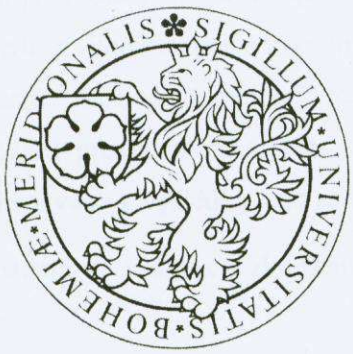
Jihočeská univerzita v Českých Budějovicích

Pedagogická fakulta, katedra informatiky

Obor: Výpočetní technika a informatika



SYSTÉM TVORBY A ZKOUŠENÍ TESTŮ S VYUŽITÍM XML

Praktická bakalářská práce

Autor: Josef Kyrian

Vedoucí práce: Ing. Miroslav Máče, Ph.D

České Budějovice, 2006

Anotace

Obsahem absolventské práce je problematika efektivity zkoušení. Jde o návrh systému, jež umožňuje pomocí XML technologie vymezit jakoukoliv oblast testů, definovaných jako relaci otázek a odpovědí. V rámci absolventské práce byl proveden návrh a implementace systému, který umožňuje jak náhodné generování testů, včetně možnosti zařazování dříve špatně zodpovězených otázek, tak i provádění finálních, již nadefinovaných testů. Provádění testů je zaznamenáváno a výsledky jsou statisticky vyhodnocovány.

Příkladem užití systému je realizace zkoušení z pravidel provozu na pozemních komunikacích.

Abstract

This dissertation work is dedicated to problems of learning. The main goal is to design a system which allows, using XML technology, to specify any set of tests defined as relation of questions and answers. The System can generate random tests including adding previously wrong answered questions as well as performing exactly defined final tests. All finished tests are stored and results are evaluated to get summary statistics.

Example of usage is created program intended for learing final tests of driving school.

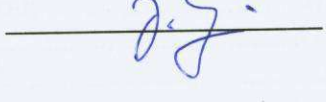
Poděkování

Děkuji všem, kteří mi jakýmkoliv způsobem pomohli dokončit tuto bakalářskou práci a to hlavně Ing. Miroslavu Máčemu, Ph.D. za odborné vedení během mé práce.

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně, a že jsem veškerou použitou literaturu uvedl v Seznamu použité literatury.

Josef Kyrian



Obsah

1 Úvod

Obsahem absolventské práce je problematika efektivity zkoušení. Jde o návrh systému, jež umožňuje pomocí XML technologie vymezit jakoukoliv oblast testů, definovaných jako relaci otázek a odpovědí. V rámci absolventské práce byl proveden návrh a implementace systému, který umožňuje jak náhodné generování testů, včetně možnosti zařazování dříve špatně zodpovězených otázek, tak i provádění finálních, již nadefinovaných testů. Provádění testů je zaznamenáváno a výsledky jsou statisticky vyhodnocovány.

Příkladem užití systému je realizace zkoušení z pravidel provozu na pozemních komunikacích. Z vlastní zkušenosti a mých kamarádů z okolí, kteří namísto učebnice použili k naučení tento program, musím uvést, že systém zkoušení je natolik účinný a otázky jsou vybírány takovým způsobem, že kupříkladu naučení celých pravidel silničního provozu, se stalo otázkou několika hodin až dní. Následné složení finálního testu naostro se stává formalitou. Pro žáka se stává nezábavné drilové bezmyšlenkové učení interaktivní hrou, která je vede k lepším výsledkům.

2 Technologie XML

Definice testů, otázek a data programu musí být uchovány v nějaké univerzální struktuře. Jednou možností bylo nadefinovat si takovou strukturu vlastní s vlastními pravidly a způsoby zpracování. Druhou, lepší, možností bylo se podívat po již vytvořeném standardu, který je univerzální, snadno se s ním pracuje, je moderní natolik, že není problém sehnat skupinu nástrojů pro práci s ním, je nelicencovaný, platformově nezávislý a široce rozšířený. Standard vyhovující těmto podmínkám je bezesporu XML. Zkratka XML znamená eXtensible Markup Language, česky rozšiřitelný značkovací jazyk. Byl vyvinut konsorciem W3C (World Wide Web Consortium) jako pokračování jazyka HTML a SGML.

2.1 Historie

Vývoj XML začal v roce 1996 a stal se doporučením W3C od ledna 1998, což může budit dojem, že jde o nevyspělou technologii. Ve skutečnosti tato technologie není zcela nová. Před XML existovalo SGML, vyvíjené v osmdesátých letech, standard ISO od roku 1986, široce používaný pro rozsáhlé dokumentace. Vývoj HTML začal v roce 1990. Tvůrci XML převzali to nejlepší ze SGML podle zkušenosti s HTML, a vytvořili něco stejně mocného jako SGML, ale jednoduššího na používání. A je třeba říci, že zatímco SGML je určeno spíše pro technickou dokumentaci než data jiného druhu, u XML je tomu naopak. Zkratka SGML je, díky své složitosti, pro mnohé spíše než „Standard Generalized Markup Language", „Sounds Great, Maybe Later" („Zní to dobře, možná později").

2.2 Použití XML

Jazyk je určen především pro výměnu dat mezi aplikacemi a pro publikování dokumentů. XML hned od samého počátku myslel na potřeby i jiných jazyků než je angličtina. Jako znaková sada se implicitně používá ISO 10646 (také Unicode). V XML proto můžeme vytvářet dokumenty, které obsahují texty v mnoha jazycích najednou – můžeme přepínat mezi různými jazyky v jednom dokumentu. Současně je přípustné i jiné libovolné kódování (např. windows-1250, iso-8859-2), musí však být v každém dokumentu přesně určeno. Odpadají tak problémy s konverzí z jednoho kódování do druhého.

Podstatnou novinkou jazyka XML je logické uspořádání dat. Využívá se hlavně tam, kde je třeba elektronicky zpracovávat data či hledat nějaké informace. Svou univerzalitou a jednoduchostí se ale stává mocným nástrojem.

2.3 Syntaxe XML

Nechci zde podrobně popisovat jazyk XML, ani vysvětlovat co jsou elementy, atributy, entity, DTD, od toho jsou jiné publikace. Uvedu zde jen pár pravidel XML a krátký příklad použití.

XML dokument je text, obvykle Unicode kódovaný jako UTF-8, ale jsou přípustná i jiná kódování.

Na rozdíl od např. HTML, efektivita XML je silně závislá na struktuře, obsahu a integritě. Aby byl dokument považován za správně strukturovaný "well-formed", musí mít nejméně následující vlastnosti:

- Musí mít právě jeden kořenový (root) element.
- Neprázdné elementy musí být ohraničeny startovacím a ukončovacím tagem. Prázdné elementy mohou být označeny tagem "prázdný element".
- Všechny hodnoty atributů musí být uzavřeny v uvozovkách - jednoduchých (') nebo dvojitých ("), ale jednoduchá uvozovka musí

být uzavřena jednoduchou a dvojitá dvojitou. Opačný pár uvozovek může být použit uvnitř hodnot.

- Tagy mohou být vnořeny, ale nemohou se překrývat, to znamená, že každý (ne kořenový) element musí být celý obsažen v jiném elementu.

Jména elementů v XML rozlišují malá a velká písmena: např. <Example> a </Example> je pár, který vyhovuje správně strukturovanému dokumentu, pár <Example> a </example> je chybný.

Jednoduchý recept na upečení chleba v XML jako příklad by mohl vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" ?>

<recipe>
  <title>Basic bread</title>
  <ingredient amount="3" unit="cups" >Flour</ingredient>
  <ingredient amount="0.25" unit="ounce" >Yeast</ingredient>
  <ingredient amount="1.5" unit="cups" >Warm Water</ingredient>
  <ingredient amount="1" unit="teaspoon" >Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together, and knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again, place in a tin, and then bake in the oven.</step>
  </instructions>
</recipe>
```

V příkladu jsou použity elementy *receipe, title, ingredient, instructions* a *step*. Význam jednotlivých elementů je následující:

- ***receipe*** – kmenový element zaobalující zbylé child elementy. V atributech má popsáno o jaký recept jde, kolik minut trvá příprava a jak dlouho trvá vlastní proces vaření
- ***title*** – definuje název receptu
- ***ingredient*** – definuje ingredienci k přípravě chleba. V atributech má popsáno jednotku a množství dané ingredience

- ***instructions*** – rodičovský element obsahující child elementy *step*
- ***step*** – definuje krok (postup) přípravy jídla

2.4 Nástroje XML

Výhodou XML je, že jde o prostý text a tak jej lze editovat v jakémkoliv textovém editoru. Pro zpracování programem je však nutný tzv. parser. Parser načte daný dokument a uloží jej do vhodných datových struktur. S takovými strukturami se pak pracuje snáz, díky implementaci mnoha funkcí, které zpřístupňují procházení jednotlivých xml elementů, načítání a ukládání atributů, vyhledávání, vytváření nových elementů, přesuny, úpravu a mazání elementů starých.

Na internetu najdete jistě mnoho XML parserů pro použití v jazyce C++. Stačí zadat do Googlu „C++ XML parser" a vybrat si z 2 a půl miliónů odkazů. Některé knihovny (parsery) jsou placené a některé zdarma. Liší se množstvím implementovaných funkcí a možnostmi práce s XML dokumentem. V mé aplikaci jsem použil neveřejný XML parser od Ondřeje Žižky. Jde o velmi propracovaný nástroj, umožňující snadnou práci s daty.

Pro zkompilování je potřeba zařadit složku s hlavičkovými soubory mezi includes a do projektu, kde chcete tento parser použít, přidat lib knihovnu. Pak už je použití následující:

```
xDoc doc;       // objekt s xml parserem
xElm *xroot;    // element rootu

// rozparsovat XML
if (doc.ParseXMLText(szXmlData) != XE_NONE) {
  // načíst top element
  if (xroot = doc.GetTopElement() != NULL) {
    // ...
  }
}
```

V příkladu je nejprve rozparsován XML text a pak z něj následně načten root element. Následná práce už probíhá s tímto kmenovým elementem.

3 Způsob ukládání dat

K ukládání veškerých dat programu jsem vytvořit a použil knihovnu FilesWrapper. Tato knihovna opět využívá k ukládání dat XML. Vše ukládá do jednoho souboru, v kterém simuluje systém souborů a implementuje funkce pro práci s ním. V systému složek a souborů může být uloženo v podstatě cokoliv s jakoukoliv velikostí. Data jsou úmyslně uchovávána v jednom souboru s daným souborovým systémem, jelikož to umožňuje využít práva na čtení resp. zápis a zamezuje zneužiti resp. zkopírování dat. Navíc to také usnadňuje systém zacházení s datovými soubory, kde jsou uložena všechna data související s daným testem. Takto se může použít test pouhým zkopírováním jednoho souboru a navíc s možností použití prav a znemožnění užití testu neoprávněným osobám bez licence.

Fakt, že systém balíčkovacího souboru je uložen do XML, poskytuje neomezenou flexibilitu s možností rozšíření do budoucích verzí, jako např. čtecí a zapisovací práva, použití podle aktivačního klíče, systém uživatelů a hesel, kódování a dekódování dat apod.

3.1 Formát souboru knihovny FilesWrapper

Většina datových souborů je rozdělena na dvě části - hlavičku a samotná data. Hlavička je pevná struktura, podle které se řídí práce se samotnými daty. U souborů, které používá knihovna FilesWrapper, tomu není jinak.

Hlavička se skládá ze čtyřech částí. Těmi jsou indikátor FilesWrapper souboru, verze souboru, velikost hlavičky a data popisující systém souborů.

Indikátor je neměnný a je vždy 2-bytový, obsahující písmena „FW“. Při načítání takovéhoto souboru je vždy nejprve zkontrolováno, zda-li soubor tento indikátor v prvních dvou bytech obsahuje. Pokud tomu tak není, práce se

souborem končí a je vrácena chyba, že soubor má neplatný formát nebo byl poškozen.

Další části hlavičky je verze souboru. Ta je uložena ve 3 bytech a je samozřejmě proměnlivá v závislosti na verzi souboru v které byl uložen. Ukládání verze je nutné pro přípravu budoucích verzí, aby knihovna věděla jak má s takovýmto souborem zacházet. První verze je označena jako „V01".

Třetí částí hlavičky je pole, v němž je uložena velikost hlavičky. Pole je 4-bytové a velikost je zde uložena binárně.

Poslední částí jsou data popisující systém souborů. Tato data jsou reprezentována pomocí XML. Maximální velikost těchto dat = délka hlavičky - (délka indikátoru + délka pole s verzí souboru + délka pole s velikostí hlavičky). Systém souborů, uložený pomocí XML, je na začátku tohoto pole s daty a zbytek je doplněn binárními nulami.

XML dokument, popisující systém souborů, může vypadat kupříkladu následovně:

```
<?xml version="1.0" encoding="windows-1250" ?>

<fileswrapper>
  <filesystem>
    <file name="info.xml" location_pos="37802" location_length="100"/>
    <directory name="Značky">
      <file name="A 11.bmp" location_pos="40438" location_length="2086"/>
      <file name="A 12.bmp" location_pos="42524" location_length="2030"/>
    </directory>
    <directory name="Křižovatky">
      <file name="01.jpg" location_pos="173556" location_length="16505"/>
      <file name="03.jpg" location_pos="206977" location_length="15453"/>
    </directory>
    <file name="questions.xml" location_pos="658273" location_length="402078"/>
    <file name="logo.bmp" location_pos="782" location_length="36414"/>
    <file name="tests.xml" location_pos="1060351" location_length="7113"/>
  </filesystem>
</fileswrapper>
```

Dokument obsahuje kmenový element *fileswrapper,* a jeho child elementy *filesystem, file* a *directory*. Elementy *fileswrapper* a *filesystem* nemají žádné

atributy a jen zaobalují elementy zbylé. Element file a directory jsou prvky systému souborů a jejich význam je následující:

- ***file*** – popisuje soubor s názvem daným v atributu *name*, na dané pozici určené atributem *location_pos* o dané velikosti atributem *location_length*. Cesta k souboru je dána, umístěním tohoto elementu v rámci elementu *filesystem*.
- ***directory*** – popisuje složku souborů s názvem daným v atributu *name*. Cesta ke složce je dána, umístěním tohoto elementu v rámci elementu *filesystem*.

3.2 Práce s balíčkovacím souborem

Práce s balíčkovacím souborem knihovny FilesWrapper probíhá pomocí instance třídy `CFWFile`, implementující všechny potřebné funkce. Vytvoření instance probíhá následovně:

```
CFWFile *fw = new CFWFile(nazev_balickovaciho_souboru);
```

Knihovna zkontroluje, zda-li jde o platný soubor a načte daný systém souborů. Práce se soubory pak probíhá pomocí funkcí `FileOpen`, `FileRead`, `FileWrite`, `FileClose`.

Takto vypadá krátká ukázka čtení ze souboru:

```
CFWWrappedFile *wf = fw->FileOpen("info.xml");

if (wf != NULL) {
  char dataR[2] = {'\0', '\0'};

  while (fw->FileRead(wf, dataR, 1)) {
    OutputDebugString(dataR);
  }
  fw->FileClose(wf);
}
```

V příkladu je otevřen soubor „info.xml“ a následně je po bytech načítán a vypisován do debug okna. Po vypsání celého souboru je pak zavřen.

Ukázka zápisu do balíčkovacího souboru je následující:

```
char *szFileName = "Značky/test.txt";
CFWWrappedFile *wf = fw->FileOpen(szFileName);

if (wf == NULL) {
  fw->CreateFSElement(szFileName, CFWFile::_FS_ELEMENT::FILE);
  wf = fw->FileOpen(szFileName);
}

if (wf != NULL) {
  char *dataW = "zkušební data k zápisu";

  fw->FileWrite(wf, dataW, (DWORD)strlen(dataW));
  SetEndOfFile(*wf);
  fw->FileClose(wf);
}
```

V příkladu je otevřen soubor „test.txt“ umístěný ve složce „Značky“. Pokud soubor nelze otevřít, je použita funkce `CreateFSElement` pro vytvoření souboru nového. Pokud se již soubor podaří otevřít, je do něj zapsán zkušební řetězec a soubor je ukončen a zavřen.

Dalšími důležitými funkcemi knihovny jsou `RenameFSElement` a `DeleteFSElement`. Jak už sám název napovídá, jsou to funkce na přejmenování a smazání daného prvku systému souborů. Myslím, že uvádět příklad není potřeba.

Neméně důležitou je funkce pro výpis souborů resp. složek v adresáři. Jmenuje se `ListDirectory`. Její použití je opět intuitivní a tak myslím, že příklad není důležitý.

3.3 Jak to vlastně funguje

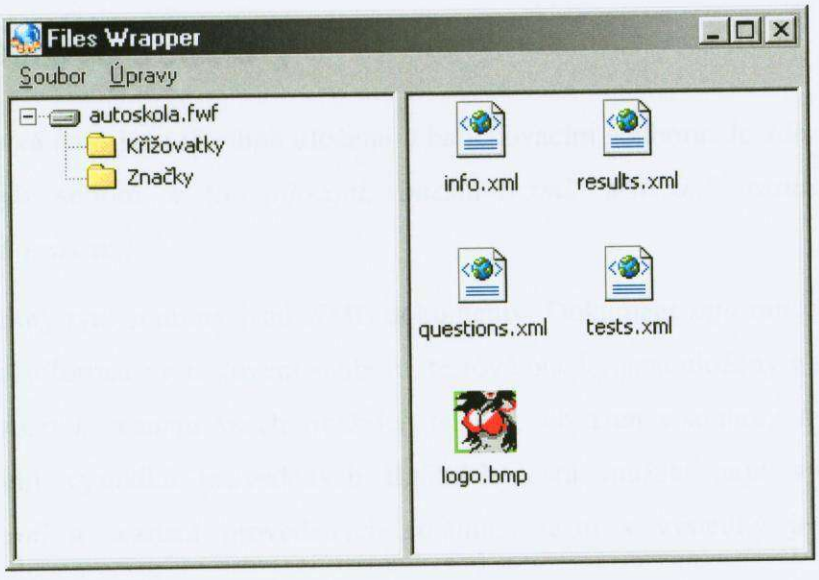
Otevřením zabaleného souboru (neplést s balíčkovacím souborem), se vytvoří dočasný soubor, kam se obsah daného souboru zkopíruje. Z něj pak lze číst, popř. do něj zapisovat. Při zavírání souboru je kontrolováno, zda-li nebyl pozměněn. Pokud bylo do souboru něco zapsáno a je tím změněn, je obsah dočasného souboru načten a uložen zpět do balíčkovacího souboru. Pozice kam se soubor uloží je hledána funkcí `GetDesiredLocationPos`, jejímž parametrem je ukazatel na daný soubor a požadovaná velikost. Algoritmus najde nejvhodnější pozici, kam lze soubor umístit, s tím že se snaží, co nejlépe využít volné místo v souboru.

Problém nastává v momentě, kdy je nějaký soubor smazán. Mazáním souborů se vytvářejí v balíčkovacím souboru prázdná místa, která jsou nevyužita. Pro odstranění hluchých míst slouží funkce `Optimize`, která přezapíše všechny soubory za sebe, čímž prázdná místa odstraní.

3.4 Uživatelské rozhraní

Pro snadnou práci s balíčkovacími soubory byl vytvořen jednoduchý program FilesWrapper_UI. Aplikace poskytuje možnost otevřít takovýto soubor a následně s ním pracovat. V menu *Úpravy* naleznete funkce pro práci se soubory resp. složkami, jejich přidávání, mazání a přejmenovávání a v neposlední řadě zde naleznete funkci na optimalizaci balíčkovacího souboru.

Následující obrázek č. 1 zobrazuje jak vypadá uživatelské rozhraní programu.



obrázek č. 1 – Uživatelské rozhraní programu FilesWrapper_UI

4 Testové soubory

Testová data jsou všechna uložena v balíčkovacím souboru. Je zde několik klíčových soborů a to: *info.xml, questions.xml, tests.xml, resuts.xml* a *finished_tests.xml.*

Všechny tyto soubory jsou XML dokumenty. Dokument *info.xml* obsahuje základní informace o testovém souboru, testové otázky jsou uloženy v souboru *questions.xml,* seznam všech finálních testů je obsažen v souboru *tests.xml,* hodnocení výsledků provedených finálních testů můžete najít v souboru *results.xml* a seznam provedených finálních testů s výsledky je uložen v *finished_tests.xml.*

4.1 info.xml

Tento soubor obsahuje základní informace o testovém souboru. Jeho obsah může vypadat následovně:

```
<?xml version="1.0" encoding="windows-1250" ?>

<gibotests>
        <name>Autoškola</name>
</gibotests>
```

Kmenovým elementem je *gibotests*, stejně jako u všech ostatních dokumentů. Jediným child elementem je zde element *name*, definující název testového komplexu. V našem případě „Autoškola".

4.2 questions.xml

V tomto dokumentu jsou uloženy všechny testové otázky včetně bodového ohodnocení roztříděné do jednotlivých sekcí. Takovýto dokument může vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" ?>

<gibotests>
  <section id="0" name="Pravidla provozu">
    <questions num="2">
      <question id="3" type="check" picture="" points="3">
        <text>Účastníkem provozu na pozemních komunikacích je:</text>
        <option correct="true">vozka s potahovým vozidlem</option>
        <option correct="true">řidič motorového vozidla</option>
        <option correct="true">průvodce vedených nebo hnaných zvířat</option>
        <option correct="true">osoba, která tlačí sáňky</option>
      </question>

      <question id="120" type="check" picture="" points="3">
        <text>Při použití sněhových řetězů na vozidle smí jet řidič:</text>
        <option correct="false">nejvyšší dovolenou rychlostí</option>
        <option correct="false">rychlostí nejvýše 80 km/h</option>
        <option correct="true">rychlostí nejvýše 50 km/h</option>
        <option correct="false">rychlostí nejvýše 30 km/h</option>
      </question>
    </questions>
  </section>

  <section id="4" name="Dopravní značky">
    <questions num="">
      <question id="1" type="check" picture="$DATA:Značky/P 1.bmp" points="2">
        <text>Tato dopravní značka patří do skupiny:</text>
        <option correct="true">upravující přednost</option>
        <option correct="false">výstražných</option>
        <option correct="false">dodatkové tabulky</option>
        <option correct="false">příkazových</option>
      </question>
    </questions>
  </section>
</gibotests>
```

Kmenovým elementem je *gibotests*, stejně jako u všech ostatních dokumentů. Význam ostatních jednotlivých elementů je následující:

- ***section*** – definuje sekci otázek. V případě autoškoly jsou to například „Pravidla provozu“, „Zásady první pomoci“, „Dopravní značky“, atd. Identifikátor sekce je určen atributem *id* a je unikátní. Název sekce je uložen v atributu *name*.
- ***questions*** – definuje element se sadou otázek v dané sekci. V atributu *num* obsahuje informaci o počtu otázek.
- ***question*** – tento element definuje jednu konkrétní testovou otázku. Identifikátor otázky je určen atributem *id* a je v rámci dané sekce

unikátní.

Dalším atributem je atribut s názvem *type* určující, zda-li jde o otázku s jednou možnou odpovědí nebo s možnostmi více odpovědí. Pokud je možná správná pouze jedna odpověď, typ je *radio*, pokud je na otázku možných odpovědí více, typ je *check*. K testové otázce může být doprovodný obrázek. Cesta k takovému obrázku je uložena v atributu *picture*. Pokud cesta začíná textem `$DATA:` jde o cestu v balíčkovacím souboru s daty. Posledním parametrem otázky je počet bodů, které testovaný obdrží po správné odpovědi. Tento parametr je uložen v atributu *points*.

- ***text*** – tento element obsahuje text testové otázky. Element nemá žádné atributy.
- ***option*** - tento element definuje jednu z odpovědí na testovou otázku a obsahuje text odpovědi. Parametrem této možnosti odpovědi je, zda-li je to odpověď správná. Tento parametr je uložen v atributu *correct* a nabývá hodnoty *true* resp. *false*, a to v případě, že odpověď je správná resp. není správná.

4.3 tests.xml

V tomto dokumentu je uložen seznam všech finálních testů. Takovýto dokument může vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" ?>

<gibotests>
  <test id="0" name="Ukázkový test" result_type="group_am_a1_a_b_c_d">
    <questions num="2">
      <section type="id" id="0">
        <question type="n" n="2" />
        <question type="n" n="3" />
      </section>
    </questions>

    <questions num="4">
```

```
      <section type="n" n="2">
        <question type="rnd" />
      </section>
    </questions>

    <questions num="3">
      <section type="n" n="3">
        <question type="range" from="0" to="50" />
      </section>
    </questions>
  </test>

  <test id="1" name="Krátký test" result_type="group_am_al_a_b_c_d">
    <questions num="1">
      <section type="rnd">
        <question type="rnd" />
      </section>
    </questions>
  </test>
</gibotests>
```

Kmenovým elementem je *gibotests*, stejně jako u všech ostatních dokumentů. Význam ostatních jednotlivých elementů je následující:

- ***test*** – element definuje jeden z finálních testů. V parametrech testu je uložen identifikátor testu a to v atributu *id*, název testu v atributu *name*, a skupinu podle které je hodnocen výsledek testu, je v atributu *result_type*.
- ***questions*** – tento element obsahuje sadu podmínek, podle kterých se má testová otázka při generování vybrat. V atributu *num* je definováno kolik otázek se má podle těchto daných podmínek vygenerovat.
- ***section*** – element definuje podmínku, podle které se má vybrat sekce, ze které bude vybrána otázka. Atributy tohoto elementu závisí na atributu *type*, který označuje, o jaký typ podmínky jde. Type může nabývat těchto hodnot:
  - ***id*** – v případě, že jde o tento typ podmínky, je určeno v atributu *id*, že bude vybrána sekce s tímto identifikátorem
  - ***n*** - v případě, že jde o tento typ podmínky, je určeno v atributu *n*, že bude vybrána sekce s tímto číslem v pořadí

- ***range*** – v případě, že jde o tento typ podmínky, je určeno v atributech *from* a *to*, že bude vybrána sekce s číslem v pořadí v tomto rozmezí (včetně)
- ***rnd*** - v případě, že jde o tento typ podmínky, bude vybrána náhodná sekce

- ***question*** - element definuje podmínku, podle které se má vybrat testová otázka. Atributy tohoto elementu závisí na atributu *type*, který označuje, o jaký typ podmínky jde. Type může nabývat těchto hodnot:
  - ***id*** – v případě, že jde o tento typ podmínky, je určeno v atributu *id*, že bude vybrána otázka s tímto identifikátorem
  - ***n*** - v případě, že jde o tento typ podmínky, je určeno v atributu *n*, že bude vybrána otázka s tímto číslem v pořadí
  - ***range*** – v případě, že jde o tento typ podmínky, je určeno v atributech *from* a *to*, že bude vybrána otázka s číslem v pořadí v tomto rozmezí (včetně)
  - ***rnd*** - v případě, že jde o tento typ podmínky, bude vybrána náhodná otázka

4.4 results.xml

V tomto dokumentu je uložen seznam skupin a podmínek hodnocení, podle kterých bude hodnocen výsledek finálního testu. Takovýto dokument může vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" ?>

<gibotests>
  <result id="3" type="group_am_al_a_b_c_d" name="Skupina B">
    <condition type="max_time" max_time="1800" />
```

```
    <condition type="min_points" min_points="56" />
    <condition type="min_section_points" section_id="5" min_points="12" />
  </result>

  <result id="4" type="group_am_a1_a_b_c_d" name="Skupina C">
    <condition type="max_time" max_time="1800" />
    <condition type="min_points" min_points="59" />
    <condition type="min_section_points" section_id="5" min_points="12" />
  </result>

  <result id="5" type="group_test" name="Ukázková skupina hodnocení">
    <condition type="min_points" min_points="100" />
    <condition type="question" section_id="1" question_id="5" />
    <condition type="question" section_id="1" question_id="6" />
  </result>
</gibotests>
```

Kmenovým elementem je *gibotests*, stejně jako u všech ostatních dokumentů. Význam ostatních jednotlivých elementů je následující:

- ***result*** – element definuje jedno z hodnocení výsledku. V parametrech je uložen identifikátor testu a to v atributu *id*, název skupiny hodnocení v atributu *name*, a skupinu podle které je hodnocen výsledek testu, je v atributu *result_type*.
- ***condition*** – element definuje podmínku, podle které se má hodnotit výsledek testu. Atributy tohoto elementu závisí na atributu type, který označuje, o jaký typ podmínky jde. Type může nabývat těchto hodnot:
  - ***max_time*** – v případě, že jde o tento typ podmínky, je určeno v atributu *id*, že bude vybrána otázka s tímto identifikátorem
  - ***min_points*** - v případě, že jde o tento typ podmínky, je určeno v atributu *n*, že bude vybrána otázka s tímto číslem v pořadí
  - ***min_section_points*** – v případě, že jde o tento typ podmínky, je určeno v atributech *from* a *to*, že bude vybrána otázka s číslem v pořadí v tomto rozmezí (včetně)
  - ***question*** - v případě, že jde o tento typ podmínky, bude vybrána náhodná otázka

4.5 finished_tests.xml

V tomto dokumentu je uložen seznam všech provedených finálních testů s výsledky. Data z toho souboru pak jsou použita pro vyhodnocení statistiky a také pro možnost zlepšení efektivity zkoušení ve zkušebním testu, kde si zkoušený může zvolit přimíchávání dříve špatně zodpovězených otázek. Takovýto dokument může vypadat takto:

```
<?xml version="1.0" encoding="windows-1250" ?>

<gibotests>
  <test time="1143662610" passed="0" points="42" max_points="65">
    <question section_id="0" question_id="33" />
    <question section_id="0" question_id="380" />
    <question section_id="1" question_id="13" />
    <question section_id="3" question_id="39" />
    <question section_id="5" question_id="7" />
  </test>

  <test time="1143661533" passed="1" points="63" max_points="65">
    <question section_id="3" question_id="40" />
  </test>
</gibotests>
```

Kmenovým elementem je *gibotests*, stejně jako u všech ostatních dokumentů. Význam ostatních jednotlivých elementů je následující:

- ***test*** – element definuje jeden z provedených finálních testů. V parametrech je uložen čas vyhodnocení testu a to v atributu *time*, zda-li byl proveden úspěšně v atributu *passed*, počet získaných bodů v atributu *points* a maximální možný počet bodů je uložen v atributu *max_points*.
- ***question*** – element definuje špatně zodpovězenou otázku. Atributy tohoto elementu jsou *section_id* a *question_id*, určující sekci a v ní konkrétní testovou otázku.

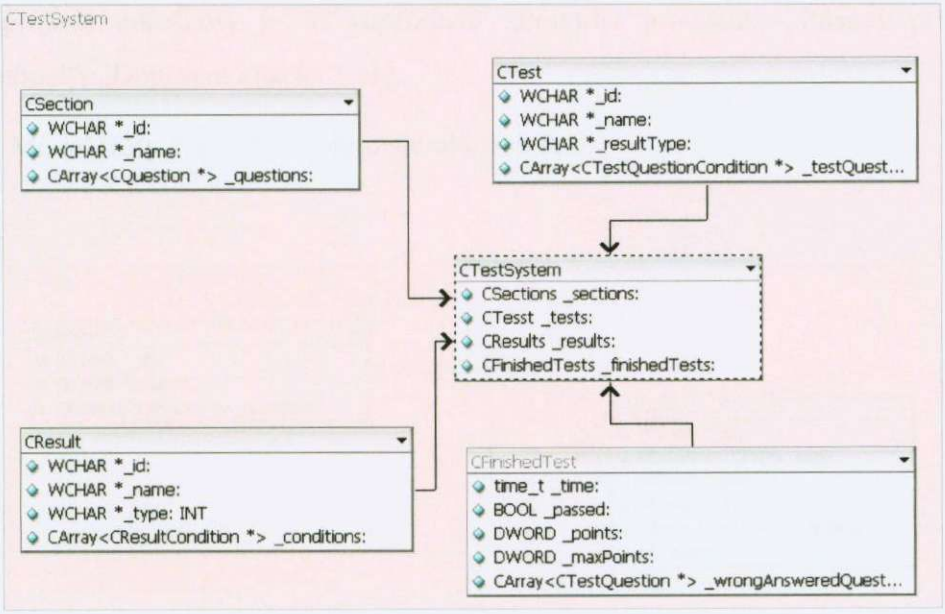
5 Jádro systému

Jádro systému tvoří několik tříd, implementujících potřebné funkce, pro správu, ovládání a uchování dat v systému.

5.1 CTestSystem

Základní rodičovská třída se jmenuje `CTestSystem` a zprostředkovává přístup do jednotlivých částí testového systému. Sama o sobě načítá jednotlivé klíčové xml dokumenty a díky xml parseru naplní systém patřičnými daty. Klíčovými soubory jsou, jak již bylo uvedeno výše, *info.xml, questions.xml, tests.xml, results.xml* a *finished_tests.xml.* Tyto soubory obsahují informace o testovém souboru, testové otázky, finální testy a hodnocení výsledků.

Data systému jsou pak uložena v dalších třídách uložených v instanci této třídy. Třída obsahuje 4 základní objekty a to `CSections`, `CTests`, `CResults` a `CFinishedTests`. Veškerá data jsou uložena právě v těchto objektech. Přístup k těmto objektům je realizován pomocí patřičných funkcí. Následná práce už pak probíhá pouze s těmito objekty. Model třídy `CTestSystem` zobrazuje obrázek č. 2.



obrázek č. 2 – Model třídy `CTestSystem`

5.2 Testové otázky

Testové otázky jsou strukturovaně uloženy v rámci několika tříd. Těmi jsou `CSection`, `CQuestion` a `COption`.

Data, která používají tyto třídy, jsou načtena z dokumentu *questions.xml*. To zajišťuje nadřazená třída `CTestSystem`.

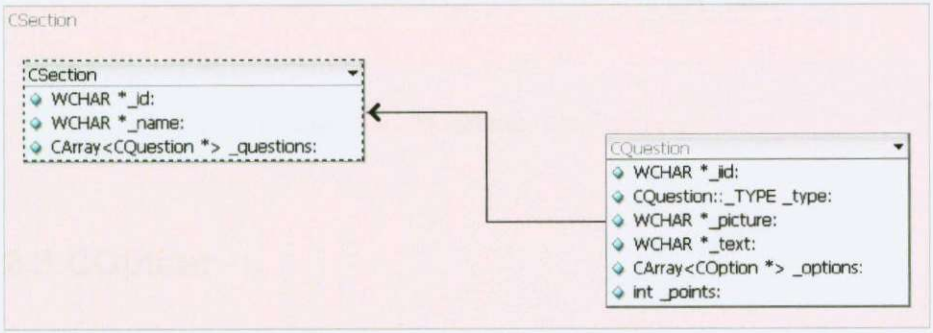
5.2.1 CSection

Tato třída reprezentuje sekci otázek a ukládá všechny testové otázky z dané sekce. Jednotlivé testové otázky jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CQuestion`.

Sekce je definována svým identifikátorem a jménem. Identifikátor je v rámci všech sekcí otázek unikátní. Jméno sekce může být libovolné.

V případě autoškoly je to kupříkladu „Pravidla provozu“, „Zásady první pomoci“, „Dopravní značky“, atd.

Model třídy `CSection` zobrazuje obrázek č. 3.

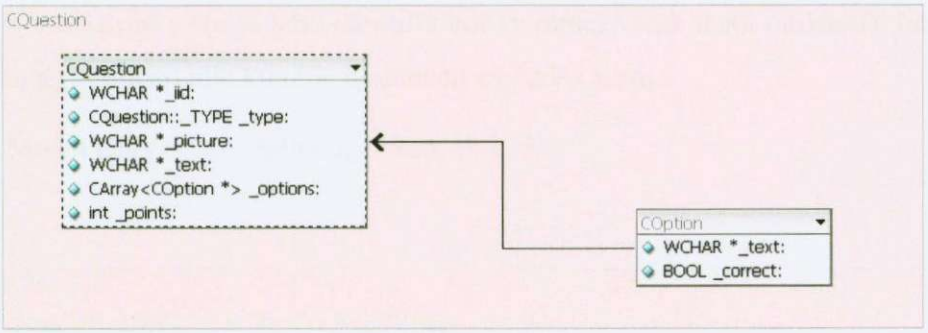


obrázek č. 3 – Model třídy `CSection`

5.2.2 CQuestion

Tato třída reprezentuje testovou otázku. Vlastnostmi otázky je její identifikátor (v rámci dané sekce otázek unikátní), cesta k obrázku, text otázky, seznam možných odpovědí a počet získaných bodů za správně zodpovězenou otázku. Jednotlivé odpovědi na otázku jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `COption`.

Model třídy `CQuestion` zobrazuje obrázek č. 4.



obrázek č. 4 – Model třídy `CQuestion`

5.2.3 COption

Tato třída reprezentuje jednu z odpovědí na testovou otázku. Vlastnostmi odpovědi je text odpovědi a zda-li je tato odpověď správná.

5.3 Finální testy

Finální testy jsou, stejně jako testové otázky, strukturovaně uloženy v rámci několika tříd. U finálních testů jsou těmito třídami `CTest`, `CTestQuestionCondition`, `CTestQuestionConditionSection` a `CTestQuestionConditionQuestion`.

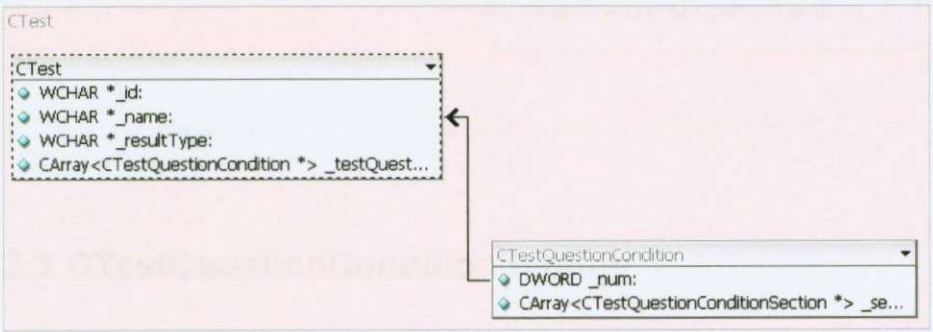
Data, která používají tyto třídy, jsou načtena z dokumentu *tests.xml*. To zajišťuje nadřazená třída `CTestSystem`.

5.3.1 CTest

Tato třída reprezentuje finální test resp. sadu podmínek pro výběr testových otázek při generování finálního testu. Jednotlivé podmínky jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CTestQuestionCondition`.

Vlastnostmi testu je jeho identifikátor (v rámci všech testů unikátní), jméno testu a skupina podle které je hodnocen výsledek testu.

Model třídy `CTest` zobrazuje obrázek č. 5.



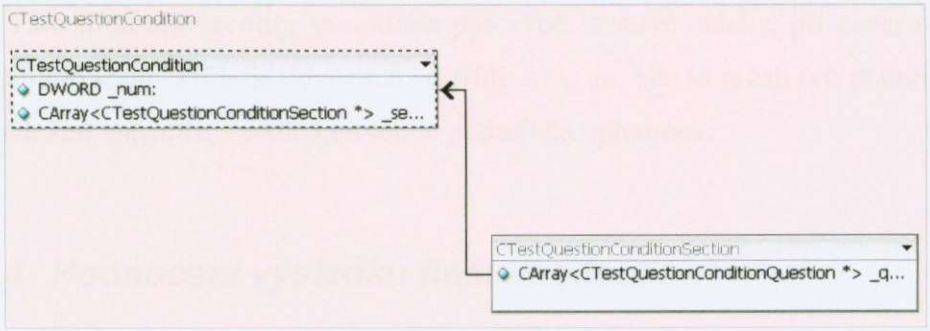
obrázek č. 5 – Model třídy `CTest`

5.3.2 CTestQuestionCondition

Tato třída reprezentuje sadu podmínek, podle kterých je generován finální test. Jednotlivé podmínky jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CTestQuestionConditionSection`.

V této sadě podmínek je jeden parametr, určující kolik otázek se má vygenerovat podle těchto podmínek.

Model třídy `CTestQuestionCondition` zobrazuje obrázek č. 6.

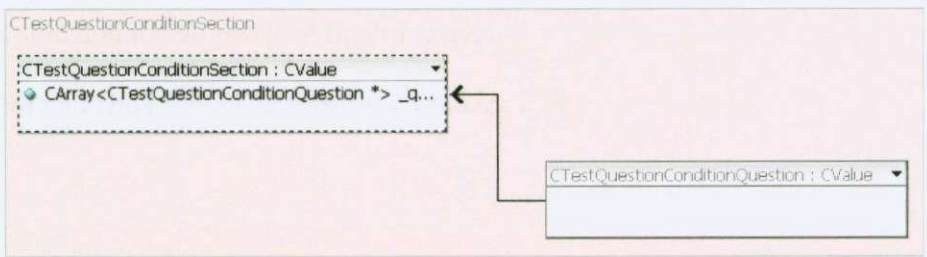


obrázek č. 6 – Model třídy `CTestQuestionCondition`

5.3.3 CTestQuestionConditionSection

Tato třída reprezentuje podmínku pro výběr sekce, při generování finálního testu. Třída je odvozena od třídy `CValue`, kde je určen typ podmínky a náležité vlastnosti, podle kterých je podmínka uplatněna. Uvnitř třídy jsou ještě uloženy kritéria pro výběr konkrétních otázek. Jednotlivé podmínky pro výběr otázek jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CTestQuestionConditionQuestion`.

Model třídy `CTestQuestionConditionSection` zobrazuje obrázek č. 7.



obrázek č. 7 – Model třídy `CTestQuestionConditionSection`

5.3.4 CTestQuestionConditionQuestion

Tato třída reprezentuje podmínku pro výběr testové otázky, při generování finálního testu. Třída je odvozena od třídy `CValue`, kde je určen typ podmínky a náležité vlastnosti, podle kterých je podmínka uplatněna.

5.4 Hodnocení výsledku finálního testu

Podmínky hodnocení výsledků finálních testů jsou strukturovaně uloženy v rámci několika tříd. Těmi jsou `CResult` a `CResultCondition`.

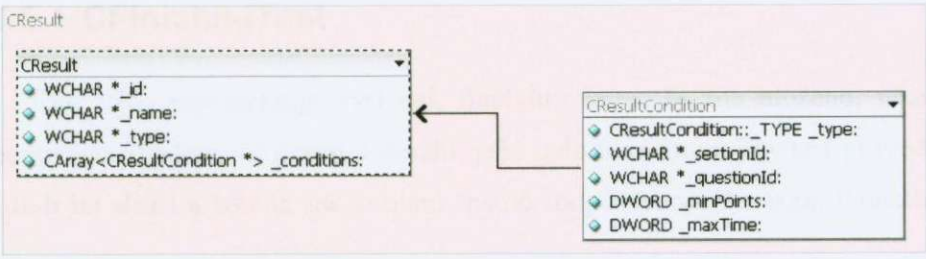
Data, která používají tyto třídy, jsou načtena z dokumentu *results.xml*. To zajišťuje nadřazená třída `CTestSystem`.

5.4.1 CResult

Tato třída reprezentuje sadu podmínek pro hodnocení výsledku finálního testu. Jednotlivé podmínky jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CResultCondition`.

Třída uchovává několik vlastností a to identifikátor (v rámci všech hodnocení unikátní), název hodnocení (v případě autoškoly kupříkladu „Řidičské oprávnění skup. B“) a typ, podle kterého jsou pak nabízeny k dané skupině hodnocení patřičné finální testy.

Model třídy `CResult` zobrazuje obrázek č. 8.



obrázek č. 8 – Model třídy `CResult`

5.4.2 CResultCondition

Tato třída reprezentuje podmínku pro hodnocení výsledku finálního testu. Třída uchovává patřičné vlastnosti, podle kterých je podmínka uplatněna. Těmi jsou minimální počet bodů z celého testu, popř. z jednotlivých sekcí, čas na test, povinně správně zodpovězené otázky atd.

5.5 Výsledky finálních testů

Výsledky finálních testů jsou strukturovaně uloženy v rámci několika tříd. Těmi jsou `CFinishedTest` a `CTestQuestion`.

Data, která používají tyto třídy, jsou načtena z dokumentu *finished_tests.xml*. To zajišťuje nadřazená třída `CTestSystem`. V tomto dokumentu je uložen seznam všech provedených finálních testů s výsledky. Data z toho souboru pak jsou použita pro vyhodnocení statistiky a také pro možnost zlepšení efektivity zkoušení ve zkušebním testu, kde si zkoušený může zvolit přimíchávání dříve špatně zodpovězených otázek.

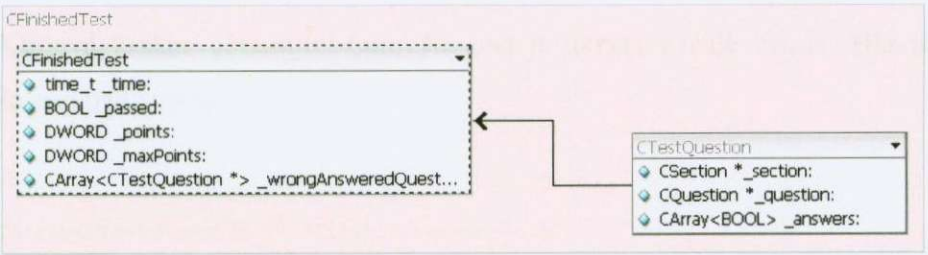
Po provedení testu naostro je výsledek testu automaticky přidán do seznamu vyhodnocených a po ukončení programu je soubor *finished_tests.xml* aktualizován.

5.5.1 CFinishedTest

Tato třída reprezentuje výsledek finálního testu. Je zde uloženo, jakého bodového ohodnocení uživatel dosáhl, jaké bylo maximum, kdy test provedl a zda-li jej složil a také je zde seznam špatně zodpovězených otázek. Jednotlivé

špatné zodpovězené otázky jsou uloženy v poli (resp. třídě `CArray` simulující pole) instancí třídy `CTestQuestion`.

Model třídy `CFinishedTest` zobrazuje obrázek č. 9.



obrázek č. 9 – Model třídy `CFinishedTest`

5.5.2 CTestQuestion

Tato třída reprezentuje testovou otázku. Je zde uloženo z jaké sekce byla tato otázka vybrána a o jakou konkrétní otázku jde. Dále je zde uložen seznam odpovědí na tuto otázku.

6 Generování testů

Systém používá ke generování testů seznam testových otázek, sadu podmínek pro výběr otázek a pro zvýšení efektivity učení také seznam dříve špatně zodpovězených otázek.

Klíčová funkce generující takovýto test je ukryta v třídě `CTest`. Hlavička funkce vypadá takto:

```
CArray<CTestQuestion *> *CTest::GenerateTest(
        CSections *sections,
        CArray<CTestQuestion *> *wrongAnsweredQuestions,
        DWORD numAddedWrongAnsweredQuestions
)
```

Prvním parametrem funkce je odkaz na instanci třídy `CSections` obsahující všechny sekce s otázkami, ze kterých může být test vygenerován. Dalším parametrem je pole s dříve špatně zodpovězenými testovými otázkami. Tento parametr může být NULL, v případě že do testu nebudou žádné takové otázky přimíchány. Posledním parametrem je počet špatně zodpovězených otázek, které se mají do testu přimíchat.

Funkce vrací pole s testovými otázkami, které mají v sobě uloženo, z které sekce byly vybrány a o jaké konkrétní otázky jde. Tyto testové otázky jsou pak servírovány uživateli a on na ně odpovídá.

V následujících kapitolách jsou popsány kroky, jak proces generování testu probíhá.

6.1 Aplikování podmínek výběru otázek

V instanci třídy `CTest`, pomocí které se generuje test je uložen seznam podmínek pro výběr otázek. Tento seznam je postupně procházen a pomocí funkce `GetAllPossibleQuestions` je získáván seznam všech otázek odpovídající dané podmínce. Tento seznam je pak filtrován, aby se některé otázky neopakovali dvakrát.

6.2 Filtrování seznamu odpovídajících otázek

Po získání seznamu všech odpovídajících otázek musejí být profiltrovány, aby se nestalo, že v testu bude vybrána víckrát stejná otázka. Filtrování probíhá ve dvou fázích, kdy se nejdřív profiltruje seznam odpovídajících otázek a pak se ještě porovná se seznamem otázek již vybraných do testu. Tento profiltrovaný seznam postupuje do dalšího kroku generování.

6.3 Náhodný výběr otázky

Ze seznamu všech profiltrovaných odpovídajících otázek je pak náhodně vybrán patřičný počet otázek.

6.4 Přimíchání špatně zodpovězených otázek

Pokud je pole se špatně zodpovězenými otázkami neNULLové a počet otázek k přimíchání je větší než nula, tak je ze seznamu náhodně vybrán patřičný počet otázek a k finálnímu seznamu testových otázek přidán.

7 Uživatelské rozhraní aplikace

Po spuštění aplikace je načten dokument *list.xml*, v němž je uložen seznam všech testový souborů. Je zde uveden název a cesta k náležitému balíčkovacímu souboru. Poté je zobrazen dialog, kde si uživatel vybere, z jakého okruhu testů chce být zkoušen (Autoškola, Zbrojní průkaz, apod.).

Dialog je vyobrazen na obrázku č. 10.

obrázek č. 10 – Dialog výběru testového okruhu

Po zvolení daného druhu testu je otevřen patřičný balíčkovací soubor. Z něj jsou načteny a zpracovány klíčové soubory *info.xml, questions.xml, tests.xml, resuts.xml* a *finished_tests.xml*. Pokud vše proběhne bez chyby je systém naplněn daty a uživateli je zobrazeno hlavní okno sloužící jako rozcestník programu.

7.1 Hlavní okno programu

Před zobrazením okna je z datového souboru načteno logo testového souboru. Logo je uloženo v souboru *logo.bmp* a může mít jakoukoliv velikost. Layout okna se přizpůsobí podle jeho velikosti.

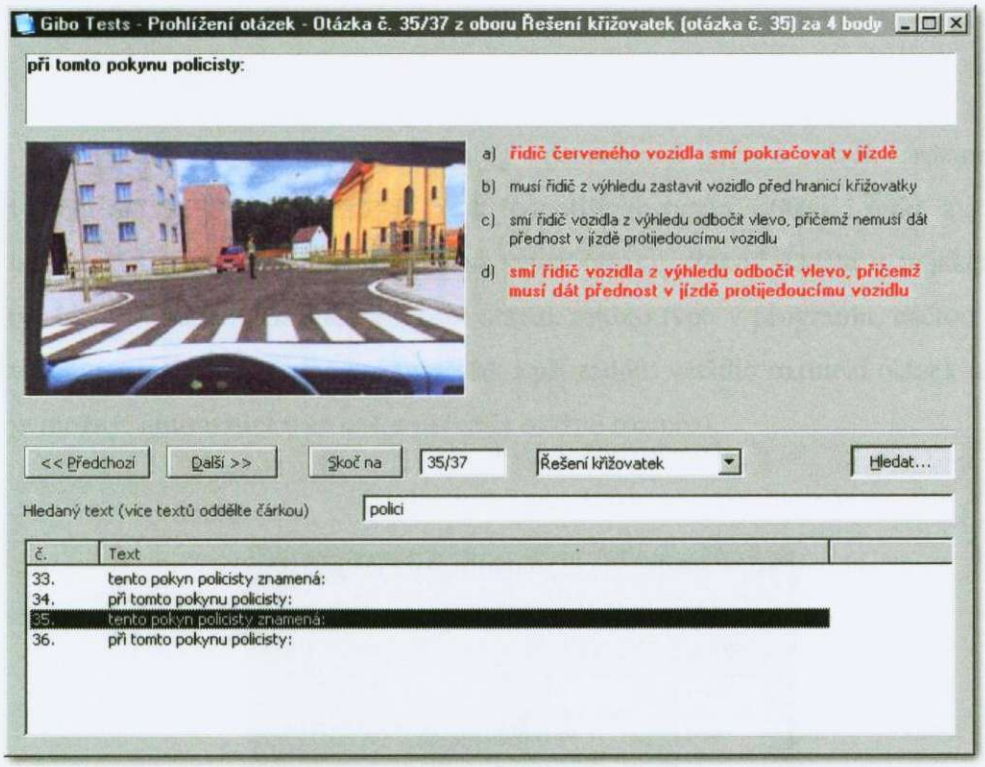
Hlavní okno slouží jako rozcestník programu. Uprostřed okna je logo aktuálního testu a kolem něj je rozmístěno 6 tlačítek zpřístupňujících základní funkce programu. Zde si uživatel může vybrat prohlížení všech testových otázek, zkusit si udělat zkušební test na nečisto, udělat si test naostro nebo zobrazit statistiku, popř. nápovědu. Program lze ukončit stiskem tlačítka „Konec".

obrázek č. 11 – Hlavní okno programu

7.2 Prohlížení otázek

V tomto dialogu máte zpřístupněny všechny otázky rozdělené do jednotlivých sekcí. Otázky lze procházet postupně, přesunout se na danou otázku nebo je vyhledávat podle zadaného textu. Po zadání hledaného textu do textového pole se naplní seznam, umístěný v dolní části, otázkami vyhovujícími hledaným kritériím. Kliknutím na danou otázku v seznamu se zobrazí.

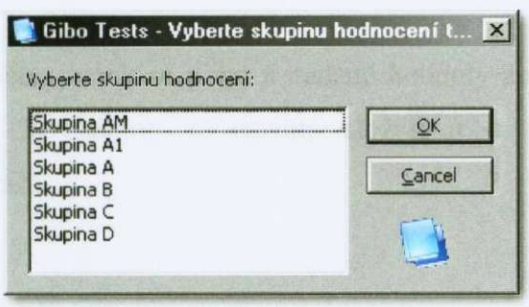
V horní části okna je zobrazen text otázky, pod ním je v levé části doprovodný obrázek a vedle něj jsou možnosti odpovědí. Správná odpověď je vždy označena barevně.



obrázek č. 12 – Prohlížení otázek

7.3 Zkušební test

Po zmáčknutí tlačítka pro spuštění zkušebního testu se spustí dialog s výběrem skupiny, podle které budete hodnoceni. Např. v případě autoškoly je na výběr, jaký typ řidičského oprávnění chcete.



obrázek č. 13 – Výběr skupiny hodnocení

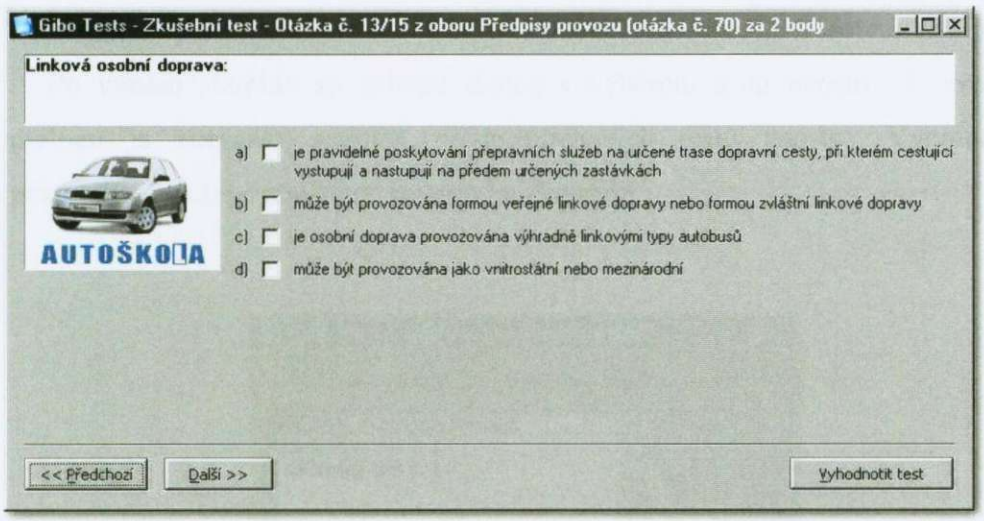
Po výběru skupiny se zobrazí dialog s nastavením počtu a otázek a rozmezí výběru otázek v testu. Nastavit lze k jednotlivým typům otázek i nulu, avšak test musí celkově obsahovat alespoň jednu otázku. Pokud zadáte u nějakého typu počet otázek větší než je počet otázek tohoto typu v programu, načtou se všechny otázky tohoto typu. Stejně tak i při zadání většího rozmezí otázek než je možný, automaticky se načte největší možné rozmezí.

obrázek č. 13 – Nastavení vlastností zkušebního testu

Po nastavení počtu otázek se potvrdí zadané hodnoty zmáčknutím tlačítka „OK“ a zobrazí se dialog s testem složeným z daných náhodně vygenerovaných otázek.

V horní části okna je zobrazen text otázky, pod ním je v levé části doprovodný obrázek a vedle něj jsou možnosti odpovědí. U možností odpovědí je možnost jejích výběru.

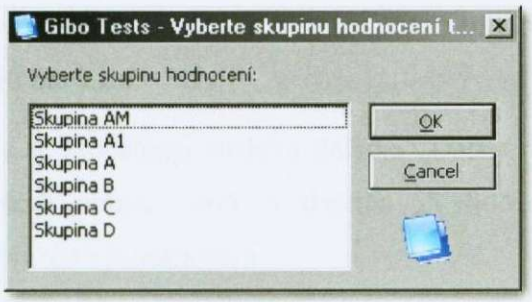
Jednotlivé otázky se procházejí stiskem tlačítka „Další" resp. „Předchozí". Po zodpovězení všech otázek, stiskem tlačítka „Vyhodnotit test" je test vyhodnocen a je zobrazen výsledek testu.



obrázek č. 14 – Zkušební test

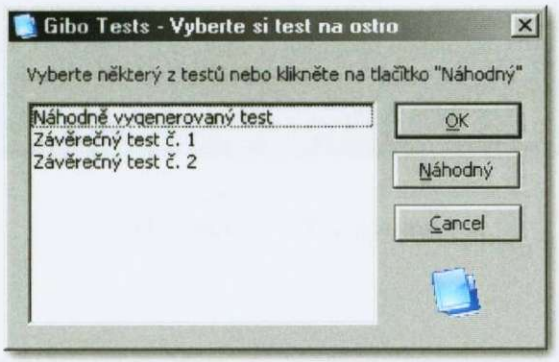
7.4 Test naostro

Po zmáčknutí tlačítka pro spuštění testu naostro se spustí dialog s výběrem skupiny, podle které budete hodnoceni. Např. v případě autoškoly je na výběr, jaký typ řidičského oprávnění chcete.



obrázek č. 15 – Výběr skupiny hodnocení

Po výběru skupiny se zobrazí dialog s výběrem testu naostro. V tomto dialogu je zobrazen seznam předpřipravených testů naostro. Vybráním některého z nich se přesunete rovnou ke zkoušení.



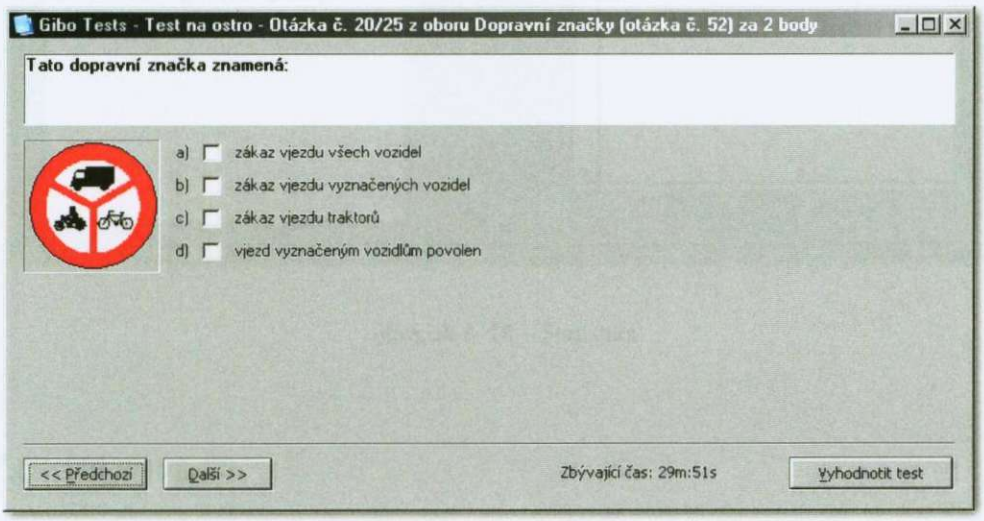
obrázek č. 16 – Výběr testu naostro

Po výběru finálního testu se již zobrazí dialog s testem složený z daných otázek. Tento dialog slouží k provedení daného testu.

V horní části okna je zobrazen text otázky, pod ním je v levé části doprovodný obrázek a vedle něj jsou možnosti odpovědí. U možností odpovědí je možnost jejích výběru.

Pokud je test časově omezen, tak v dolní části dialogu je zobrazeno odpočítávání zbývajícího času.

Jednotlivé otázky se procházejí stiskem tlačítka „Další" resp. „Předchozí". Po zodpovězení všech otázek, stiskem tlačítka „Vyhodnotit test" je test vyhodnocen a je zobrazen výsledek testu.



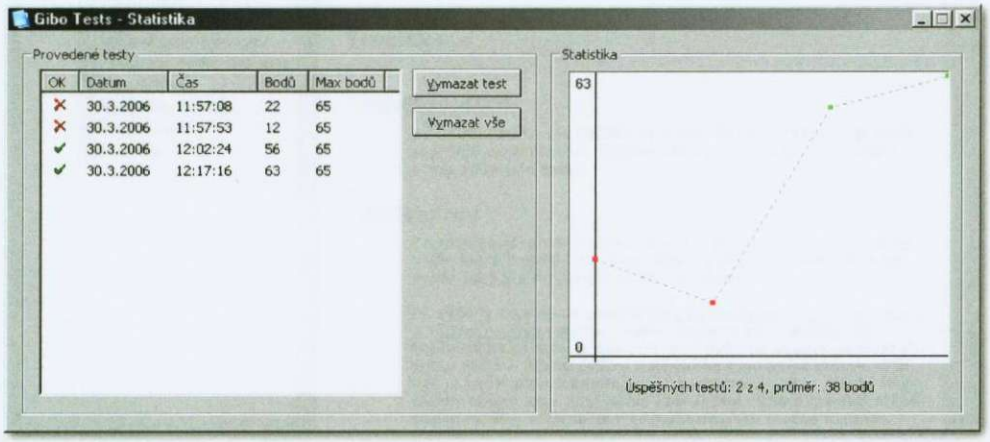
obrázek č. 17 – Test naostro

7.5 Statistika

Po zmáčknutí tlačítka pro spuštění statistiky se zobrazí okno se seznamem všech vyhodnocených testů naostro. Pokud test naostro ukončíte dříve, než jej vyhodnotíte, tak se do seznamu nepřidá. U každého testu je zobrazen čas ukončení, počet dosažených bodů, maximální počet bodů a zda-li byl test úspěšně složen (to je signalizováno obrázkem se zeleným zaškrtnutím a neúspěšné červeným křížkem).

V pravé části okna je zobrazen graf s úspěšností jednotlivých testů a pod ním statistické vyhodnocení.

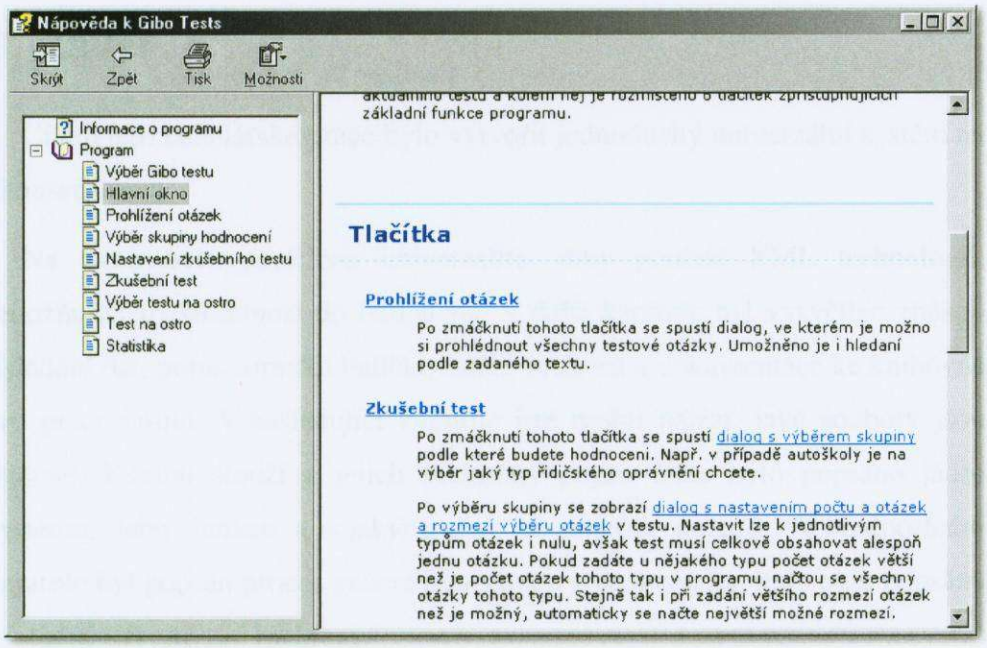
Vybrané záznamy o provedených testech lze vymazat tlačítkem „Vymazat test", popř. tlačítkem „Vymazat vše" pro vymazání všech statistických dat.



obrázek č. 18 – Statistika

7.6 Nápověda

Pokud se uživatel cítí být ztracen a neví co má udělat, může si kdekoliv nechat zobrazit nápovědu a to stiskem tlačítka F1. Po stisku tlačítka se zobrazí patřičná nápověda pro daný dialog, v kterém se nachází.



obrázek č. 19 – Nápověda

8 Závěr

Cílem této bakalářské práce bylo vytvořit jednoduchý univerzální systém na zkoušení testů.

Na úvod byl vysvětlena univerzalita, díky použité XML technologii, umožňující rozšiřitelnost do budoucna. V další kapitole byl vysvětlen způsob ukládání dat, popis formátu balíčkovacích souborů a dokumentace ke knihovně pro práci s nimi. V následující kapitole jste mohli nalézt, jaké soubory jsou klíčové, k čemu slouží a jejich podrobný popis. Dále bylo popsáno jádro systému, jeho funkce a s jakými daty a objekty pracuje. V předposlední kapitole byl popsán proces generování testů a v poslední kapitole bylo popsáno výsledné uživatelské rozhraní.

Myslím, že výsledkem práce je plně použitelný testový systém, což je doloženo realizací zkoušení z pravidel provozu na pozemních komunikacích.

Seznam použité literatury:

[1] HAROLD, E. R. a MEANS, W. S. XML v kostce. Brno : Computer Press a.s., 2002. ISBN 8072267124

[2] CHALUPA, R. 1001 tipů a triků pro Visual C++. Brno : Computer Press a.s., 2003. ISBN 8072268422

[3] PRATA, S. Mistrovství v C++. Brno : Computer Press a.s., 2004. ISBN 8025100987

Ostatní zdroje:

[4] Wikipedie, otevřená encyklopedie. Dostupné na World Wide Web: < http://cs.wikipedia.org >

[5] World Wide Web Consortium. Dostupné na World Wide Web:

< http://www.w3.org >