má vzniknout. Po ukončení činnosti manažeru jsou položky kolekce odstraněny z paměti.

Implementace oblasti statistické analýzy a vizualizace jejích výsledků se nachází v souboru *statistics_visualization.cpp* a *statistics_visualization.h*. Z pohledu délky programového kódu je tato oblast nejrozsáhlejším úsekem programu. Zahrnuje totiž samotný statistický panel s definicí podmínek komplexního filtru (v podobě prvku třídy wxPropertyGrid), určením formátu zobrazení výsledků a jejich druhy (taktéž v podobě prvku třídy wxPropertyGrid). Oblast řeší spolupráci s *Data Operation Managerem* a obsahuje své tři dynamické datové kolekce objektů pro výsledky filtrace. Zajišťuje tedy proces samotné filtrace dat, jejich analýzy a vizualizaci výsledků v požadovaných tvarech a formách. Před samotným zobrazováním výsledků vždy dojde k abecednímu seřazení všech filtrovaných dat pomocí správce *Sorting Manager*. Abecední řazení je naprogramováno s ohledem na další členitost dat. Řazení zahrnuje efektivní způsob tříděný uvedený v kapitole 6.2 teoretické části práce. Nejprve jsou data při jednom průchodu rozdělena do konkrétní třídicí přihrádky *Sorting Manageru* dle počátečního písmena. Manažer dále již zajistí vnitřní seřazení všech dílčích přihrádek elementárním třídicím algoritmem Bubble-sort a na základě principu třídění spojováním seskupí data z přihrádek do seřazeného celku. Celek pak nabídne oblasti analýzy k samotnému statistickému zpracování. Pro analýzy počtů je provedeno nejprve abecední seřazení dat filtrační kolekce, a to podle dvou významných kategorií dat (např. jméno pacienta a datum typ vyšetření). Řadím dle více kategorií, protože se počty vyhodnocují jako četnost po sobě shodně vypovídajících záznamů. Například pokud následují za sebou tři záznamy tvaru („*Novák, audio*“), je taková posloupnost ohodnocena absolutní četností tři. Zjištěné výsledky pro různé typy statistik jsou vypisovány do textové podoby (prvek třídy wxTextCtrl), tabulky (prvek třídy wxGrid) a grafu (prvek třídy wxFreeChart). Samotné prvky při více obsažených hodnotách mají zabudované automatické rolování obsahu. V kódu je pracováno s funkcemi těchto prvků z pohledu definování jejich konkrétních obsahů. Hodnoty prvku textového pole jsou snadno kopírovatelné pro další zpracování uživatelem nebo případný tisk. Pro tabulku a stejně tak pro textové pole nabízím export do CSV, XML HTML, TXT formátů. Soubor CSV je uložením dat oddělených středníky dle příkladu:

rok; počet audio vyšetření; počet tympano vyšetření
2010;6;1
2006;1;0
2003;1;0
součty;8;1

V řadě případů jsem musel práci konkrétních ovládacích prvků doupravit. Příkladem byla situace, kdy jsem potřeboval pro definici složitějších podmínek filtrace prostřednictvím prvku třídy wxPropertyGrid zajistit možnost právě tohoto podrobnější zadávání. Pomocí principu skrytí kategorií mřížky a jejich znovu vykreslení v součinnosti se stavem hodnot přepínacích tlačítek jsem vytvořil pro uživatele možnost výběru mezi skupinami nastavení. Například jsem tímto způsobem vytvořil nastavení výběru pacientů. Je možné označit volbu *Všichni pacienti* nebo *Vybrat v seznamu*. Po kliknutí na druhou možnost se zpřístupní seznam konkrétních pacientů a v případě prvním je seznam skryt.

Konkrétní popis implementace jsem řadil přímo do programového kódu ve formě anglicky psaných komentářů. Komentáře vyjadřují význam daných příkazů z pohledu terminologie programování C++, programování v knihovně wxWidgets a také z pohledu logického a funkčního. Snažil jsem se co nejvíce kategorizovat jednotlivé kódové úseky podle navržených oblastí architektury systému. Vnořování a spolupráce jednotlivých oblastí je v kódu také popsána. Jako konkrétní ukázku zdrojového kódu bych uvedl kód oblasti *Data Operation Manager*, která má v následující metodě za úkol, pro kartu záložek nastavení filtru wxPropertyGrid konkrétní statistické oblasti, aktualizovat dynamický výčet jmen pacientů pro jejich výběr v rámci definice filtru uživatelem.

Ukázka zdrojového kódu oblasti *Data Operation Manager*:

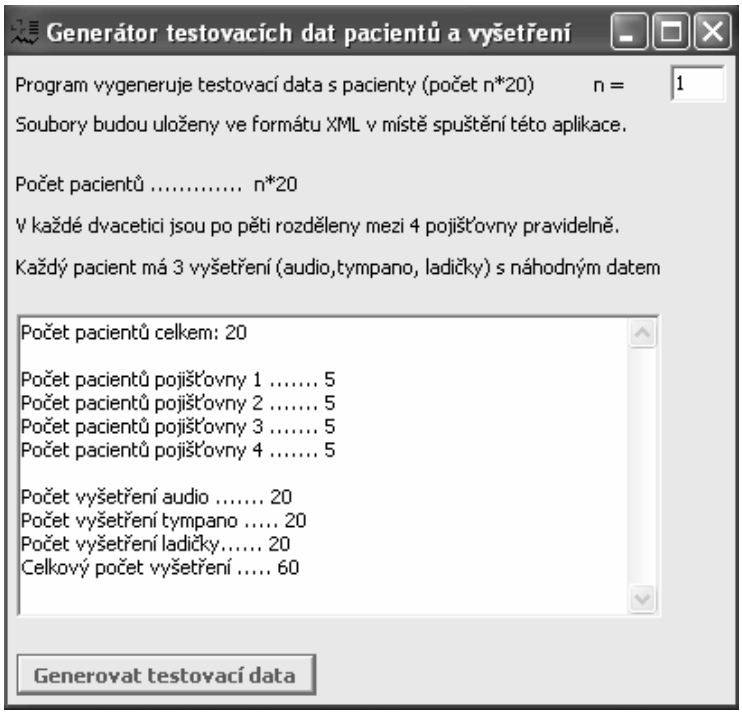
```
void CDataOperationManager::PanelFilter_Propgrid_LoadPatientsList(wxPropertyGrid* ptrPg,
wxPGId pgIdentifier)
{
   wxString patientPgLabel=wxT(""); // "Surname, Firstname (personal identif. number)"
   if (PatientsCollection.Count()>0)
   {
      for (unsigned int indexForPg=0; indexForPg<(PatientsCollection.Count()); indexForPg++)
      {
         patientPgLabel=PatientsCollection.Item(indexForPg).patient_surname;
         patientPgLabel+=wxT(", ");
         patientPgLabel+=PatientsCollection.Item(indexForPg).patient_firstname;
         patientPgLabel+=wxT(" (");
         patientPgLabel+=PatientsCollection.Item(indexForPg).patient_personal_id_number;
         patientPgLabel+=wxT(")");

         ptrPg->AppendIn(pgIdentifier,
              wxBoolProperty(patientPgLabel, wxPG_LABEL, false));
      }
   }
   return;
}
```

9.3 Tvorba programu testovacího generátoru dat

Navržený testovací systém jsem implementoval také v podobě softwarové aplikace. Soubory zdrojových kódů testovacího generátoru jsem umístil na přiložené CD přílohu P I do adresáře *zdrojove_kody /kody / kody_testovaciho_generatoru/*. Spustitelný soubor *genTestovDat.exe* se nachází v umístění *zdrojove_kody/ kody / kody_programu/.*

Program má za úkol generovat sekvenci dat pro účely testování aplikace, jak jsem již uvedl v etapě návrhu systému. Konkrétně po stránce funkční je generována *k*-tice dat o počtu *n*20*, kde *n* je celočíselný multiplikativní koeficient. Základní dvacetice tvoří základ a je možné generovat její celočíselné násobky. Statistiky pro každou dvacetici dat jsou rovnoměrné a pravidelné. Čtvrtina dat každé dvacetice patří rovnoměrně jedné ze čtyř pojišťoven. Každý pacient v rámci dvacetice má danou trojici typů vyšetření *audio, tympano* a *ladičky*. Pro generování testovaných dat je možné určit jejich počet. Výsledná data jsou uložena ve smyslu způsobu archivace ORL dat a nacházejí se po samotném procesu generování v místě spustitelného souboru *genTestovDat.exe*. V okně generátoru se nachází souhrn statistik ve formě četností jednotlivých údajů pro množinu generovaných dat. Tyto údaje musí souhlasit právě pro vyhodnocení statistik navrhovaným systémem.



Obr. 32. Implementovaný testovací generátor pro ověření správnosti výpočtů systému

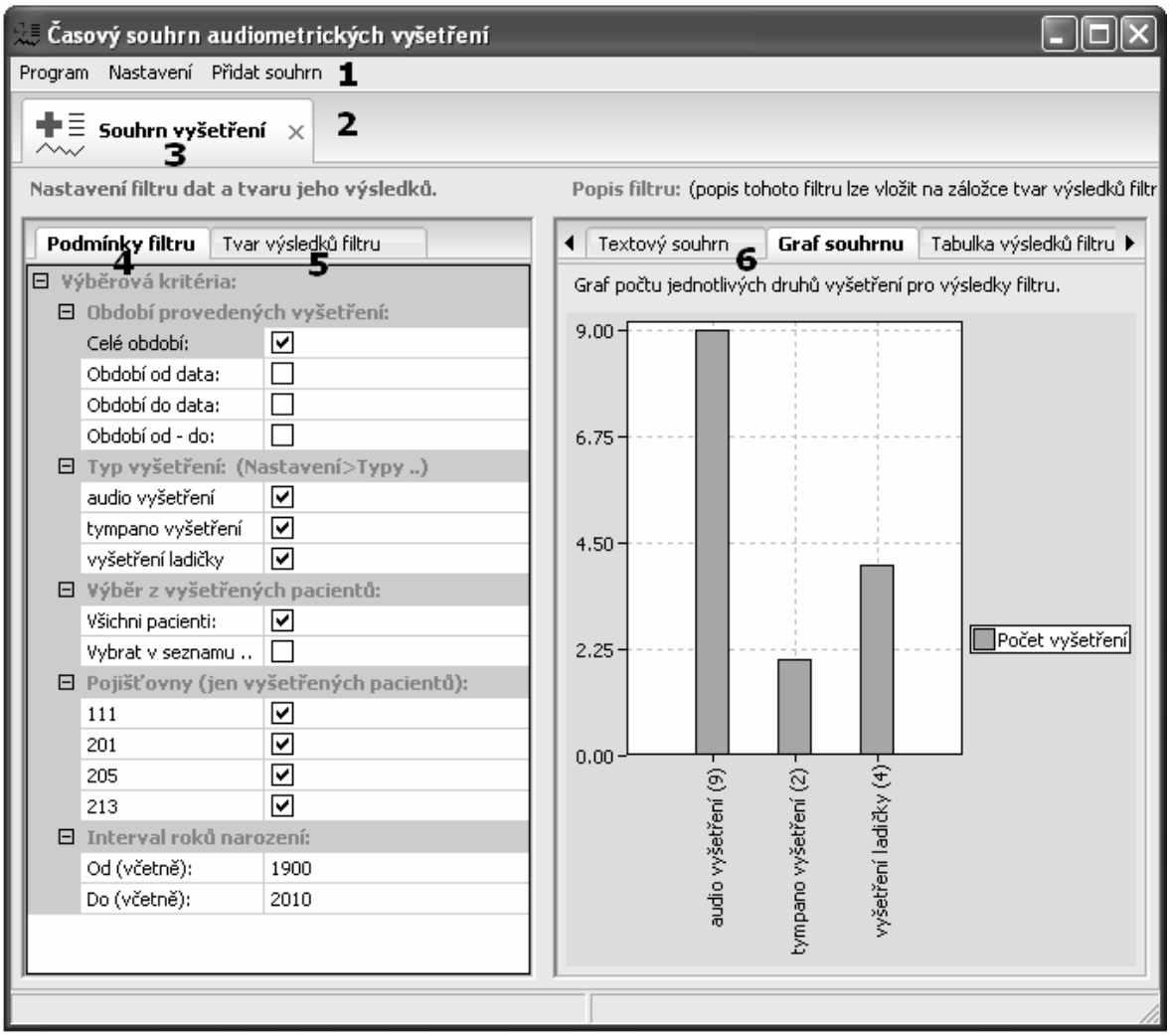
10 PŘEHLED FUNKCÍ VYTVOŘENÉHO SYSTÉMU

Vytvořený systém bude nyní popsán z pohledu implementovaných funkcí. Praktické ukázky práce s programem a detailní vysvětlení je provedeno v přiložené vytvořené nápovědě k programu v umístění *program/napoveda/* na přiloženém CD příloze P I. Samotná složka programu je v adresáři *program/*. Složku je nutné celou zkopírovat na pevný disk, čímž je také provedená instalace programu. Program není doporučeno spouštět přímo z disku CD, protože se jedná o médium určeno pouze pro čtení a tato skutečnost by neumožnila programu ukládat své nastavení a konfiguraci. Program je reprezentován spustitelným souborem *program_dp.exe* v adresáři *program/*.

Než-li bude samotný program zobrazovat cokoliv ve svém obsahu, musí uživatel nadefinovat v úvodu po jeho spuštění v zobrazeném dialogu cestu k datům ORL vyšetření. Pokud uživatel zadá adresář chybně nebo adresář nebude obsahovat data pacientů, neumožní program provádět uživateli další činnosti. Nastavený adresář, pokud se v průběhu času nezmění jeho umístění, bude zapamatován programem a po příštím spuštění nebude již jeho zadávání znovu potřeba. Nastavená cesta se uloží totiž do souboru konfigurace *program_settings.xml*. Přes menu programu je dále umožněno během práce s programem kdykoliv adresář změnit, a to pro případ, že existují například dva adresáře s daty vyšetření.

Obr. 33. Příklad nekorektně a korektně určeného adresáře s daty vyšetření ORL

Po zadání správného adresáře uživatelem programu se zpřístupní hlavní část programu. Je zobrazena úvodní obecná záložka statistické analýzy. Aby byl význam jednotlivých částí okna programu zřejmý, popíši nyní skladbu jejich částí na následujícím obrázku (Obr. 34).



Obr. 34. Základní skladba výsledného systému

Popis části dle schématu (Obr.34):

- **1** - Programové menu - *Program, Nastavení, Přidat souhrn*
- **2** - Záložkový systém pro práci s více přidanými statistikami jako celky
- **3** - Konkrétní zobrazená záložka statistické analýzy
- **4** - Definice filtračních podmínek pro data ordinace ORL
- **5** - Definice druhů zobrazované statistiky výsledků a jejich formát
- **6** - Zobrazení statistik v různých tvarech podle **5**, pouze pro filtrovaná data podle **4**

Menu programu obsahuje nabídky pro zajištění následujících funkcí:

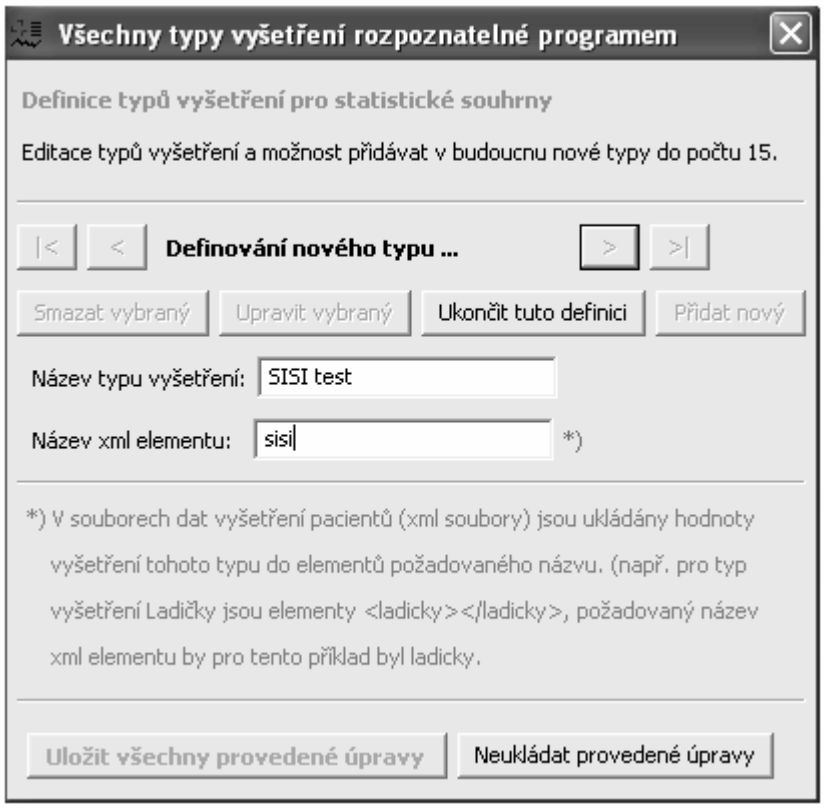
Menu *Program*:
- *Aktualizace obsahu opětovným načtením dat (klávesa F5)*
- *Informativní seznam pojišťoven a jejich kódů*
- *Ukončit program*

Menu *Nastavení*:
- *Určit adresář s daty pacientů…*
- *Typy vyšetření rozpoznatelné programem/přidat nový typ ..*

Menu *Přidat souhrn*:
- *Nový souhrn*
- *Seznam pacientů*
- *Seznam pojišťoven*
- *Seznam druhů vyšetření*
- *Přehled pacientů a jejich vyšetření*
- *Přehled provedených vyšetření*
- *Počty vyšetření pacientů*
- *Počty vyšetření pojišťoven*
- *Počty pacientů pojišťoven*
- *Roční souhrn vyšetření*
- *Měsíční souhrn vyšetření*
- *Denní souhrn vyšetření*

Za běhu programu lze kdykoliv aktualizovat statistické výsledky klávesou *F5* nebo skrze menu, pokud nastala změna v datech vyšetření. Tato situace může nastat v případě, že je program spuštěný a současně jsou též spuštěným programem ordinace ORL provedeny změny v datech (např. přidání nového pacienta s vyšetřením). Aby bylo možné analyzovat konkrétní informace o vyšetřeních z dat ORL, musí být nadefinováno, které vyšetření program dokáže rozpoznat. Po prvním spuštění je sada rozpoznatelných typů vyšetření nastavena automaticky na *audio, tympano*, a *ladičky*. Pokud bude v budoucnu potřeba vyhodnocovat programem také jiný nový typ vyšetření, pak je nutné tento typ dodefinovat v nastavení typů. Jedná se o technické nastavení, které se provádí jen při

zavedení nových typů vyšetření. Spuštění nastavení je možné skrze menu *Nastavení > Typy vyšetření rozpoznatelné programem/přidat nový typ*. Bude zobrazen konfigurační dialog, ve kterém uživatel může mazat či pojmenovávat již vložené typy vyšetření a také přidat požadovaný nový typ, který se má posléze také ve statistikách objevovat. Na obrázku (Obr. 35) uvádím způsob přidání nového typu vyšetření, který standardně není nastaven. Jedná se o postup nadefinování typu „SISI test“. Vložení se provede stisknutím tlačítek *Ukončit tuto definici* a následném *Uložit všechny provedené změny*.



Obr. 35. Přiklad přidání nového typu vyšetření v budoucnu

Pouze v nastavení uvedené typy budou programem rozpoznatelné. Nastavení se ukládá do konfiguračního souboru *program_settings.xml*. Pokud bude soubor omylem smazán, vytvoří si ho program znovu s opět nadefinovanými základními typy *audio, tympano, ladičky*.

Základním objektem statistické analýzy je obecná záložka souhrnu tvořená nastavitelným filtrem a výsledky. Po prvním spuštění programu je automaticky přidána do programu. Na schématu (Obr.34) je vyznačena číslem 3. Lze ji libovolně vícekrát přidat skrze menu

Přidat souhrn > Nový souhrn. Obecná záložka je charakteristická tím, že není přednastavena a je na uživateli aby si nadefinoval filtr a zvolil druh statistiky a formát zobrazení výsledků. Tyto nastavení se dějí na kartě nastavení *Podmínky filtru* a *Tvar výsledků filtru*. Změnou těchto podmínek se okamžitě vykreslí a změní výsledky statistické analýzy pro data, která uživatelem takto nadefinovaným filtrem projdou. Nastavení filtru a formátu výsledků je velice univerzální.

V programu jsem obecné záložky již přednastavil do konkrétních často používaných souhrnů. Jedná se o již nastavené obecné záložky s konkrétním předchystaným nastavením filtru a tvaru výsledků. Takto předdefinované záložky lze otevřít a přidat do programu přes programové menu *Přidat souhrn* a ze seznamu si vybrat již konkrétní typ přednastavené záložky statistiky. Jedná se o souhrny četností (*Počty vyšetření pacientů, Počty vyšetření pojišťoven, Počty pacientů pojišťoven)*, seznamy (*Seznam pojišťoven, Seznam pacientů, Seznam druhů vyšetření)*, přehledy (*Přehled pacientů a jejich vyšetření, Přehled provedených vyšetření)*, časové souhrny (*Roční souhrn vyšetření, Měsíční souhrn vyšetření, Denní souhrn vyšetření*).

Zaměřím se nyní na popis částí obecné záložky statistické analýzy. Popis platí i pro předdefinované záložky, s tím rozdílem, že jsou již předem za uživatele nastaveny. První částí záložky je nastavení filtru pro data a formát zobrazení výsledků. Druhá část je systém sestav a souhrnů ve formě grafů, textových polí a tabulek pro vyjádření filtrovaných výsledků a jejich statistik. Pokud nastane změna v první části záložky, projeví se ihned v části druhé. To platí také pro případ aktualizace klávesou *F5*.

Filtr je navržen jako dynamický a univerzální. Řadu hodnot v rámci svých seznamů nastavení načítá dynamicky přímo z dat (výčet seznamu pojišťoven, pacientů). Uživatel provede nastavené podmínek filtru zatržením příslušných kolonek myší. Filtr funguje také na přímém principu zadávání hodnot (období) a jejich automatické kontroly. Podmínkami jsou *Období provedených vyšetření*, *Typy vyšetření*, *Výběr z vyšetřených pacientů, Pojišťovny, Interval roků narození.* Filtrem projdou pouze ta data, která splní všechny uživatelem stanovené podmínky současně. Pro případ, že neprojdou filtrem žádná data, je ve výsledcích uvedena zpráva o 0 nalezených výsledcích pro filtr.

Na uvedeném výčtu uvádím všechny typy definic filtračních podmínek pro data:

Období provedených vyšetření - projdou data pouze pro takto stanovené období:

- *Celé období* (nebude provedena časová filtrace)
- *Období od data* (zadání počátečního data období do dnešního data)
- *Období do data* (zadání koncového data období, standardně zobrazen aktuální datum)
- *Období od-do* (zadání úplného časového intervalu)

Typy vyšetření - projdou data pouze takto vybraných druhů vyšetření:

- Dynamicky načtený seznam typů vyšetření rozpoznatelných programem

Výběr z vyšetřených pacientů - výběr konkrétních pacientů:

- *Všichni pacienti* (nebude provedeno omezení na data konkrétního pacienta)
- Načtený seznam pacientů z dat vyšetření ORL seřazený dle jmen
- Načtený seznam pacientů z dat vyšetření ORL seřazený dle rodných čísel

Pojišťovny - projdou data vyšetření pacientů konkrétně vybraných pojišťoven:

- Načtený seznam pojišťoven z dat vyšetření ORL seřazený dle jmen

Interval roků narození - projdou data pacientů odpovídajících ročníků narození:

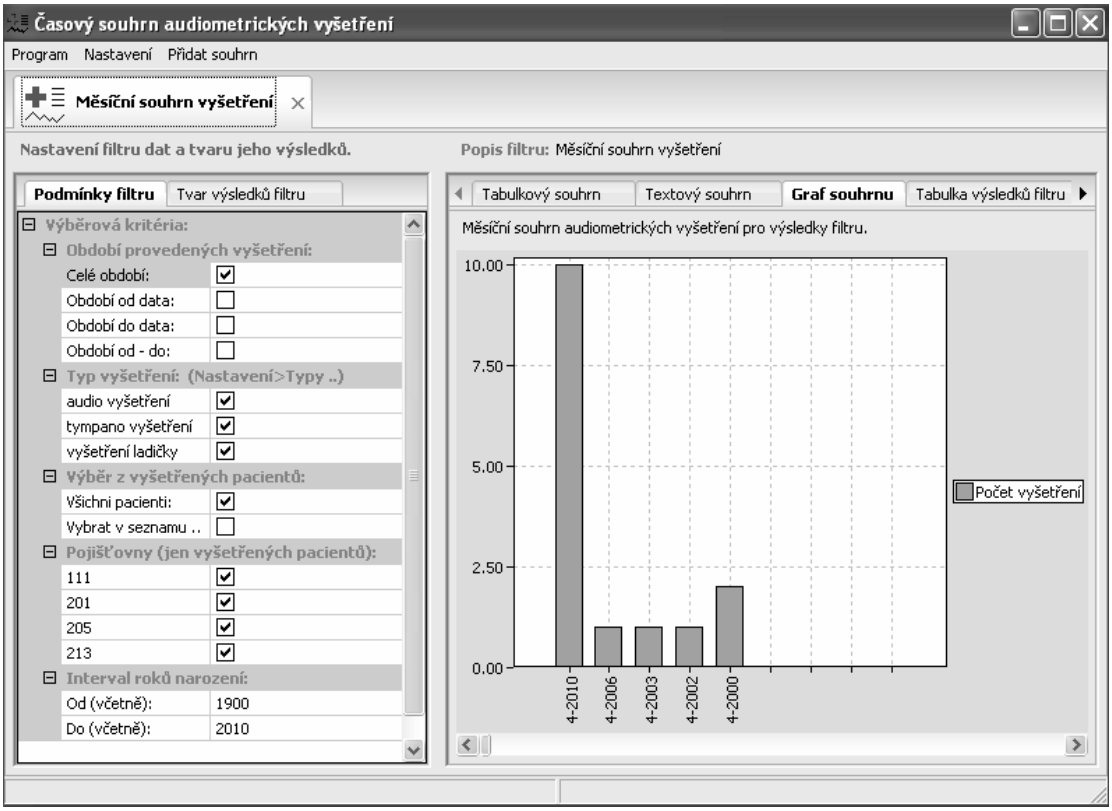
- *Od (včetně)*
- *Do (včetně)* (standardně zobrazován rok aktuálního data)

Pro výsledky analýzy dat, které prošly filtrem, program zahrnuje několik forem jejich zobrazení. Jedná se o výsledkové souhrny - *Tabulka výsledků filtru, Sestava výsledků filtru* a statistické souhrny - *Tabulkový souhrn, Textový souhrn, Graf souhrnu.* Jejich zobrazování lze nastavit na kartě *Tvar výsledků filtru* v rámci záložky statistické analýzy. *Tabulkový souhrn, Textový souhrn, Graf souhrnu* se nastavují společně v kritériu *Souhrny*. *Tabulka výsledků filtru, Sestava výsledků filtru* mají oddělené nastavení formátu zobrazení výsledků v položkách stejně pojmenovaných - *Tabulka výsledků filtru, Sestava výsledků filtru*. Pro souhrny (tabulka, text, graf) se v nastavení určí, jaký druh statistiky budou vyobrazovat. Se změnou nastavení formátu zobrazení výsledků se změny ihned projeví v obsahu ovládacích vizualizačních prvků. Pokud je dat k analýze mnoho, může jejich zpracování trvat určitou dobu. Pro tento případ se zobrazuje hlášení indikátorem čekání.

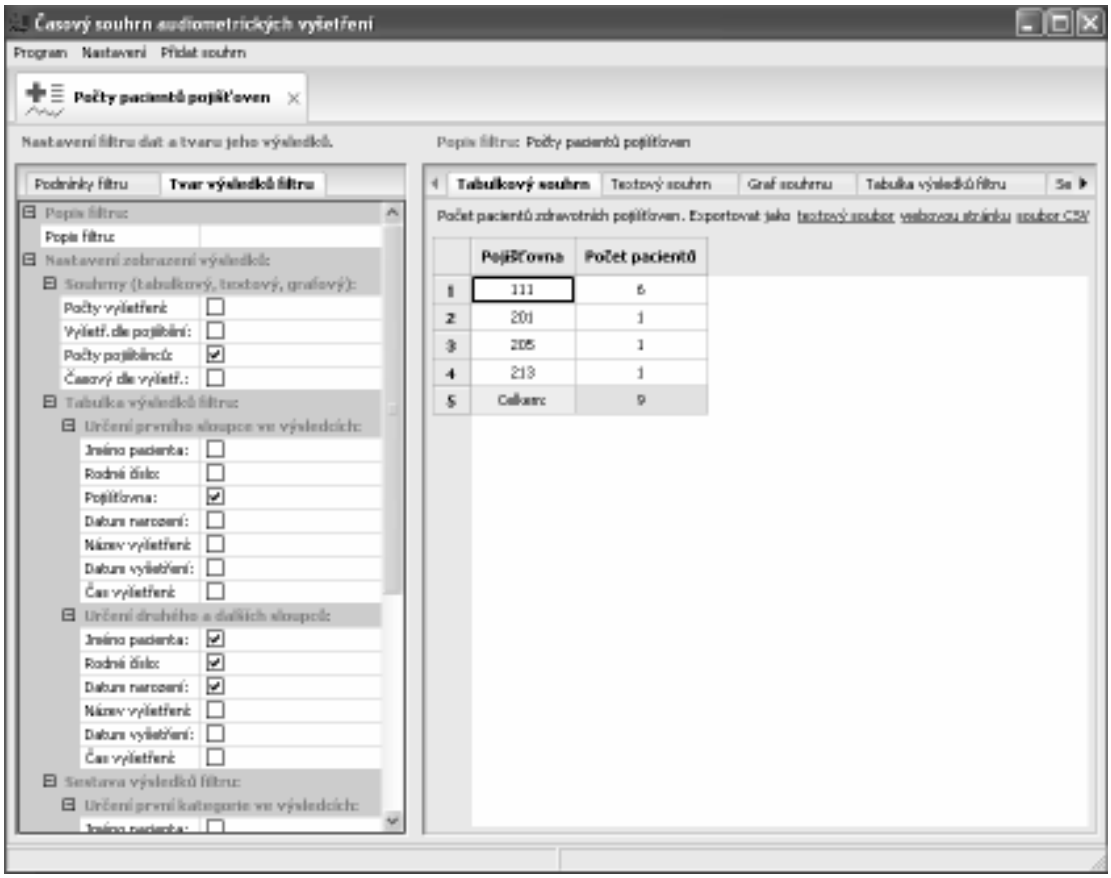
U výpisu výsledků (*Tabulka výsledků filtru, Sestava výsledků filtru)* lze nadefinovat, které sloupce zobrazovat a které nikoliv, který sloupec uvést jako první (sloupce: *Jméno pacienta, Rodné číslo, Pojišťovna, Datum narození, Název vyšetření, Datum vyšetření, Čas vyšetření*). První sloupec určuje abecední řazení. Tabulka i textová sestava výsledků se dá určením zobrazovaných sloupců využít i pro získání informací. Například pro jediný sloupec - *Pojišťovna* obdržíme seznam pojišťoven. Skrytím sloupců v tabulce nebo textové sestavě výsledků dále můžeme obdržet seznam typů vyšetření, jmenný seznam pacientů - všechny možné kombinace zobrazení pohledu na tytéž výsledky filtrace. Nastavení, že se mají zobrazit veškeré sloupce, poskytne vyobrazení kompletních údajů o filtrovaných datech. Pokud při nezobrazení určitých sloupců se ve výsledcích vyskytnou shodné záznamy např. *„Novák 550606/6666“* a *„Novák 550606/6666“* budou sloučeny do jednoho záznamu.

Tabulkový, textový souhrn a graf vyjadřuje v různých podobách zobrazeny tytéž výsledky statistické analýzy. Jedná se především o vyhodnocení počtů daných entit vzhledem k jiné entitě. Například vyhodnocení počtu vyšetření vzhledem k pojišťovnám nebo jednotlivým pacientům. Pro časové souhrny se vyhodnocují počty vzhledem k časovým intervalům (roky, měsíce, dny). U časovým souhrnů je provedeno řazení dat opačné, než-li pro nečasově orientované výsledky. Důvodem je zobrazování nejstarších časových údajů v posledních řádcích tabulky, grafu, textového výpisu. Pokud je v prvcích výsledků souhrnů mnoho, je zajištěna možnost rolovat obsah prvků pomocí posuvníků. Druhy statistik jsou *Počty vyšetření, Vyšetření dle pojištění, Počty pojištěnců, Časový souhrn – roční, měsíční, denní*. Pro souhrn typu *Počty vyšetření* lze dále nastavit, zda-li mají výsledky zahrnovat pouze jméno a příjmení pro jednotlivé pacienty nebo zobrazit kompletní informace o pacientovi s rodným číslem, datem narození.

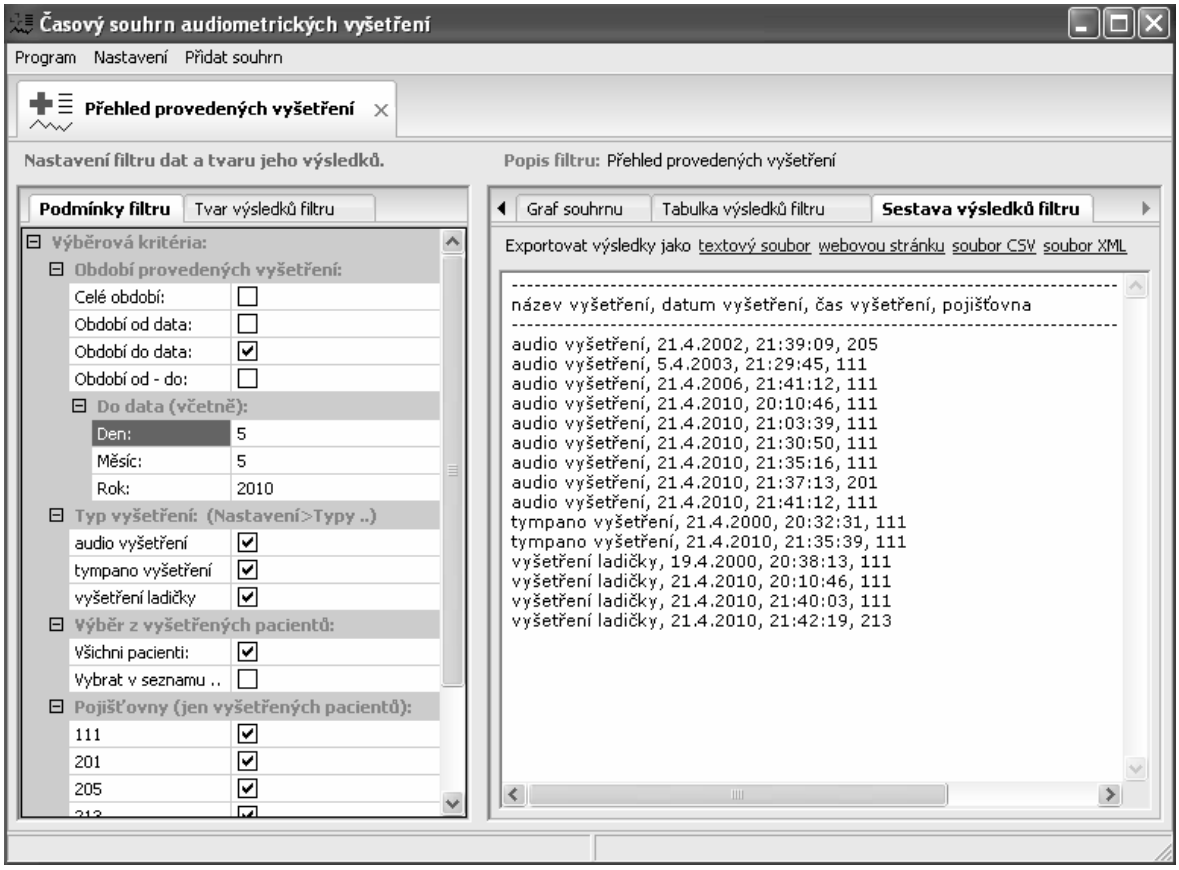
Nastavení filtru ovlivňuje, jaká data projdou podmínkami stanovenými uživatelem. Tvar zobrazení výsledků určí druh statistického výpočtu nad filtrovanými daty a formát jejich zobrazení. Výsledky z výpisů i souhrnů lze ukládat pro účely dalšího zpracování mimo tento program a dále je například tisknout. Export dat je umožněn do textového souboru, webové stránky s textem nebo tabulkou a do souborů XML a CSV.



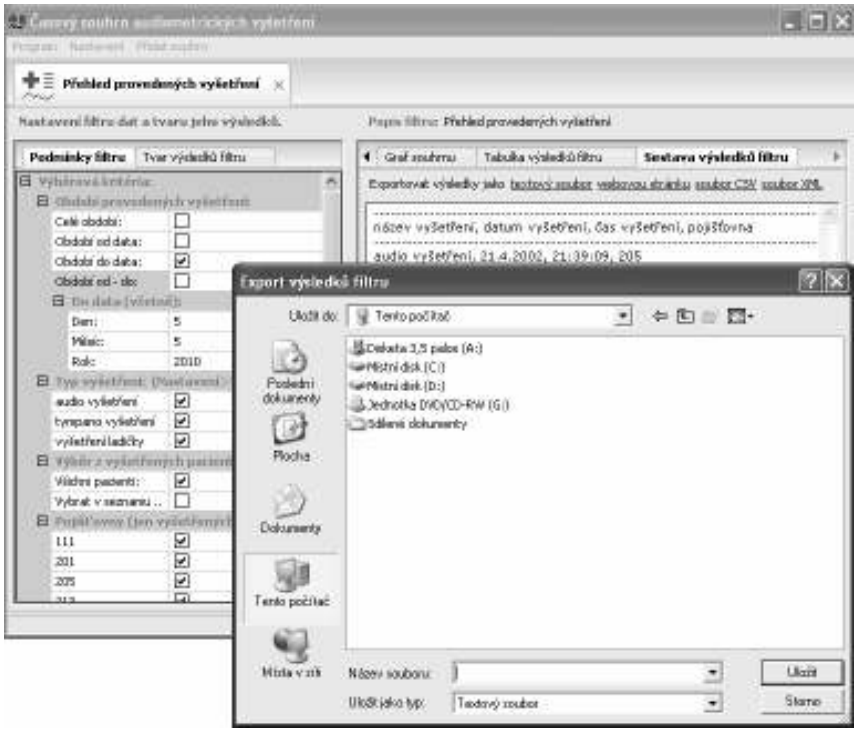
Obr. 36. Ukázka programu s měsíčním souhrnem dat ORL



Obr. 37. Ukázka programu s přehledem počtů pacientů pojišťoven



Obr. 38. Ukázka programu s vyobrazením výsledků textovou sestavou



Obr. 39. Ukázka exportu výsledků analýzy pro případné další zpracování

11 TESTOVÁNÍ SPRÁVNOSTI VÝSLEDKŮ SYSTÉMU, MĚŘENÍ ČASOVÉ SLOŽITOSTI VYBRANÝCH POSTUPŮ ŘEŠENÍ

11.1 Testování správnosti výsledků

Aby mohl být systém nasazen do praxe, musí být nejprve provedeno otestování správnosti jeho výsledků. Jedná se o výsledky statistické analýzy. Fáze testování je velice důležitou, protože ověří stabilitu programu také při nadměrné datové zátěži. Pro ověření správnosti používám testovací generátor *genTestovDat.exe* (dostupný na přiloženém CD - příloze P I. v adresáři *zdrojove_kody/ kody /kody_programu/generovanaData*), který jsem pro potřeby testování vytvořil. Ve fázi realizace jsem uvedl, v jakém tvaru testovací generátor dává hodnoty statistik pro svá generovaná data. Výsledky musí být hodnotami, které vydá systém ve formě tabulek, grafů a sestav. Obsahem jednotlivých testovacích dat jsou náhodně tvořená jména pacientů a údajů vyšetření. V testovacím generátoru se nejdříve určí počet generovaných údajů, které jsou načteny vytvořeným systémem. V okně generátoru se objeví pro data výsledky statistik, které mají totožně vyjít v systému. Testování se provádí pro menší počet dat, ale také pro jejich nadměrné množství, což ověřuje činnost systému v zátěžových situacích. V kapitole 11.2 navíc uvádím, jak dlouho trvaly jednotlivé fáze statistické analýzy a načtení dat právě pro tyto velké objemy. Dále uvádím záznam z procesu testování (Tab.3), který dokazuje správnost práce navrženého systému.

Průběh testování (výsledek generátoru/**výsledek ze systému**)							
Počet pacientů	*Počet vyšetření*			*Počet pacientů i-té pojišťoven (i =1,2,3,4)*			
	audio	*tympano*	*ladičky*	*1.*	*2.*	*3.*	*4.*
60	60/**60**	60/**60**	60/**60**	15/**15**	15/**15**	15/**15**	15/**15**
240	240/**240**	240/**240**	240/**240**	60/**60**	60/**60**	60/**60**	60/**60**
420	105/**105**	105/**105**	105/**105**	420/**420**	420/**420**	420/**420**	420/**420**
600	150/**150**	150/**150**	150/**150**	600/**600**	600/**600**	600/**600**	600/**600**
780	195/**195**	195/**195**	195/**195**	780/**780**	780/**780**	780/**780**	780/**780**
960	240/**240**	240/**240**	240/**240**	960/**960**	960/**960**	960/**960**	960/**960**
1140	285/**285**	285/**285**	285/**285**	1140/**1140**	1140/**1140**	1140/**1140**	1140/**1140**
1320	330/**330**	330/**330**	330/**330**	1320/**1320**	1320/**1320**	1320/**1320**	1320/**1320**
1500	375/**375**	375/**375**	375/**375**	1500/**1500**	1500/**1500**	1500/**1500**	1500/**1500**
1680	420/**420**	420/**420**	420/**420**	1680/**1680**	1680/**1680**	1680/**1680**	1680/**1680**

Tab. 3. Testování správnosti výsledků systému pomocí testovacího generátoru

11.2 Měření časové složitosti aplikace

V programovém kódu jsem měřil časovou složitost výpočetně náročnějších postupů, kterými jsou načítání dat vyšetření, analýza s výsledky směrovanými do textového, tabulkového výpisu a statistického souhrnu obecně. Protože pro velké objemy dat dochází ke zpožděním, zmapoval jsem je následujícím měřením, které se opírá o teorii stanovení výpočetní složitosti algoritmu, uvedené v kapitole 2.5. Způsoby výpočtů, jejichž časovou složitost měřím jsou vylepšeny zvolenými postupy v rámci návrhu řešení. Pro konkrétní postup řešení vždy stanovuji odhad složitosti (ze struktury programového zápisu řešení) a uvádím naměřenou variantu. Naměřené hodnoty časových složitostí se nachází v souboru *mereniCasoveSlozitosti.xls* v adresáři *zdrojove_kody/kody_programu/* na CD příloze P I. Pro měření jsem použil v rámci programu tento kód:

```
#include "time.h"
// --- the time measuring start ---
wxLongLong timeStart, timeEnd;
long timeResult;
timeStart=wxGetLocalTimeMillis();
/*  the testing code  */
// -----the time measuring end ------
timeEnd=wxGetLocalTimeMillis()-timeStart;
timeResult=timeEnd.ToLong();
wxString strTimeResult;
strTimeResult=wxString::Format(wxT("%d [ms]"),timeResult);
wxMessageBox(strTimeResult,wxT("time measuring result"));
```

Odhady řádu asymptotické funkce podle počtu vnořených for-cyklů v kódu:

Načítání dat pacientů a vyšetření:

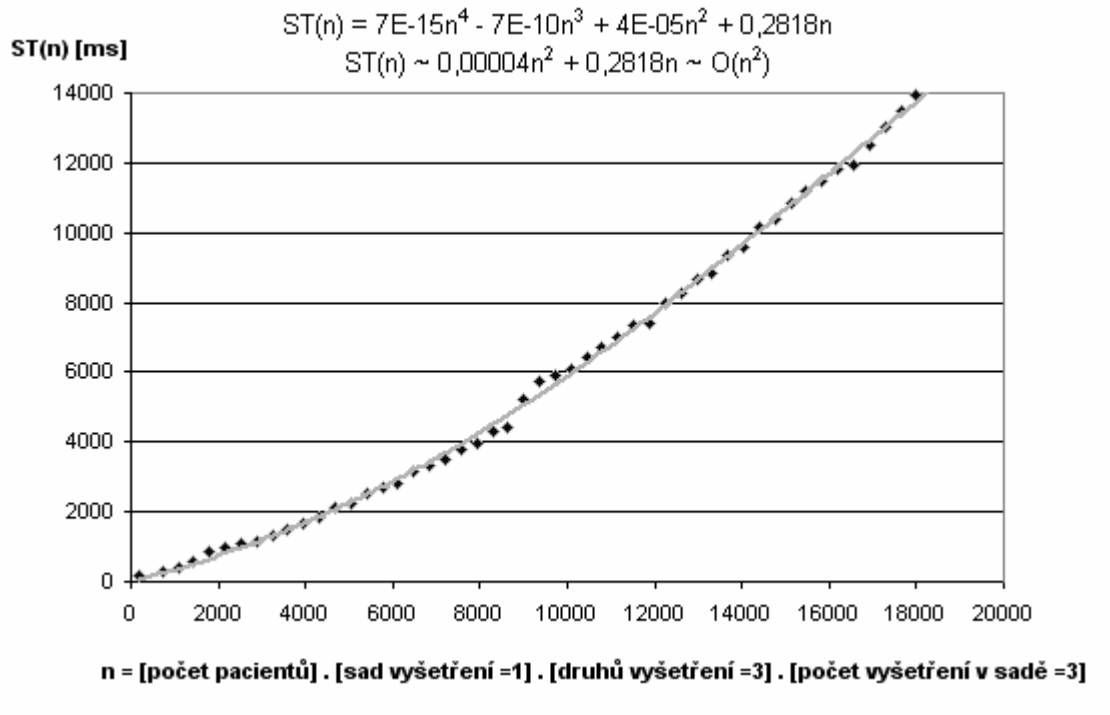
*[počet pacientů] * [počet sad vyšetření] * [počet druhů vyšetření] * [počet vyšetření]* = $\boldsymbol{n^4}$

Vytvoření řazení textové sestavy a tabulky výsledků filtru:

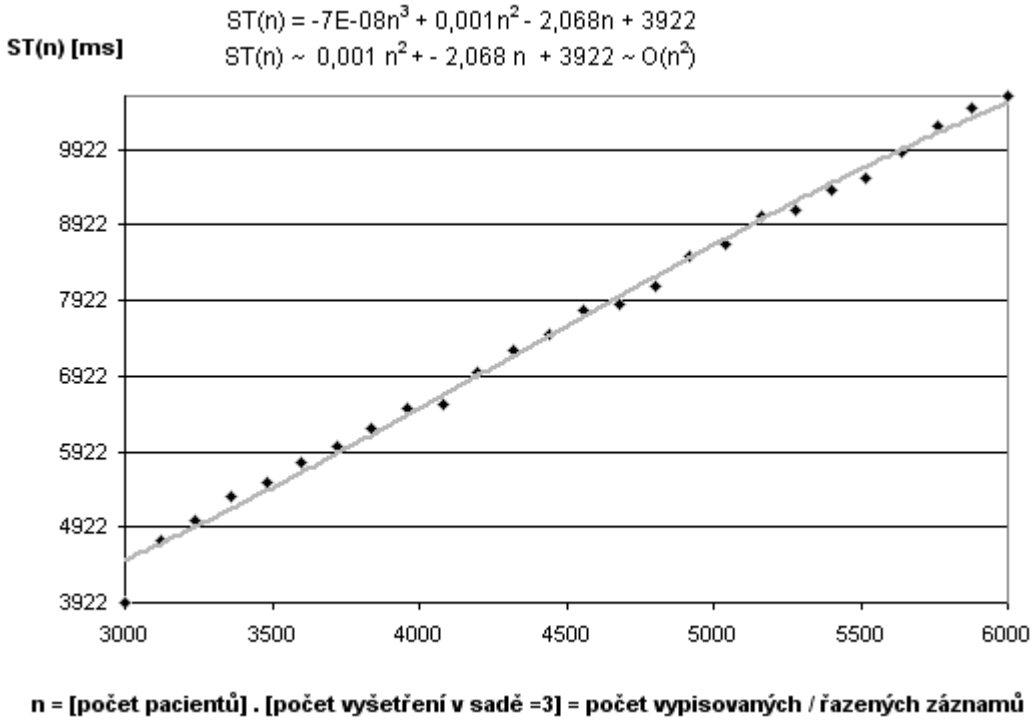
2 for-cykly pro abecední řazení + 1 cyklus pro výpis (asymptoticky neovlivní) = $\boldsymbol{n^2}$

Provedení statistických výpočtů a jejich výpis do statistického souhrnu:

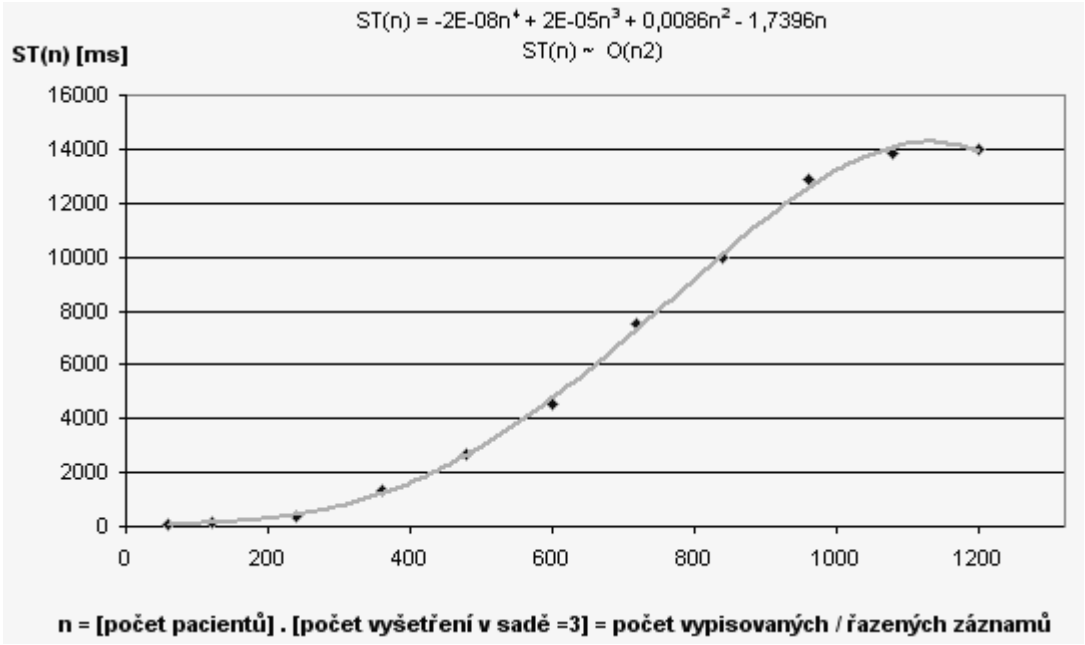
2 for-cykly pro abecední řazení + 1 cyklus pro výpis (asymptoticky neovlivní) = $\boldsymbol{n^2}$



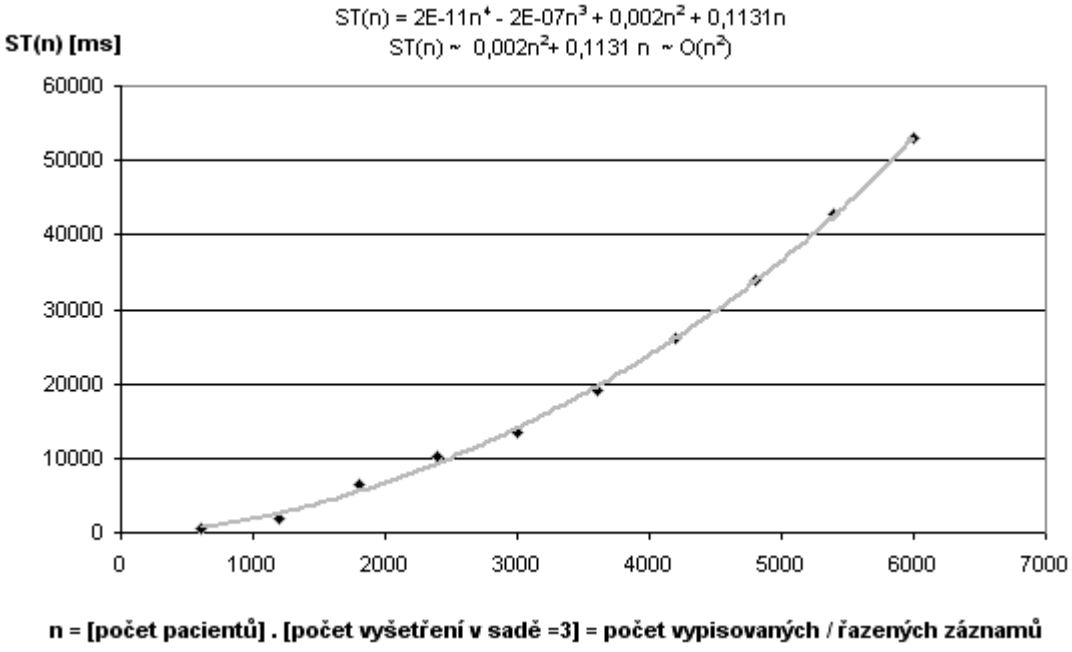
Obr. 40. Graf naměřené časové složitosti operace načtení dat pacientů a vyšetření



Obr. 41. Graf naměřené časové složitosti vytvoření a řazení výsledků textové sestavy



Obr. 42. Graf naměřené časové složitosti vytvoření a řazení výsledků tabulky



Obr. 43. Graf naměřené časové složitosti statistických výpočtů a zobrazení výsledků

Z výsledků měřené časové složitosti popisované asymptotickou funkcí *ST(n)* vyplývá, že pro velký objem dat *n* se zvýší výpočetní složitost, a to úměrně druhé mocnině počtu vstupních parametrů problémů. Protože jsou zvolené metody řešení (načítání dat, jejich řazení a analýza) navrženy co nejvíce efektivně, tak jako opatření pro čekání na výpočty nadměrného množství dat zobrazuji popisek s informací o probíhajícím výpočtu nebo načítání dat. Samotný postup načítání údajů ze souborů pevného disku do datových struktur a způsob jejich analýzy urychlit nikterak nelze. Aby uživatel nebyl čekáním na zpracování např. 6000 dat znepokojen, je mu zobrazen popisek vypovídající o stavu programu. Pro počet dat přibližně 6000 by doba čekání neměla podle naměřených hodnot přesáhnout hranici 60 sekund pro jednotlivé postupy řešení v rámci programu. Pro počet dat vyšetření pohybujících se kolem hodnoty 1000 se zpoždění činnosti programu nikterak významně neprojevuje.

12 OVĚŘENÍ ŘEŠENÍ NA ROZSÁHLEJŠÍ DATABÁZI VYŠETŘENÍ

Korektní výsledky testování správnosti výsledků pro menší i rozsáhlou sadu testovacích dat v kapitole 11.1 ukázaly, že program vyhodnocuje všechny prováděné statistické výpočty správně. A to i pro rozsáhlé sekvence testovacích dat ORL vyšetření o počtu šesti tisíc. V této kapitole je uvedeno již konkrétní ověření řešení na skutečných datech z reálné praxe. Před samotným nasazením vytvořeného programu do praktického používání je nutné tento druh ověření provést. Pro výsledky statistických analýz uvádím také jejich interpretaci, co které hodnoty znamenají z pohledu jejich praktického významu.

Z důvodu, že se jedná o reálnou databázi skutečných pacientů, budu uvádět pouze výsledky číselného charakteru. Jména ani rodná čísla ve výpisech uvádět nemohu z důvodu ochrany osobních údajů pacientů. Konkrétní data vyšetření, na kterých ověření provádím, jsou z lékařské ordinace otrokovické polikliniky se specializací ORL. Jedná se o vyšetření pacientů jednorázových ORL vyšetření, ale také pacientů, jenž absolvovali opakovaná vyšetření. Po načtení dat z archivačního adresáře program statisticky vyhodnotil počty provedených vyšetření a další typy analýz a jejich výsledky nyní uvádím.

Druh souhrnu: *Seznam pacientů*
Interpretace výsledků: Výčet pacientů, kteří ordinaci navštívili.
Výsledky zahrnují jména o počtu *198* s uvedením rod. čísel, dat narození a pojišťovny.
(neuvádím blíže z důvodu ochrany osobních údajů pacientů)

Druh souhrnu: *Seznam pojišťoven*
Interpretace výsledků: Výčet všech pojišťoven, ke kterým jednotliví pacienti náleží.
Výsledky: *pojišťovny s číselnými kódy: 111, 201, 205*
(jedná se o obecný seznam bez uvedení dalších specifických údajů)

Druh souhrnu: *Seznam druhů vyšetření*
Interpretace výsledků: Výčet všech typů vyšetření rozpoznatelných programem.
Výsledky: *audio vyšetření, tympano vyšetření, vyšetření ladičky*
(jedná se o obecný seznam bez uvedení dalších specifických údajů)

Druh souhrnu: *Přehled pacientů a jejich vyšetření*
Interpretace: Uvedení informací o všech provedených vyšetřeních pacientů ve tvaru *jméno rod.č., pojišťovna, datum narození, název vyšetření, datum vyšetření, čas vyšetření*.
Výsledky zahrnují *419* údajů, informací o kompletně provedených vyšetření pro jednotlivé pacienty. Výsledky jsou řazeny abecedně dle jmen pacientů a dále vnořeně.
(neuvádím blíže z důvodu ochrany osobních údajů pacientů)

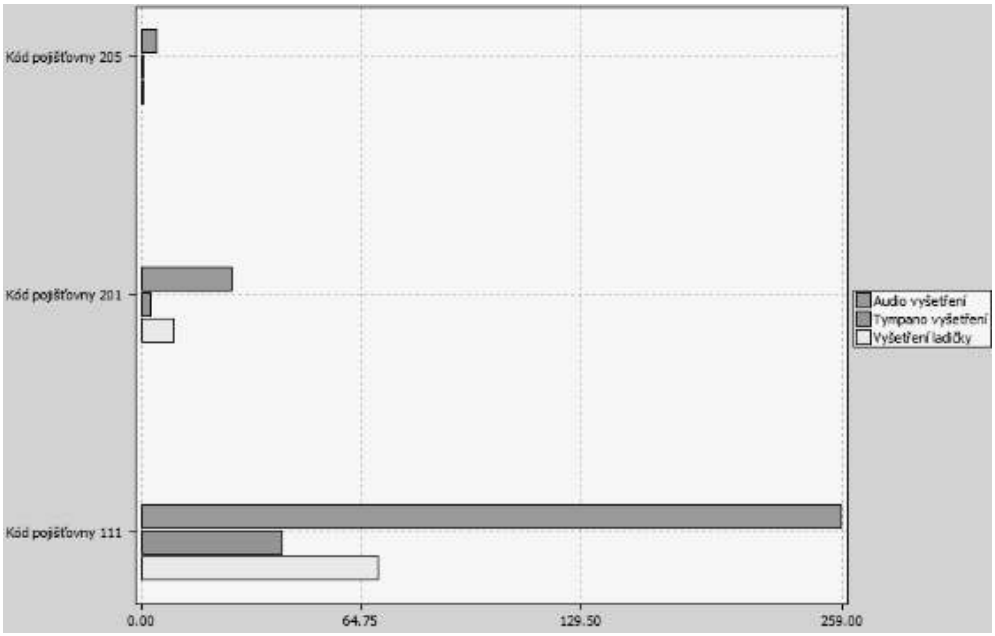
Druh souhrnu: *Přehled provedených vyšetření*
Interpretace: Uvedení informací o všech provedených vyšetřeních pacientů ve tvaru *název vyšetření, datum vyšetření, čas vyšetření, jméno, rod.číslo, pojišťovna, datum narození*.
Výsledky zahrnují *419* údajů, tedy informace o kompletně provedených vyšetření pro jednotlivé pacienty. Výsledky jsou řazeny abecedně dle názvů typů vyšetření.
(neuvádím blíže z důvodu ochrany osobních údajů pacientů)

Druh souhrnu: *Počty provedených vyšetření*
Interpretace: Uvedení počtů provedených vyšetření u jednotlivých jmen pacientů. Jsou uvedeny počty vyšetření pro typy *audio vyšetření, tympano vyšetření a vyšetření ladičky.*
Výsledky zahrnují *292 audio vyšetření, 46 tympano vyšetření, 81 vyšetření ladičky.*
V celkovém počtu *419 vyšetření*.
(neuvádím blíže z důvodu ochrany osobních údajů pacientů)

Druh souhrnu: *Počty vyšetření pojišťoven*
Interpretace: Pro každou pojišťovnu je uveden počet jednotlivých typů vyšetření.
Vyjádření <*nezařazeno*> znamená, že pacientu nebyla přiřazena pojišťovna.

Pojišťovna	Počet audio vyšetření	Počet tympano vyšetření	Počet vyšetření ladičky	Celkem vyšetření
111	259	42	70	371
201	27	3	10	40
205	5	1	1	7
<nezařazeno>	1	0	0	1
Celkem:	292	46	81	419

Tab. 4. Výsledky souhrnu počtů vyšetření pojišťoven



Obr. 44. Grafické znázornění souhrnu počtů vyšetření pojišťoven

Druh souhrnu: *Počty pacientů pojišťoven*

Interpretace: Pro každou pojišťovnu je uveden počet jejích vyšetřených pacientů.

Výsledky: *pojišťovna 111 - 191 pacientů, pojišťovna 201 - 1 pacient, pojišťovna 205 - 5 pacientů.*

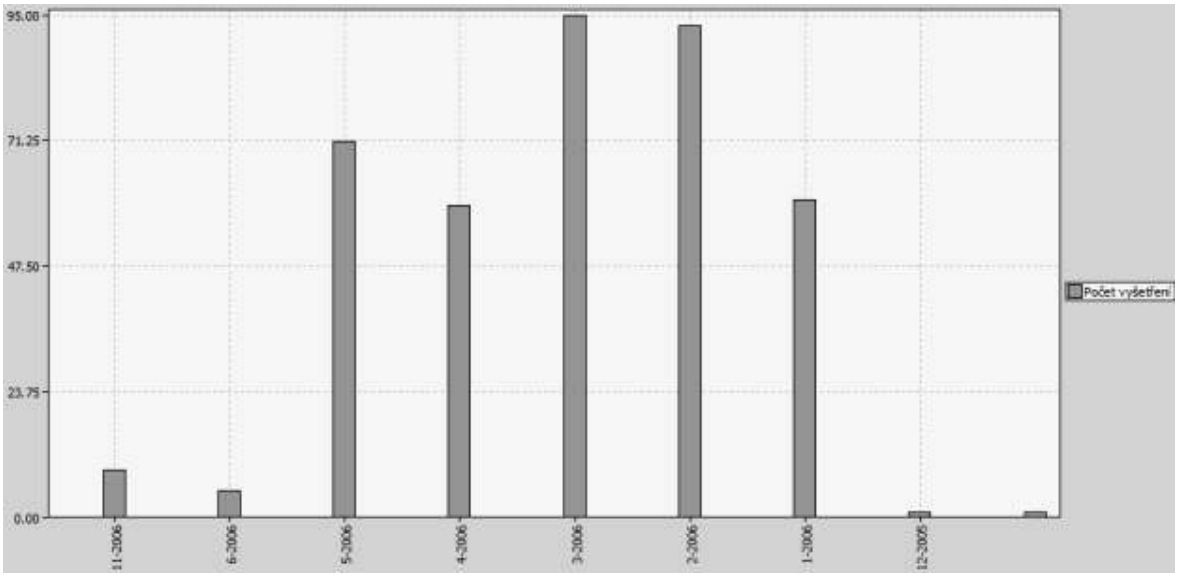
Druh souhrnu: *Roční souhrn vyšetření*

Interpretace: Po jednotlivých ročních obdobích jsou vyhodnoceny počty jednotlivých typů vyšetření. Řazení výsledků je provedeno od nejnovějších směrem k nejstarším.

Rok	Počet audio vyšetření	Počet tympano vyšetření	Počet vyšetření ladičky	Celkem vyšetření
2010	2	0	0	2
2009	2	0	0	2
2007	13	0	3	16
2006	269	46	78	393
2005	4	0	0	4
2004	1	0	0	1
2000	1	0	0	1
Celkem:	292	46	81	419

Tab. 5. Výsledky ročního časového souhrnu vyšetření

Druh souhrnu: *Měsíční souhrn vyšetření*
Interpretace: Po jednotlivých měsíčních obdobích jsou vyhodnoceny počty jednotlivých typů vyšetření. Řazení výsledků je provedeno od nejnovějších směrem k nejstarším.



Obr. 45. Výsledky měsíčního souhrnu počtu provedených vyšetření

Druh souhrnu: *Denní souhrn vyšetření*
Interpretace: Po jednotlivých dnech jsou vyhodnoceny počty jednotlivých typů vyšetření. Řazení výsledků je provedeno od nejnovějších směrem k nejstarším.
Výsledky: Pro zobrazení je výsledků mnoho - jsou shodné formy s tabulkou ročního a grafem měsíčního časového souhrnu. Součty vyšetření pro jednotlivé typy vyšetření jsou - *audio vyšetření 292, tympano vyšetření 46, vyšetření ladičky 81, celkový součet 419*.

K uvedeným vyjádření výsledků statistické analýzy je nutné poznamenat, že každý typ souhrnu zahrnuje jak podobu textovou, tabulkovou, tak i podobu ve formě grafu. Pro měsíční časový souhrn jsem uvedl pouze grafickou podobu interpretace výsledků, pro roční souhrn tabulkovou. Výsledkové tabulky jsem z navrženého programu získal exportem do souboru webové stránky a zde pak uvedl. Ve výsledcích analýz se součty vyšetření a celkové počty shodují, což také dokazuje správnost výsledků vypočítaných systémem. U jednotlivých typů výsledků jsem zdůraznil jejich interpretaci a cíl jejich výpočtů z pohledu hodnocení provozu ORL ordinace.

13 MOŽNOSTI PRAKTICKÉHO NASAZENÍ PROGRAMU

Praktické nasazení vytvořeného programu se nabízí ve formě doplňku k již existujícímu programu *Fowler*, jenž může být používán v lékařských ordinací se specializací ORL. Program vychází z konkrétního způsobu archivace dat vyšetření systému *Fowler*. Přesto je tvořen obecně, a to z pohledu přizpůsobení různé definici typů vyšetření, které program dokáže rozpoznat a potřebuje k další své statistické analýze. Proto lze program použít i pro jiné praktické použití, ale musí být splněn předpoklad podobného způsobu archivace, který je zaveden systémem *Fowler*. Vytvořený program nabízí uživateli obecné možnosti filtrace dat vyšetření s širokou škálou možností provedení statistických analýz i definice zobrazení jejich výsledků. Z těchto důvodů vyplývá, že je program řešen tak, aby splňoval do budoucna všechny přípustné varianty a nebylo potřeba ho významně doupravovat. Pro případné úpravy jsou zdrojové kódy podrobně komentovány a celkově bych proto program označil jako otevřený systém.

Pro běh programu postačuje v základní podobě nastavený operační systém MS Windows XP, popřípadě jeho vyšší verze. Vyžaduje pouze minimální požadavky na programové i technické vybavení počítačové stanice. Program má spustitelný soubor s názvem *program_dp.exe.* Instalace probíhá zkopírováním celého adresáře *program/* z přiloženého CD přílohy P I. Program není doporučeno přímo spouštět z CD, protože se jedná o médium režimu pro čtení a nikoliv pro zápis, což je důležité pro zápis konfiguračního souboru nastavení programu. Po zkopírování adresáře *program*/ na pevný disk je program připraven k provozu. V adresáři se nachází složka *napoveda/*, kde jsou uvedeny praktické ukázky prvního nastavení programu a další práce s ním. Veškerou konfiguraci a nastavení program ukládá do souboru *program_settings.xml*, který vznikne automaticky po první práci s programem. Soubor zahrnuje definici typů vyšetření rozpoznatelných programem a také cestu k adresáři s daty. Pokud například v budoucnu přibude nový typ vyšetření, který má program statisticky zpracovávat, může se provést jeho přidefinování v nastavení popsaném v nápovědě programu. Soubor nastavení *program_settings.xml* může být nabídnut ostatním lékařům k nakopírování do jejich adresářů. Tím se přenese tato konfigurace bez nutnosti dalších nastavování. Nápověda programu je vytvořena ve formátu webové stránky, aby bylo možné její umístění na síť Internet pro případ možné distribuce programu.

ZÁVĚR

Cílem diplomové práce bylo vytvoření systému pro vyhodnocení statistických souhrnů počtu provedených vyšetření v oblasti ORL lékařství. Předpokladem pro tvorbu praktické části diplomové práce bylo nastudování teorie typů vyšetření ORL. Systém měl zadáno vyhodnocovat data vyšetření, která již existují a výsledky jejich statistické analýzy v různých formách zobrazovat. Na počátku jsem provedl zjištění způsobu archivace dat vyšetření programem Fowler, jenž může být používán v lékařských ordinacích specializace ORL. Podle principu archivace dat, požadavků na podobu systému, stanovení druhů statistického vyhodnocení dat a forem zobrazení výsledků, jsem provedl návrh řešení. Podle metodiky návrhu softwarových aplikací jsem naplánoval podobu architektury celého systému s návrhem funkčního provázání jejích jednotlivých oblastí. Stanovil jsem vhodné implementační prostředky a postupy řešení. Zvolil jsem vlastní způsob postupu načítání souborů dat vyšetření, který jsem založil na principu zpracování znaků textového souboru Meallyho konečným automatem. Celý systém jsem realizoval jako softwarovou okenní aplikaci s minimálními provozními i ekonomickými požadavky. Zmapoval jsem a popsal postup instalace a nastavení zvolených programovacích nástrojů a podrobně během programování komentoval zdrojové kódy. Tím jsem zajistil podporu otevřenosti systému do budoucna. Vytvořený systém jsem testoval pomocí vlastního vytvořeného testovacího generátoru dat vyšetření. Využil jsem ho při ověření správnosti výpočtů systému na rozsáhlém objemu vstupních dat. Ve fázi testování jsem určil též vyhodnocení výpočetní složitosti řešení. V praktické části jsem také uvedl přehled jednotlivých funkcí systému a rozbor různých druhů statistické analýzy dat. Pro implementovaný systém jsem vytvořil nápovědu s konkrétními ukázkami práce s programem. V obecném pohledu systém nabízí možnost načíst data vyšetření ordinace ORL, nastavit omezující filtrační podmínky, zvolit druh statistické analýzy a tvar zobrazení výsledků. Řazené výsledky jsou zobrazovány ve formě šestice textových sestav, tabulek a grafů. Výstupní hodnoty lze dále exportovat. Před stanovením možností nasazení systému do praxe jsem ověřil řešení na skutečných datech. Finální systém je vytvořen tak, aby při jeho používání lékaři ORL ordinací našli rozmanité druhy statistických analýz a formy interpretací jejich výsledků. Aplikace nabízí obecně řešené zadávání filtračních podmínek pro data s podporou možného vzniku nových typů vyšetření v budoucnu. Systém je obecným a dynamickým nástrojem použitelným v praxi, a to jako nástroj statistické analýzy provozu lékařských ordinací primárně specializace otorhinolaryngologie.

ZÁVĚR V ANGLIČTINĚ

The thesis aim was to create the system for the time-count summaries evaluation, which is implemented for the examinations in the ORL medical clinics. At first was necessary to study the ORL examination types. According to specifications the system should evaluate exist data and present the statistical results in the different forms. First, the measured data, stored in Fowler application format, were analyzed. The Fowler is the existing application, which can be used in the ORL medical clinics. The data format, system requirements, the selection of the statistic analysis types and results visualization forms, determine the solution design. According to the software development methology was designed the system architecture and the concrete architecture parts. In the thesis are described the acceptable implementation tools, solution methods and my own way of the examination data loading. I designed this concrete way according to the characters processing princip with the Meally determined automat. The whole system is realized as the window application with the minimal operating and economic requirements. The system source codes and the development tools instalation are commented. That is why the system can be opened. The system was testing with implemented test-generator. The generator was used for the statistical results verification, for the small and large input data content. Then it was performed the evaluating of the calculation complexity. In the practical part is stated the list of the system functions and concrete statistical types. For the system was created the help with the concrete examples. The system offers examination data loading, filtration conditions settings, selection of the statistical analysis type and results format definition. The sorted results are visualized in the text reports, tables and graphs. The programm user can export the output values. Before the system practical using definition, were the program results on the real data-set tested. The ORL doctors can find in the system the different statistical analysis types and result interpretation forms. For this reason was the program created. The application contains the general solved filtration definition with the support of the new examination types addition. The system can be used in the practice as the statistical analysis tool for the medical clinics with the specialisation Otorhinolaryngology.

SEZNAM POUŽITÉ LITERATURY

[1] LEJSKA, M., et al. *Základy praktické audiologie a audiometrie*. 1. vyd. Brno: IDVPZ, 1994. 171 s. ISBN 80-7013-178-0.

[2] PATTON, Ron. *Testování softwaru*. 1. vyd. Praha: Computer Press, 2002. 314 s. ISBN 80-7226-636-5.

[3] GUNDERLOY, Mike. *Z kodéra vývojářem: Nástroje a techniky pro opravdové programátory*. 1. vyd. Brno: Computer Press, 2007. 288 s. ISBN 978-80-251-1517-6.

[4] SMART, J., HOCK, K., CSOMOR, S. *Cross-Platform GUI Programming with wxWidgets*. 1st ed. Indiana: Prentice Hall, 2005. 700 s. ISBN 0-13-147381-6.

[5] BLIŽŇÁK, Michal. *Systémové programování*. 1. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2005. 202 s. ISBN 80-7318-364-1.

[6] SMART, J., et al. *WxWidgets 2.8.11: A portable C++ and Python GUI toolkit* [online]. c 1992-2006, last revision February 2010 [cit. 2010-05-01]. <http://docs.wxwidgets.org/stable/>.

[7] VANÍČEK, J., et al. *Teoretické základy informatiky*. 1. vyd. Praha: Kernberg Publishing, 2007. 436 s. ISBN 978-80-903962-4-1.

[8] HAROLD, E. R., MEANS, W. S. *XML v kostce*. 1. vyd. Praha: Computer Press, 2002. 439 s. ISBN 80-7226-712-4.

[9] CASTRO, Elizabeth. *XML pro World Wide Web: Praktická vizuální příručka*. 1. vyd. Brno: SoftPress, 2001. 254 s. ISBN 80-86497-07-0.

[10] MLÝNKOVÁ, I., et al. *XML technologie: Principy a aplikace v praxi*. 1. vyd. Praha: Grada, 2008. 267 s. ISBN 978-80-247-2725-7.

[11] DEMLOVÁ, M., KOUBEK, V. *Algebraická teorie automatů*. 1. vyd. Praha: Státní nakladatelství technické literatury, 1990. 287 s. ISBN 80-03-00348-2.

[12] AJZERMAN, M. A., et al. *Logika, automaty a algoritmy*. 1. vyd. Praha: Academia, 1971. 407 s.

[13] WRÓBLEWSKI, Piotr. *Algoritmy: Datové struktury a programovací techniky*. 1. vyd. Brno: Computer Press, 2004. 352 s. ISBN 80-251-0343-9.

[14] SEDGEWICK, Robert. *Algoritmy v C: Části 1-4., Základy, datové struktury, třídění, vyhledávání*. 1. vyd. Praha: SoftPress, 2003. 688 s. ISBN 80-86497-56-9.

[15] *Fowler* [počítačový program a nápověda programu]. Ver. 3.01. [Česká republika], 2009 [citováno 2010-05-07]. Program pro ordinace ORL lékařství.

[16] *WxPack* [online]. c2010, last revision 18th of February 2010 [cit. 2010-05-05]. <http://wxpack.sourceforge.net/>.

[17] *WxCode* [online]. c2010, last revision March 2010 [cit. 2010-05-05]. <http://wxcode.sourceforge.net/>.

[18] *WxCode - wxFreeChart* [online]. c2010, last revision 15th of February 2010 [cit. 2010-05-05]. <http://wxcode.sourceforge.net/components/freechart/>.

[19] *Code::Blocks* [online]. c2009 [cit. 2010-05-05]. <http://www.codeblocks.org/>.

[20] *MinGW - Minimalist GNU for Windows* [online]. c2009 [cit. 2010-05-05]. <http://www.mingw.org/>.

[21] *WxFormBuilder* [online]. c2009, last revision 11th of June 2009 [cit. 2010-05-05]. <http://www.wxformbuilder.org/>.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

ORL	Otorhinolaryngologie
SISI	Short Increment Sensitivy Index
XML	Extensible Markup Language
HTML	Hypertext Markup Language
CSV	Comma Separated Values
MinGW	Minimalist GNU for Windows

SEZNAM OBRÁZKŮ

Obr. 1. Ukázka použití audiogramu pro grafické znázornění sluchových prahů [1] 13

Obr. 2. Ukázka systému Fowler se zkouškou vložení dat fiktivního pacienta [15].....15

Obr. 3. Ukázka diagramu ze systému Fowler pro fiktivního testovacího pacienta [15].....15

Obr. 4. Postup vývoje softwarového produktu dle vodopádového modelu [2] 17

Obr. 5. Náročnost nápravy chyby v různých fázích tvorby produktu [2] 19

Obr. 6. Ukázka kroku instalace knihovny wxWidgets instalačním balíčkem wxPack [16] 23

Obr. 7. Konfigurace knihovny wxWidgets v editoru CodeBlocks [19].....24

Obr. 8. Nastavení kódování znaků na UTF-8 pro editor CodeBlocks [19].....24

Obr. 9. Ukázka návrhu designu okna programu nástrojem wxFormBuilder [21] 25

Obr. 10. Přehled nečastěji používaných typů mřížek polohového uspořádání prvků [4] 28

Obr. 11. Ukázka rámcového okna s programovým menu a panelem nástrojů 29

Obr. 12. Ukázka a schématický popis rámcového okna s prvkem AUI notebook.....30

Obr. 13. Příklad standardního dialogu reprezentovaného třídou wxFileDialog 31

Obr. 14. Dialog s ukázkou klasických ovládacích a vizuálních prvků [4].....32

Obr. 15. Dialog s ukázkou ovládacích prvků klasická tabulka a tabulka vlastností [4] 33

Obr. 16. Princip a struktura dynamické kolekce objektů [4][5].....34

Obr. 17. Ukázka ovládacího prvku wxFreeChart sloužícího pro zobrazení grafů [18] 36

Obr. 18. Znázornění principu tvorby dokumentů XML na příkladu z oblasti fyziky.....38

Obr. 19. Ukázka seznamu pacientů v systému Fowler s daty fiktivního pacienta [15] 39

Obr. 20. Ukázka souboru výsledků vyšetření fiktivního pacienta v systému Fowler [15] ..40

Obr. 21. Vyjádření návrhu Meallyho konečného automatu stavovým diagramem [11].....42

Obr. 22. Schéma postupu řazení textových dat metodou třídění spojováním.....43

Obr. 23. Schéma architektury vnitřní části navrženého systému 48

Obr. 24. Schéma začlenění navrhovaného systému do vnějšího okolí 48

Obr. 25. Návrhu řešení načítání XML dokumentů v oblasti Data XML Hierarchy 50

Obr. 26. Obecně navržený postup filtrace dat dle podmínek uživatele 51

Obr. 27. Schéma principu statistického vyhodnocení dat a vizualizace výsledků.....53

Obr. 28. Návrh datové struktury oblasti Data XML Hierarchy pro práci s XML daty.....54

Obr. 29. Návrh datové struktury oblasti návrhu Data Operation Manager.....55

Obr. 30. Začlenění navrhovaného testovacího generátoru do procesu testování.....56

Obr. 31. Nastavení podmínek překladu projektu - v editoru CodeBlocks [19] 59

Obr. 32. Implementovaný testovací generátor pro ověření správnosti výpočtů systému 64

Obr. 33. Příklad nekorektně a korektně určeného adresáře s daty vyšetření ORL 65

Obr. 34. Základní skladba výsledného systému.....66

Obr. 35. Přiklad přidání nového typu vyšetření v budoucnu 68

Obr. 36. Ukázka programu s měsíčním souhrnem dat ORL.....72

Obr. 37. Ukázka programu s přehledem počtů pacientů pojišťoven.....72

Obr. 38. Ukázka programu s vyobrazením výsledků textovou sestavou 73

Obr. 39. Ukázka exportu výsledků analýzy pro případné další zpracování.....73

Obr. 40. Graf naměřené časové složitosti operace načtení dat pacientů a vyšetření 76

Obr. 41. Graf naměřené časové složitosti vytvoření a řazení výsledků textové sestavy 76

Obr. 42. Graf naměřené časové složitosti vytvoření a řazení výsledků tabulky.....77

Obr. 43. Graf naměřené časové složitosti statistických výpočtů a zobrazení výsledků.....77

Obr. 44. Grafické znázornění souhrnu počtů vyšetření pojišťoven 81

Obr. 45. Výsledky měsíčního souhrnu počtu provedených vyšetření.....82

SEZNAM TABULEK

Tab. 1. Ukázky zdrojového kódu pro práci s dynamickou kolekcí objektů [4][5] 35

Tab. 2. Vyjádření návrhu Meallyho konečného automatu ve formě tabulky [11][12] 42

Tab. 3. Testování správnosti výsledků systému pomocí testovacího generátoru 74

Tab. 4. Výsledky souhrnu počtů vyšetření pojišťoven.....80

Tab. 5. Výsledky ročního časového souhrnu vyšetření.....81

SEZNAM PŘÍLOH

P I CD disk se spustitelným programem, zdrojovými kódy

P II Struktura adresářů na přenositelné příloze P I

PŘÍLOHA P II: STRUKTURA ADRESÁŘŮ NA PŘENOSITELNÉ PŘÍLOZE P I

Obsah hlavního adresáře disku CD:

- DiplomovaPrace.doc - diplomová práce ve formátu MS Word
- DiplomovaPrace.pdf - diplomová práce ve formátu Pdf
- program/ - adresář s program, určený ke zkopírování z disku CD na pevný disk
- zdrojove_kody/- adresář tvorby programu, obsahující kódy a nástroje pro jejich překlad

Obsah adresáře program/:

- Navod_instalace.txt - návod k instalaci programu
- program_dp.exe - spustitelný program (po zkopírování adresáře program/ na pevný disk)
- insuranceList.txt - soubor s nastavením informativního seznamu pojišťoven pro program
- napoveda/ - soubory nápovědy programu – se spustitelným souborem index.htm

Obsah adresáře zdrojove_kody /:

- kody/ - adresář se všemi zdrojovými kódy
- programovaci_nastroje/ - adresář s instalacemi programovacích nástrojů psaní kódů

Obsah adresáře zdrojove_kody /kody /:

- kody_programu/ - adresář se zdrojovými kódy programu
- kody_testovaciho_generatoru/ - adresář s kódy testovacího generátoru pro program

Obsah adresáře zdrojove_kody /programovaci_nastroje /:

- instalační soubory knihovny wxWidgets (wxPack_v2.8.5.01.exe), jejích doplňků (doplnkyWxWidgets.zip), překladač (mingw.zip) a editor kódu (CodeBlocks.zip)