TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N 2612 – Elektrotechnika a informatika

Studijní obor: 1802T007 – Informační technologie

Vytvoření nástroje pro tvorbu scénářů

Creation Tool for Production Scenarios

Diplomová práce

Autor: **Bc. Alena Mlejnková**

Vedoucí práce: Ing. Igor Kopetchke

Konzultant: Michal Brejtr (Red Fox)

V Liberci 1. 5. 2008

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé diplomové práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum

Podpis

Poděkování

Chtěla bych poděkovat Ing. Igoru Kopetchkemu za profesionální vedení diplomové práce. Druhým člověkem, kterému taktéž děkuji za jeho čas a konzultace, je Michal Brejtr.

Anotace

V první části práce je vysvětlen pojem autorský nástroj a stručně popsán eLearning. Dále jsou uvedeny základní informace o XML, jak implementovat DOM strukturu a způsoby pro validaci dokumentů. Následuje popis vývojového prostředí Delphi a programovacího jazyka Object Pascal. V neposlední řadě je zmíněn význam pojmu scénář a navazuje pár slov o stávajícím řešení vývoje scénářů. V závěru práce je popsáno, jak bylo postupováno při programování aplikace.

Annotation

At the first part of the work there is explained the term Authoring Tools and shortly described eLearning. There is mentioned the basic information about XML, how to implement DOM structure and manners for validation documents. Next, there is the description of developmental environment Delphi and programming language called Object Pascal. At least, there is mentioned the meaning of term Scenario and there is written few words about existing solution of development of scenarios. At the end of my work, there is described how it was progressed during programming application.

Obsah

Prohlášení ..... - 3 -

Poděkování ..... - 4 -

Anotace ..... - 5 -

Annotation ..... - 5 -

Obsah ..... - 6 -

Seznam obrázků ..... - 10 -

Seznam zkratek ..... - 11 -

Úvod ..... - 12 -

1 Autorský nástroj ..... - 13 -

1.1 Základní úroveň ..... - 13 -

1.1.1 PowerPoint ..... - 13 -

1.1.2 HTML a JavaScript ..... - 13 -

1.1.3 Dreamweaver a Flash ..... - 14 -

1.2 Střední úroveň ..... - 14 -

1.2.1 Macromedia Authorware ..... - 14 -

1.2.2 ToolBook II Instructor ..... - 14 -

1.3 Jak najít vhodný autorský nástroj ..... - 15 -

1.3.1 Snadné použití versus svoboda tvořivosti ..... - 15 -

1.3.2 Automatizované programování ..... - 15 -

1.3.3 Vzájemná kompatibilita a standard ..... - 16 -

1.3.4 Typy otázek ..... - 16 -

1.3.5 Media a podpora souborů ..... - 16 -

1.3.6 Rozšiřitelnost ..... - 16 -

2 ELearning ..... - 17 -

2.1 Historie eLearningu ..... - 17 -

3 XML ..... - 18 -

3.1 Úvod do XML ..... - 18 -

3.2 Implementace DOM ..... - 18 -
3.3 Návrh DTD ..... - 19 -
3.4 XML Schéma (XSD) ..... - 20 -
4 Vývojové prostředí Delphi ..... - 21 -
5 Jazyk Object Pasal ..... - 22 -
5.1 Třídy a objekty ..... - 22 -
5.1.1 Přetěžování ..... - 22 -
5.1.2 Self ..... - 23 -
5.1.3 Dynamické vytváření komponent ..... - 23 -
5.1.4 Konstruktor ..... - 23 -
5.1.5 Destruktor ..... - 23 -
5.1.6 Objektový model ..... - 23 -
5.2 Přiřazování objektů ..... - 23 -
5.2.1 Objekt a zpráva paměti ..... - 24 -
5.3 Zapouzdření ..... - 24 -
5.3.1 Zapouzdření programové jednotky ..... - 24 -
5.3.2 Zapouzdření proměnných do vlastností ..... - 24 -
5.3.3 Upozornění ..... - 25 -
5.4 Dědění ..... - 25 -
5.5 Dynamická vazba a polymorfismus ..... - 25 -
5.5.1 Potlačení a předefinování metody ..... - 26 -
5.5.2 Virtuální a dynamické metody ..... - 26 -
5.5.3 Abstraktní metody ..... - 26 -
5.6 Bezpečné přetypování ..... - 26 -
5.7 Rozhraní ..... - 27 -
5.8 Výjimky ..... - 27 -
6 Scénář ..... - 28 -
6.1 K čemu bude program sloužit ..... - 28 -
7 Stávající způsob vytváření scénářů ..... - 29 -
7.1 Nevýhody ..... - 29 -

8 Realizace ..... - 30 -

8.1 Použité komponenty ..... - 30 -
8.2 Návrh interface ..... - 30 -
8.2.1 Funkce seznamu listů ..... - 31 -
8.2.2 Stavový řádek ..... - 32 -
8.2.3 Vzhled XP ..... - 32 -
8.3 Šablona ..... - 33 -
8.3.1 Formátování písma ..... - 34 -
8.4 Vlastní komponenta ..... - 34 -
8.5 XML ..... - 34 -
8.6 Nový projekt ..... - 34 -
8.7 Otevření projektu ..... - 35 -
8.8 Uložení projektu ..... - 35 -
8.9 Média ..... - 36 -
8.9.1 Vložení obrázku ..... - 36 -
8.9.2 Vložení audia ..... - 37 -
8.9.3 Vložení a přehrání videa ..... - 37 -
8.9.4 Smazání média ..... - 37 -
8.10 Vložit a smazat list ..... - 37 -
8.10.1 Vložení listu ..... - 38 -
8.10.2 Smazání listu ..... - 38 -
8.11 Zavření projektu a aplikace ..... - 38 -
8.12 Překlady ..... - 38 -
8.12.1 Překlad aplikace ..... - 39 -
8.12.2 Překlad projektu ..... - 40 -
8.13 Okno „O programu“ ..... - 40 -
8.14 Validace oproti XSD ..... - 40 -
8.15 Komprimace ..... - 41 -
8.16 Struktura programu ..... - 42 -
8.17 Automatické přečíslování ..... - 42 -
8.17.1 Pojmenování ..... - 42 -
8.18 Práce s adresáři a soubory ..... - 43 -
8.19 Podporované formáty ..... - 44 -

Závěr ..... **- 45 -**

Použitá literatura ..... **- 47 -**

Příloha A – Metoda NewProject ..... **- 48 -**

Příloha B – Metoda InsertPicture ..... **- 50 -**

Příloha C – Stručný manuál ..... **- 52 -**

Seznam obrázků

Obrázek 1: Hierarchická stromová struktura DOM ..... - 19 -
Obrázek 2: Lišta tlačítek rychlého spuštění ..... - 31 -
Obrázek 3: Vzhled aplikace ..... - 32 -
Obrázek 4: Šablona a s načtenými položkami ..... - 33 -
Obrázek 5: Okno O programu ..... - 40 -
Obrázek 6: Struktura stránkově orientovaného programu ..... - 43 -

Seznam zkratek

AICC	Aviation Industry Compute-based Training Committee
CBT	Computer Based Training
CLX	Component Library for Cross Platform
DHTML	Dynamic HTML
DOM	Document Object Model
DTD	Document Type Definition
HTML	HyperText Markup Language
IDE	Integrated Development Environment
LMS	Learning Management System
OI	Object Inspector
OOP	Objektově orientované programovaní
RAD	Rapid Application Development
SCORM	Sharable Content Object Reference Model
SGML	Standard Generalized Markup Language
VCL	Visual Component Library
WBT	Web Based Training
XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformation

Úvod

Práce začíná teoretickým popisem autorského nástroje, eLearningu a použitých technologií, navazuje vyjasnění pojmu scénář a jeho dosavadního vytváření a plynule přechází k řešení jednotlivých požadavků od zadavatele na výslednou aplikaci.

Autorský nástroj je aplikace pro vytváření eLearningových kurzů s multimediálním obsahem. Pokud chceme vybrat pro nás vhodný program, je třeba se zaměřit na podporované formáty a standard, který nástroj dodržuje. Je to důležité kvůli případné kompatibilitě.

V části použitých technologií je popsáno XML, struktura DOM, DTD a XSD pro validaci XML dokumentu. Základní informace o vývojové prostředí Delphi a programovacím jazyka Object Pascal. Jsou zde ještě rozebrány známé principy OOP, tak jak je řeší Delphi. Principy o které se jedná, jsou abstrakce, dědičnost, polymorfizmus, objekty a zapouzdření.

Další pasáží je vysvětlení, co v tomto případě scénář vlastně znamená a jakým způsobem byly scénáře doposud vyvíjeny.

Na závěr práce je uveden popis a postup vývoje aplikace dle požadavků zadavatele.

1 Autorský nástroj

Termín autorský nástroj může být pro někoho matoucí. Když lidé slyší tento termín poprvé, často předpokládají, že se jedná o specializovanou formu softwaru pro zpracovávání textu, určenou profesionálním spisovatelům. Avšak autorské nástroje jdou daleko za zpracováním textu. Pro tuto kategorii softwaru je pravděpodobně přesnější název Nástroje pro vytvoření eLearningových kurzů.

ELearningové autorské nástroje umožňují prezentovat materiály formou kurzů za použití řady různých médií, jedná se hlavně o animace, audia a videa. Tento nástroj je určený pro vytváření profesionálních, poutavých, interaktivních tréninkových kurzů, testů s případnou možností test následně vyhodnotit a uložit známkování pro pozdější potřeby. Některé nástroje dokonce umožňují předělat již existujících kurzy, aby se daly znovu použít jako nové kurzy. [9]

Autorské nástroje se dají rozdělit na úrovně. Na základní úrovni jsou jako autorské eLearningové aplikace jako Microsoft PowerPoint, Macromedia Dreamweaver a Flash. Macromedia Authoware, Direktor a SumTotal ToolBook II jsou aplikace patřící do střední úrovně a jsou používány hlavně jako CBT aplikace, které jsou šířeny formou CD-ROMu či DVD. [11]

1.1 Základní úroveň

1.1.1 PowerPoint

PowerPoint či jiné jednoduché nástroje založené na prezentaci je možné využít pro vytváření eLearningových kurzů. Ačkoliv výsledná prezentace z tohoto programu bývá lineární, lze do ní přidáním hypertextových odkazů přeskočit do jiné části. Do prezentace lze zahrnout také audio nebo video klipy.

Testování a následné vyhodnocení znalostí získaných v kurzu je v tomto nástroji celkem komplikované.

Výhodou je možnost distribuce jako CTB i jako WTB. [11]

1.1.2 HTML a JavaScript

Standardní HTML stránky nabízejí kvizy psané v jazyce JavaSkript. Což je jednoduchá cesta k vytváření eLearningových kurzů pro použití na internetu. [11]

1.1.3 Dreamweaver a Flash

Dreamweaver nabízí vytváření kurzů, který umožňuje přidat testování do webových stránek s eLearningovou aplikací.

Pokud jde o animace, ty mohou být udělány ve Flashi za použití ActionScriptu. Takto tvořené aplikace jsou dobře použitelné a flexibilní. Tyto programy jsou distribuovány hlavně na webových stránkách. Někdy je místo Flashe používán Direktor kvůli jeho jazykovým možnostem. [11]

1.2 Střední úroveň

1.2.1 Macromedia Authorware

Nejkomplexnější autorský nástroj pro tvorbu eLearningových aplikací, interaktivních katalogů a publikací, populárně naučných elektronických publikací, interaktivních vzdělávacích kurzů, digitálních kiosků a nejrůznějších simulátorů.

Plně podporuje eLearningové standardy AICC, SCORM a IMS. Obsahuje Web Player, který umožňuje vytvořenou aplikaci snadno přenést na internet, intranet a následně prohlížet ve webových prohlížečích.

Ovládání prostředí programu je jednoduché a intuitivní. Lze importovat prezentace z PowerPointu. Podporuje používání interaktivních animací vytvořených v programech Macromedia Flash nebo Director. [8]

1.2.2 ToolBook II Instructor

V ToolBooku lze vyvíjet WBT, CBT, multimediální aplikace, simulace pro software a další multimediální obsah. Nástroj kombinuje použití šablon, pomocníků a předpřipravených položek. Obsahuje plně rozvinutý programovací jazyk OpenSricpt.

Zahrnuje podporu vkládání například MS Word i PowerPoint. Vytvořené kurzy lze exportovat do DHTML, umožňuje streamování videa a audia, začlenění animace a simulace. Mezi podporované standardy patří AICC, SCORM, IMS, ADL a IEEE. [6]

1.3 Jak najít vhodný autorský nástroj

Výběr toho nejlepšího autorského nástroje vyžaduje detailnější seznámení s nástroji a bez ohledu na to jako moc (jak málo) víme o autorských nástrojích, stanovit funkcionalitu, která je nejvíce důležitá pro nás. Např. pokud budeme potřebovat kurzy tisknout či zveřejnit na internetu.

Aktuálně dostupné nástroje nabízejí rozmanité vlastnosti, tedy i pro specifické nároky se určitě najde aplikace, která bude vyhovovat těmto požadavkům. Některé nástroje jsou navržené k vývoji rozsáhlých, softwarových simulací nebo pro kapesní počítače. Nicméně mnoho nástrojů je navrženo k vytváření základních eLearningových kurzů pro stolní počítače a laptopy. [9]

1.3.1 Snadné použití versus svoboda tvořivosti

Na jedné straně, jsou nástroje založené na šablonách, které nevyžadují prakticky žádné zkušenosti s ovládáním. Tyto programy jsou velmi dobře formátované již při vývoji kurzu, který je řízen posloupností dialogových oken. Bohužel flexibilita těchto programů je omezena. Ale snadné použití může být důležitější než možnost kreativity.

Na opačné straně jsou autorské nástroje navržené k produkci špičkových multimediálních simulací se sofistikovanou grafikou a zvuky. Tyto nástroje přirozeně vyžadují několik týdnů či měsíců příprav pro to, abychom se nástroj naučili dobře ovládat, avšak nabízí velkou svobodu tvorby.

A někde mezi těmito stranami jsou nástroje, které vyžadují sice více cviku v ovládání, ale nabízejí širší pole pro tvořivost. Většina balíků na trhu spadá do této kategorie a je používána k vytváření většiny eLearningových kurzů současné doby. [9]

1.3.2 Automatizované programování

Automatickým programováním pro online distribuce autorských nástrojů osvobozují vývojáře kurzu od jejich závislosti na programátorech. Nabízejí podporu pro různá média a typy souborů, jako text, grafika a jejich následný export do některých obecně používaných formátů, například HTML, XML nebo DHTML. Typ výstupního formátu se podstatně liší v závislosti na použitém nástroji. Při výběru autorského nástroje, který negeneruje zdrojový kód, se musíme naučit programovat či se spolehnout na někoho, kdo této problematice rozumí. Jinak nebudeme schopni publikovat své kurzy. [9]

1.3.3 Vzájemná kompatibilita a standard

Schopnost autorských nástrojů pracovat s dalšími eLearningovými programy a systémy je odkázána na kompatibilitu.

ELearningové společenství obsahuje několik technologických standardů a neustále se vyvíjí doplňkové technologie. Cílem je, aby byla kompatibilita po celém eLearningové trhu. Do této doby je eLeraningové společenství rozdělené do rozdílných systémů dodržující různé standardy. Čtyři nejvíce obvyklé standardy jsou SCORM - referenční model pro eLearning, Microsoft LRN, AICC – výcvik v leteckém průmyslu a IMS – globální vzdělávací konsorcium.

Autorské nástroje se liší ve standardech, které podporují. Pokud již organizace používá nějaký LMS, při rozhodování o novém autorském nástroji je dobré zvážit, jestli je kompatibilní se stávajícím LMS. [9]

1.3.4 Typy otázek

Ti, kteří kurzy vyvíjejí, se zajímají o rozdílné typy kladených otázek, které mohou vytvořit v autorském nástroji. Jedná se vyplňování formulářů, odpovídání pravda či nepravda, krátké nebo dlouhé eseje. [9]

1.3.5 Media a podpora souborů

Je důležité věnovat velkou pozornost podporovaným médiím a formátům souborů. Většina autorských nástrojů podporuje běžné formáty jako JPG, WAV a GIF. Zatímco některé podporují tok videa či audia. [9]

1.3.6 Rozšiřitelnost

Někdy je potřeba modifikovat software pro specifický účel. Schopnost, která toto umožňuje, se nazývá rozšiřitelnost. Pokud je požadována nějaká modifikace autorského nástroje, je vhodné se ujistit, že vybraný nástroj disponuje Open Source kódem, který dovoluje zkušenému počítačovému programátorovi přizpůsobit software potřebám školitele. [9]

2 ELearning

Vzhledem k nepřetržitému a dynamickému vývoji této oblasti existuje mnoho definicí pojmu eLearning. ELearning je vzdělávací proces, který využívá multimediální prvky, jako jsou například animace a videa k tvorbě kurzů pro podporu distanční formy studia. Může obsahovat mnoho dílčích prvků, jako jsou testovací rozhraní, diskusní fóra a další synchronní a asynchronní komunikační nástroje. [4]

2.1 Historie eLearningu

Začátek eLearningu se datuje na konec 60. let minulého století. V této době se začínalo experimentovat se stroji. V naší zemi byl vyvinut stroj nazvaný UNITUTOR (Tesla). Celosvětově byl považován za jeden z nejlepších. Nebyl to PC, ale zařízení pro jeden účel složené z tranzistorů a obrazovky. Látka k výuce byla rozdělena na stránky s kontrolní otázkou na konci a několika možnými odpověďmi. Stroj uměl dle odpovědi větvit anebo pokračovat dál. Tento způsob se časem ukázal jako neúčinný a neujal se.

Příchod osobního počítače znamenal pokrok. Výukové programy byly orientované především na testování, následné obodování, hodnocení vyplývalo ze součtu bodů a bylo možné uchovávání výsledků. Pouhé testování však nestačilo. Výklad se začal doplňovat o multimediální prvky. Kurzy se skládaly z klíčových prvků výklad, procvičování a test. Již v této době se objevili prvky umělé inteligence, program musel umět zpracovat všechny možnosti, které mohou nastat.

Na počátku. 90. let nastal převrat s příchodem a rozvojem počítačových sítí zejména Internetu a změnami v oblasti komunikace, například masivní rozmach e-mailů, CD-ROMů – jež byli levnější variantou než učitel, ač jejich výroba byla stále dosti nákladná. Používání CD-ROMů bylo izolovaným způsobem výuky, pokud nastal problém, nebylo s kým problém konzultovat. Tento nedostatek začalo odstraňovat používání e-mailů, asynchronních diskusních tabulí, synchronních chatů atd.

Internet začínal být cenným zdrojem informací a zábavy pro studenty. Proto se začalo s digitalizací učebních materiálů, jako jsou skripta a publikováním na Internetu. Díky těmto materiálům došlo k vytvoření virtuální univerzity, která nabízí kurzy pro distanční studium. Tyto kurzy jsou vhodné i pro pracující lidi, kteří si chtějí doplnit vzdělání a neztrácet čas s dojížděním do vzdálené školy. [7]

3 XML

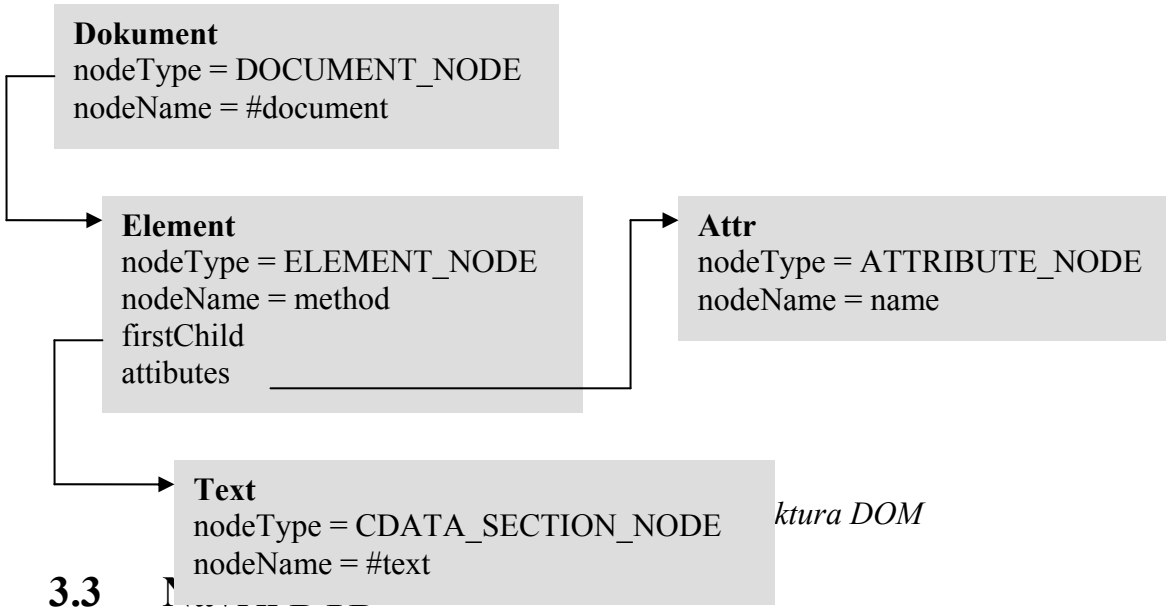
3.1 Úvod do XML

XML (Extensible Markup Language) je jednoduchý rozšiřitelný strukturální meta-značkovací jazyk. Data jsou označkována jednoduchými a srozumitelnými značkami. Tato množina používaných značek není pevně daná, jako např. u HTML (HyperText Markup Language). Lze stanovit vlastní množinu značek. Množina značek, která se používá pro určitou oblast, je označována jako aplikace XML. Syntaxe pro značky je striktně dána a musí se dodržovat.

XML je uloženo v souboru majícího tvar textového řetězce, který je čitelný pro člověka, v paměti nebo jako záznam v databázi. Parsery jsou programy, které dokážou rozebrat dokument na jednotlivé elementy, atributy a ostatní. Validační parsery porovnávající DTD (Dokument Type Definition), které je buď v dokumentu, nebo dokument obsahuje odkaz na DTD. Validační chyby nejsou fatální, stačí, když je dokument správně formulovaný, pokud není, nedojde ke zpracování. XML není reprezentačním ani programovacím jazykem, nemá kompilátor, který by napsaný kód zkompiloval a poté vytvořil spustitelný soubor. Samotné XML na rozdíl od programu (aplikace) nevykonává vůbec nic. Název značky udává význam dat nikoli způsob formátování. [1]

3.2 Implementace DOM

DOM (Dokument Object Model) objektový model dokumentu je určen pro ukládání dokumentů XML do paměti a není závislý na programovacím jazyce ani operačním systému. Do hierarchie DOM se ukládají odkazy mezi jednotlivými uzly v dokumentu. Pro práci s XML dokumentem jej nejprve musíme načíst do paměti, a pak s ním může aplikace pracovat. Tento způsob je výhodný, pokud potřebujeme přistupovat k datům náhodně či opakovaně. Nevhodné použití je zejména pro aplikace, které transformují dokumenty. Takže se příliš nehodí pro postupné zpracovávání, jelikož je tento model pomalý. [1]



ktura DOM

3.3 N

Definice typu dokumentu udává, které elementy a atributy se mohou nacházet v XML dokumentu. Určuje, jak po sobě jednotlivé elementy následují a v jakém počtu se budou vyskytovat. Jestli dokument vyhovuje DTD, ověřuje validátor. Vyhovuje-li, je validní, jinak je neplatný.

Deklarace elementů je ve tvaru:

```
<!ELEMENT jméno (kontextový_model)>
```

Kontextový model obsahuje jeden či více synovských elementů. Model určuje, v jakém pořadí se mají vyskytovat. Kontextový model je možné nahradit zápisem *#PCDATA*, který říká, že element obsahuje jen text.

DTD použitím znaku blíže specifikuje počet potomků. Znak udává počet ? žádný či jeden, + jeden či více a * žádný, jeden či více. Důležité je i správně označeni, že se jedná o prázdné elementy:

```
<!ELEMENT jméno EMPTY>
```

Při deklaraci atributů je nutné uvést všechny atributy pro jednotlivé elementy.

```
<!ATTLIST jméno_elementu atribut1 CDATA #hodnota
atribut2 CDATA #hodnota >
```

CDATA je jeden z typů atributů a značí, že může obsahovat textová data, ale zároveň musí být platným názvem XML. Hodnoty tributů, které byly v návrhu použity, jsou # *IMPLIED* pro nepovinné atributy a # *REQUIRED* pro povinné atributy.

Pro připojení DTD k souboru je zvolen tento zápis:

```
<!DOCTYPE source [DTD]>
<source>
</source>
```

[1]

3.4 XML Schéma (XSD)

XML schéma je na XML založená alternativa k DTD, také popisuje přípustnou strukturu dokumentu, ale navíc umožňuje kontrolovat správnost dat, práci s daty uloženými v databázi a konvertovat data mezi různými datovými typy.

Schéma je označováno za silnější nástroj než je DTD a je nástupcem DTD. Dnes se již používá v mnoha webových aplikacích.

Zápis XSD je stejný jako zápis XML dokumentu. Značky ovšem obsahují prefix xs, <xs:schema>. Jednotlivé elementy samozřejmě mohou obsahovat také atributy.

Definice elementů a atributů:

```
<xs:element name="xxx" type="yyy" default="zzz" />
<xs:attribute name="xxx" type="yyy" default="zzz" />
```

Typy, které lze použít jsou, string, decimal, integer, boolean, date a time. Přidáním *default* lze nastavit počáteční hodnotu, nebo pomocí *fixed* nastavit pevnou hodnotu. Hodnoty atributu lze omezit typem, výčtem parametrů, rozsahem, délkou stringu atd.

Omezit elementy je možné podle toho, jestli mohou obsahovat potomky (*complexType*) nebo nemohou (*simpleType*), podle minimálního (*minOccurs*) a maximálního (*maxOccurs*) výskytu elementů.

K definici elementů v uzlu *complexType* se používají identifikátoru *seqence*, *all* a *choice*. *Seqence* značí sekvenci elementů, kde na pořadí záleží, pomocí *all* se definuje výčet, kde na pořadí nezáleží a posledním identifikátorem je *choice*, který dovoluje pouze jednu z možných variant. [10]

4 Vývojové prostředí Delphi

Delphi je integrované vývojové prostředí - IDE pro vývoj aplikací pro Windows. Používá programovací jazyk Object Pascal, který je založený na vyšším programovacím jazyce s OOP. Delphi používá knihovny VCL (Visual Component Library) a multiplatformní CLX (Component Library for Cross Platform). Objektový model je nezávislý na počtu implementací jednotlivých tříd.

Obsahuje systém RAD (Rapid Application Development), který umožňuje vizuální návrh uživatelského rozhraní skládáním komponent na kontejner. Tyto komponenty automaticky generují zdrojový kód, což vede k rapidnímu zrychlení návrhu. Prostředí již obsahuje řadu komponent, další je možné stáhnout a importovat. Výhodou je možnost vytvoření vlastních komponent. Pro jednodušší nastavení vlastností a událostí komponent je tu inspektor objektů – OI.

Prostředí kompiluje kód do jednoduchého spustitelného kódu s eliminací funkcí dynamických knihoven, je zde možnost kompilace do x86 kódu nebo převedení do .NET kódu. Prostředí dále zahrnuje podporu pro vývoj databází, nabízí nástroje pro tvorbu XML, podpora internetových aplikací atd.

Výhodou je přenositelnost vytvořeného zdrojového kódu mezi platformami a to do prostředí. Zdrojové kódy vytvořené ve starších verzích Delphi jsou kompatibilní s novými verzemi Delphi. [2]

5 Jazyk Object Pasal

Programování v Delphi vychází z jazyka Object Pascal, ale na rozdíl od Pascalu je Delphi založené na vizuálním prostředí. Delphi není case sensitive a je přísně typovým jazykem, a proto je vhodný pro výuku na školách.

Jako u ostatních vyšších programovacích jazyků se jedná také o objektově orientovaný jazyk, čili zde platí také všechny známé principy OOP:

- Objekty
- Abstrakce
- Zapouzdření
- Dědičnost
- Polymorfismus

Oproti Pacsalu jsou zde přidány další direktivy překladače jako $IF a $ELSEIF pro podmíněný překlad, $WARN umožňuje vypnout chybová hlášení, $MESSAGE slouží k předávání uživatelských informací mezi hlášeními překladače. [2]

5.1 Třídy a objekty

Jedná se o 2 základní termíny OOP. Třída je datový typ, který je definován pomocí klíčového slova *class*.

Definice třídy obsahuje deklarace atributů a metod. Object je instancí třídy. Vztah mezi objektem a třídou je stejný jako mezi proměnnou a typem.

Pokud chceme použít třídu, je nutné vytvořit její instanci. Instance se vytváří pomocí konstruktoru create a nepotřebnou instanci musíme zrušit destruktorem, čili zavoláním metody free. [2]

5.1.1 Přetěžování

Přetěžování funkcí a metod znamená použití stejné metody ale s jinými a jiným počtem parametrů, ale pozor nelze odlišit změnou typu návratové hodnoty. Zapisuje se klíčovým slovem *overload* za deklaraci metody. [2]

5.1.2 Self

Klíčové slovo self je používáno k vyřešení konfliktu jmen metod a zlepšuje čitelnost kódu. Je-li při vytváření komponenty třeba explicitně odkázat na aktuální formulář. V rámci metody je možné se odkazovat na tento parametr pomocí self. [2]

5.1.3 Dynamické vytváření komponent

Komponentu vytvoříme dynamicky za použití konstruktoru a nastavením její vlastnosti *parent*. Termíny *parent* a *owner* se liší, *owner* je zodpovědný za zrušení objektu. [2]

5.1.4 Konstruktor

Konstruktor jakožto metoda *Create* alokuje paměť pro instanci příslušného objektu. Instance může být dále přiřazena do proměnné. Při zavolání konstruktoru jsou počáteční hodnoty data objektu nastaveny na *nil*. Pokud potřebujeme jiné počáteční hodnoty, musíme si vytvořit vlastní konstruktor přetížením toho původního. [2]

5.1.5 Destruktor

Destruktor, metoda *Free* zavolá virtuální metodu *Destroy*, která uvolní systémové prostředky včetně paměti. *Free* kontroluje před voláním metody *Destroy*, zda objekt není *nil*. Pokud ne, vykoná *Destroy*. Případně je možné použít metodu *FreeAndNil*, která provede Destroy a potom nastaví ukazatel na objekt na hodnotu *nil*. [2]

5.1.6 Objektový model

Instance komponent, které se přidávají na formulář, se paměť alokuje a odstraňuje z paměti automaticky. Ukazatel na objekt, jež je instancí některé třídy je pouze odkazem do paměti, kde je objekt uložen. Deklarací proměnné se rezervuje místo v paměti pro odkaz na objekt, instanci je nutné udělat ručně. [2]

5.2 Přiřazování objektů

Přiřazením se jen zkopíruje odkaz na objekt, na to je třeba si dát pozor. Pokud bychom potřebovali zkopírovat i obsah objektu použijeme příkaz *assign*. [2]

5.2.1 Objekt a zpráva paměti

Každý objekt je třeba nejprve vytvořit, než jej budeme moci použít. Pokud objekt nepotřebujeme, je třeba ho zrušit a to pouze jednou. Čili co sami vytvoříme, sami musíme zrušit.

Jiná možnost je, objekt vytvořit a specifikovat vlastníka, který se o zrušení již sám postará.

Řetězce, dynamická pole a objekty odkazované proměnnými jsou uvolňované Delphami, není-li odkaz již dále používán. [2]

5.3 Zapouzdření

OOP vyžaduje skrývání dat ve třídě. Je třeba umožnit vnitřní úpravy bez negativního vlivu na funkčnost aplikace. Zapouzdření je pro představu něco jako černá skříňka, nemáme přímí přístup k vnitřním datům, ale jen pomocí příslušných metod, aby nedošlo k narušení aplikace. [2]

5.3.1 Zapouzdření programové jednotky

Toto zapouzdření se realizuje pomocí 3 modifikátorů

Private – volně přístupné

Public – v rámci jednotky

Protected – zde jsou deklarovány věci potřebné v potomcích této třídy

[2]

5.3.2 Zapouzdření proměnných do vlastností

property Bod: Integer read GetBod write SetBod default 1;

Pokud chceme definovat vlastnost pouze pro čtení, vynecháme část s *write*, pokud pouze pro zápis, což se spíše nepoužívá, vynechali bychom *read*.

Vlastnosti nejsou alokovány v paměti, tudíž je nelze předávat jako parametry procedur či funkcí. Tyto deklarované vlastnosti se zobrazují v OI. [2]

5.3.3 Upozornění

- Data definovaná ve var příslušného formuláře se neváží na konkrétní instanci formuláře, ale na celý program, tedy další instance tohoto formuláře budou tato data sdílet
- Je lepší použít privátní proměnnou než veřejnou globální proměnnou a deklarovat veřejnou metodu pro přístup k ní, aby nedošlo při změně proměnné ke změně kódu [2]

5.4 Dědění

Dědění znamená možnost odvozovat od již existujících tříd. Tento způsob je mnohem lepší než pouhé kopírování. Problémem kopírování je, že kód je třeba spravovat vícekrát, je tu také riziko, že zkopírujeme kód i s chybou. Poděděný objekt obsahuje všechny metody, datové položky, vlastnosti i události. Prapředek všech objektů je TObject.

Zápis: TfrFrame = class(TFrame);

[2]

5.5 Dynamická vazba a polymorfismus

Funkce a procedury používají statickou vazbu (časovou) – volání metody řeší přímo překladač nebo vývojový program – náhradí se volaná metoda místem v paměti, kde je uložena. OOP umožňuje i jiný typ vazby a to dynamickou (pozdní), aktuální adresa volané metody se určuje za běhu programu. Tento přístup je nazýván polymorfismus (vícetvarost).

Hlavní myšlenkou je zapsání metody a použití jí na proměnnou typu objekt. Metoda, kterou Delphi zavolají, záleží na typu objektu, ke kterému se proměnná vztahuje.

V několika příbuzných třídách mohou existovat různé verze jedné metody, na které může odkazovat jediné volání metody instance objektu určité třídy. Verze volané metody závisí na typu instance. [2]

5.5.1 Potlačení a předefinování metody

Pokud jsou metody předka definované jako *virtual*, je možné tyto metody překrýt. Překrývající metodu je nutné označit jako *override*.

Překrytí lze realizovat dvěma způsoby, prvním je metodu překrýt celou a druhým metodu doplnit a to za použití klíčového slova *inherited* v těle metody.

Při překladu programu překladač kontroluje, zda názvy metod odpovídají. [2]

5.5.2 Virtuální a dynamické metody

Virtuální metody jsou založené na tabulce virtuálních metod VMT. Obsahem tabulky je seznam adres metod, při volání překladač skočí na adresu metody v tabulce VMT příslušného objektu. Tato tabulka urychluje hledání, ale nevýhodou je záznam všech virtuálních metod, tedy i těch, které nejsou v předkovi potlačeny, což může vést k velkým nárokům na paměť.

Dynamické metody jsou realizované voláním pomocí jedinečného kódu označujícího metodu. Hledání odpovídajících funkcí je pomalejší než hledání v tabulce. Na druhou stranu k volání této dynamické metody dojde, jen pokud ji nějaký objekt potlačuje. Z toho vyplývá, že tento způsob není tak náročný na paměť.

U virtuálních metod se nakonec definice metody píše slovo *virtual* a u dynamické *dynamic*. [2]

5.5.3 Abstraktní metody

Abstraktními metodami jsou ty, které jsou výhradně určeny k definování metody v potomkovi. Smysl těchto metod plyne z polymorfismu. [2]

5.6 Bezpečné přetypování

Pokud se potřebujeme dostat k metodě předka třídy, je potřeba jej přetypovat. Explicitní přetypování může způsobit chybu, protože překladač neví, jestli je typ objektu správný a zda obsahuje volanou metodu. Řešením je bezpečné přetypování, které se provádí voláním podmínky. Nejprve se pomocí klíčového slova *is*, zeptáme, zda je objekt potřebného typu, a pak klíčovým slovem *as* provedeme bezpečné přetypování. [2]

5.7 Rozhraní

Jelikož Delphi nepodporují vícenásobnou dědičnost, je možné si dopomoci rozhraním, které se velmi podobá třídě. Obsahuje řadu deklarací tříd, které jsou ryze virtuální a abstraktní. Rozhraní vychází z Interface. Správa paměti je zde automatizovaná. Je možné implementovat i více rozhraní najednou. V deklaraci typu je vhodné uvést GUID číslo, jinak by se mohlo stát, že flexibilita rozhraní bude omezená. [2]

5.8 Výjimky

Výjimky zvyšují odolnost programu doplněním o schopnost jednoduchého a jednotného řešení softwarových a hardwarových chyb. Výjimky lze přečkat, skončit a předtím umožnit uložení dat. Ideální je při výskytu výjimky, ji zpracovat a vyřešit. Výjimky jsou založené na čtyřech klíčových slovech:

Try – začátek bloku

Exception {výjimka} {příkaz}

Finally – část, která se vykoná vždy, např. uvolnění objektů

Raise – příkaz pro vykonání výjimka

Je doporučeno blok chránit slovem finally, aby nedocházelo ke ztrátě či mrhání pamětí.

Samozřejmě je možné vytvořit si vlastní třídu výjimek, jež je odvozená od *Exeption*. Objekty vytvořené při výjimkách není potřeba rušit, ke zrušení dojde jejich automatickou obsluhou. Každý typ výjimky se dá zpracovat různým způsobem a lze je sdružovat do skupin. [3]

6 Scénář

Scénář je literární dílo určené k dalšímu zpracování. Základní struktura bývá dvousloupcová – vlevo je popis situace a vpravo je dialog. Dále existuje také tzv. technický scénář ve filmové branži, popisuje umístění kamery a typ záběru (detail, polodetail, jízda kamery). [5]

Nejedná se tudíž o scénář filmový, ale o programové a obsahové návaznosti – interaktivní propojení a popis děje na stránce.

6.1 K čemu bude program sloužit

Účel programu vyplývá z jeho názvu – **Aplikace pro vytváření scénářů.** Program by měl pomoci při vytváření obsahu aplikací. Při tvorbě většinou už vznikají i všechny vizuální části a v době dokončení scénáře bývá z obsahu programu hotovo již 60-80%. Proto je dobré využít situace a všechny prvky už mít řádně usazené v konečné struktuře a pojmenované tak, aby programátor při konečné realizaci měl co nejméně práce.

Zároveň je to základní dokument pro kontrolu úplnosti obsahu. Při schvalování obsahu zákazníkem vznikají požadavky na různé změny – od úpravy grafiky, změny nebo přesunutí textu až po zásah do struktury – v případě našich stránkově orientovaných programů – přeskupení stránek – jiné větvení apod. Náš Nástroj by měl usnadnit tyto změny realizovat na úrovni scénáře tak, aby k programátorovi odcházel jednoznačný a hlavně schválený produkt zákazníkem.

7 Stávající způsob vytváření scénářů

Dosud byly k vytváření scénářů používány nástroje z řady MS Office, a to zejména PowerPoint a Exel, někdy i Word.

V PowerPointu byly vytvářeny scénáře jako takové. Doteď měl scénář například 150 stánek, ale ve skutečnosti podle struktury stránkové orientované aplikace bylo stran zhruba 14.

Dále se používali dva soubory vypracované v Exelu. První soubor Basics.xls, který je podkladem pro databázi, spravuje veškeré texty aplikace. Texty k obrázkům, audiím a další potřebné texty. Ke každému z textů soubor obsahuje i překlady.

Struktura.xls je druhý soubor, který určuje základní a přehledné schéma programu. Každý uvedený kód je jednou stránkou, které odpovídá i adresář s médii.

7.1 Nevýhody

- Velké množství stránek
- Při vkládání textu zvenčí do pole dochází k ne vždy správnému zformátování
- Není možné celou strukturu rovnou zabalit do zipu
- Obrázky, které jsou vloženy na slide, je třeba mít uložené ještě někde v adresáři ve správné velikosti, chybí informace o jméně souboru, případně o cestě vloženého obrázku
- Texty, které jsou na slidu, je třeba mít spravované a zapsané například v Excelu, aby byly připraveny pro změny, korektury nebo překlady
- Když se změní kód strany, je třeba tuto změnu provést ručně ve scénáři, i u všech médií

8 Realizace

K vytvoření aplikace byl vybrán programovací jazyk Delphi, resp. Object Pascal. Delphi je příjemné prostředí pro psaní aplikací určených pro stolní počítače.

8.1 Použité komponenty

Tabulka 1: Seznam a popis použitých komponent

typ komponenty	stručný popis
TForm	základní kontejner pro ostatní komponenty
TfrTemplate	šablona poděděná od TFrame
TTimer	časovač pro zobrazování času do stavového řádku
TLabel	textové popisky
TRichEdit	podporuje víceřádkový text s formátováním, možnost scrolleru
TImage	pro zobrazení obrázků
TListBox	seznam objektů
TFrameListBox	poděděn od TListBoxu, seznam jednotlivých šablon
TStatusBar	panel pro zobrazení stran, jazyka a data s časem
TToolBar	kontejner pro ToolButtony
TToolButton	ikony pro rychlejší manipulaci s programem
TMainMenu	menu aplikace
TXMLDocument	pro vytváření XML dokumentu
TOpenDialog	zajišťuje dialogové okno pro výběr jména souboru pro otevření
TSaveDialog	zajišťuje dialogové okno pro výběr jména souboru pro uložení
TOpenPictureDialog	pro otevření obrázku

8.2 Návrh interface

Při návrhu bylo třeba dbát na požadavky zadavatele a vymyslet pokud možno snadné a intuitivní ovládání. V popisku aplikace je uveden název, a pokud je otevřen nějaký projekt, tak je za pomlčkou zobrazen název projektu.

Aplikace obsahuje menu s položkami Soubor, Vložit, Smazat, Video, Jazyk a O aplikaci. Menu bez položek se chová obdobně jako tlačítko. Přehledné zobrazení struktury menu nabízí následující tabulka.

Tabulka 2: Struktura menu

Menu	**Soubor**	**Vložit**	**Smazat**	**Video**	**Jazyk**	**O aplikaci**
Položky	Nový Otevřít Uložit Importovat ZIP Exportovat ZIP Zavřít projekt Konec	List Obrázek Audio Video	List Obrázek Audio Video	Přehrát Ukončit		

Pod menu je umístěna lišta s tlačítky, na kterých jsou ikony s obrázky a nápověda pro snadnou orientaci. Na liště nejsou zobrazena tlačítka pro všechny funkce uvedené v menu, protože některá nebudou tak často používána.

Lišta je rozdělena na 5 částí. První část obsahuje tlačítka pro založení nového projektu, otevření, uložení a zavření. V druhé části jsou tlačítka pro vložení a smazání listu. Třetí část má na starosti vkládání médií, a to obrázků, audia či videa. Čtvrtá část tato média maže. A poslední část se stará o přehrání či ukončení přehrávání videa.

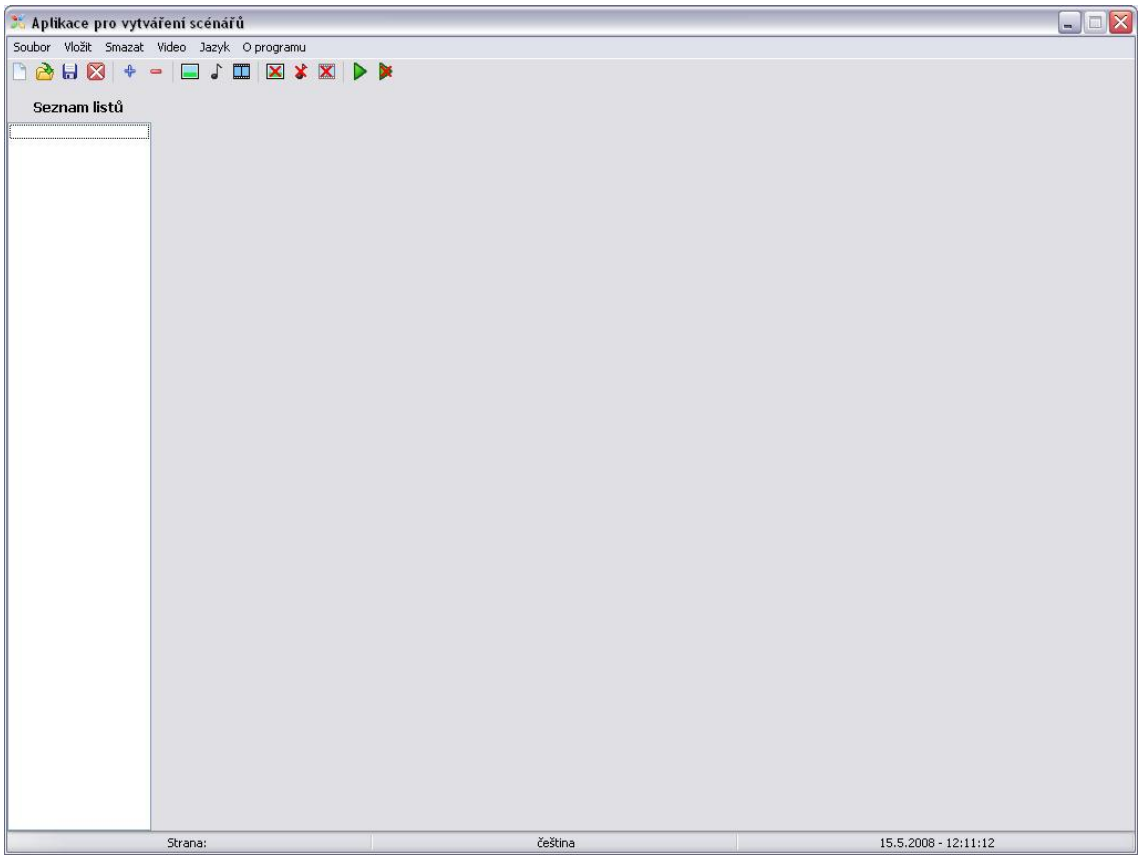
Obrázek 2: Lišta tlačítek rychlého spuštění

V levé části aplikace je dlouhý seznam, který spravuje jednotlivých listů. Seznam disponuje i některými funkcemi, které budou popsány níže. Dále aplikace disponuje stavovým řádkem a do volného prostoru v aplikaci se vkládá šablona.

8.2.1 Funkce seznamu listů

Pro práci se seznamem jsou zde 2 důležité funkce.

- Označení položky – při označení položky se zobrazí příslušná stránka scénáře
- Přejmenování – při dvojitém kliknutí na položku aplikace nabídne dialogové okno pro zadání nového názvu položky, následně se automaticky nastaví tento text na příslušném místě šablony



Obrázek 3: Vzhled aplikace

8.2.2 Stavový řádek

Stavový řádek v této aplikaci má 3 panely a každý zobrazuje nějakou informaci. V prvním je zobrazeno číslo aktuální stránky. Druhý panel sděluje, který jazyk aplikace je zvolen. Poslední obsahuje aktuální datum a čas. Realizace data a času je pomocí API funkce *GetLocalTime* s parametrem typu SYSTEMTIME.

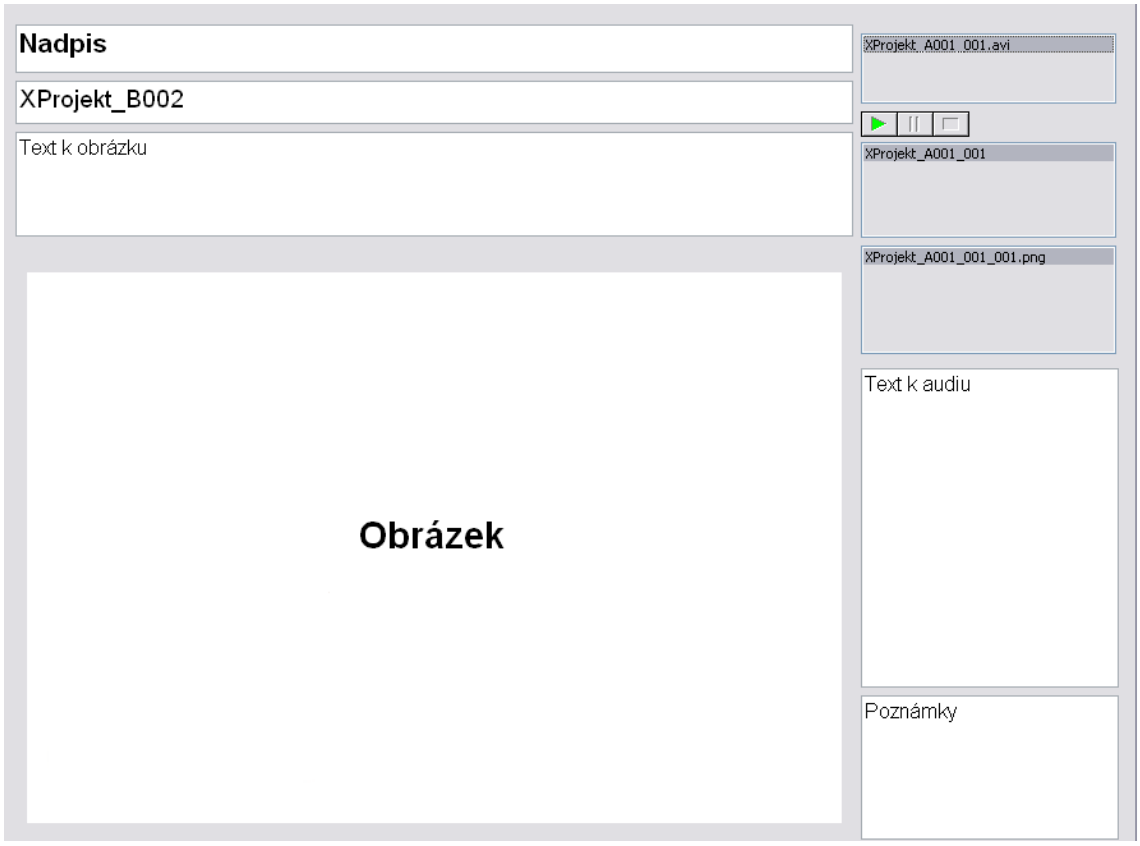
8.2.3 Vzhled XP

Aby mohla mít aplikace vzhled jako Windows XP, je potřeba vytvořit soubor s názvem jméno_aplikace.exe.manifest. Soubor obsahuje XML kód, který umožní styl XP. Dále je třeba ještě zdrojový soubor xp.rc, který obsahuje řádek s příkazem. Soubor je nutné přidat do projektu. Nyní stačí již jen zkompilovat a je hotovo.

8.3 Šablona

Problematika stránkově orientované aplikace je řešena pomocí šablony. Samotná šablona je komponenta *TFrame*, což je pro tento případ velmi vhodným kontejnerem. Na tento rámec jsou umístěny potřebné komponenty: *TRichEdit*, *TLabel*, *TImage*, *TListBox* a *TMediaPlayer*.

Jak je vidět názorně na obrázku prvním textem je nadpis, pod ním je zobrazen kód stránky a dále text, který souvisí s právě zobrazeným obrázkem. Po levé straně jsou vidět tři seznamy, které zobrazují názvy vložených médií i s příslušnou příponou. Nad seznamem sdružujícím audia je zobrazena komponenta umožňující přehrání vybraného audia. Pro přehrání videa slouží menu a tlačítka na liště. Prohlížení obrázků je možné označováním položek v seznamu nebo kliknutím přímo na obrázek. Pod seznamy je místo pro text právě označeného audia. V pravém dolním rohu je ještě možnost zapsání nějakých poznámek.



Obrázek 4: Šablona a s načtenými položkami

8.3.1 Formátování písma

Formát písma pro texty v šabloně je neměně nastaven v metodě *SelChangeRichEdit*. Velikost písma a popř. tučnost je individuální dle písma, bylo požadováno bezpatkové *Arial*. Při vložení textu z vnějšího zdroje dojde k automatickému zformátování dle nastavení.

8.4 Vlastní komponenta

Jelikož žádná komponenta svými funkcemi plně nevyhovovala, bylo nutné pro zjednodušení práce, vytvořit komponentu vlastní. Nová komponenta pojmenovaná *TFrameList* je poděděná od *TListBoxu*. Jedná se tedy o seznam šablon (objektů typu *TFrameList*). Tato komponenta obsahuje nové metody pro zjednodušení práce, asi nejvýznamnější jsou *AddFrFrame* pro vkládání objektů typu *TfrTemplate* do seznamu, *DeleteFrFrame* pro smazání položky a uvolnění objektu z paměti a metoda Last, která vrací poslední vložený objekt.

8.5 XML

Aby bylo možné používat XML v Delphi, je potřeba importovat potřebnou knihovnu. V tomto případě knihovnu Microsoft XML verze 3.0, tato verze byla vybrána, protože je to nejvyšší možná verze, která je nainstalována již po čisté instalaci OS Windows XP. Po importu je možné komponentu nainstalovat.

Tvorba struktury DOM probíhá v paměti, pro uložení do souboru je nutné projekt uložit.

8.6 Nový projekt

Založit nový projekt lze buď přes menu Soubor, nebo tlačítkem na liště. Před založením nového projektu je uživatel vyzván na případné uložení rozdělaného projektu. Pokud souhlasí, projekt se uloží a ukončí, pokud nesouhlasí, metoda se ukončí a nic se neděje.

Další dialogové okno slouží pro zadání jména nového projektu a vytvoří se stejně jmenující adresář pro ukládání dalších věcí projektu. Nyní je třeba začít vytvářet DOM strukturu v paměti. Jako první se vytvoří kořenový element *struct* a následují další elementy *project* s atributem jména projektu a *pages*.

```
<struct>
     <project name="name" />
     <pages>
     </pages>
</struct>
```

Nakonec dojde k vložení prvního listu, což znamená i další generování DOM struktury.

8.7 Otevření projektu

Při načítání projektu aplikace zjistí, zda není již nějaký projekt načtený, pokud ano vybídne uživatele k uložení. Stávající projekt se uloží a zavře, potom se nabídne dialogové okno s filtrem pro vybrání XML souboru, který se načte, následuje validace XML souboru oproti XSD. Jestliže se otevření nebo validace nezdaří, metoda se ukončí.

V metodě pro načítání souboru je již zabudovaný DOM parser, který dále umožní další práci s XML v paměti. Nejprve se načte název projektu, který bude zobrazen i v popisku aplikace. Následuje vytváření stránek v cyklu. Když se vytváří stránka, zavolá se metoda *InsertList*, avšak bez poslední části, kde se vytváří XML. Po vytvoření je třeba vložit texty na patřičná místa a všechna média do příslušných seznamů, což se děje opět v cyklu. Jako vybraná položka je nastavena první z každého neprázdného seznamu. Při načítání obrázků je nutné ještě zobrazit příslušný text i obrázek samotný. Obdobné je to i u audia s rozdílem, že se první záznam otevře do přehrávače. Načtení videa se liší jen nezobrazováním textu.

8.8 Uložení projektu

Metoda ovládající uložení se jmenuje *Save*. Majoritní část je cyklus přes všechny stránky. Do XML se uloží texty nadpisů a poznámek, pokud je vložen obrázek nebo audio, dojde k uložení textu obrázku a audia pro aktuálně označené médium. Uloží se celá DOM struktura do souboru s příponou XML a ten je uložen do projektu. Nakonec se proměnná indikující změnu v projektu nastaví na *false*.

8.9 Média

Použitými médii v této aplikaci se rozumí audia, obrázky a video. Média jsou vkládány do seznamu v *ListBoxu*, který podporuje funkce:

- Označení položky v seznamu médií:
    - Obrázek – kliknutím na jinou položku se zobrazí příslušný obrázek
    - Audio – označené audio lze přehrát kliknutím na *play* zobrazeného přehrávače
- Přejmenování – po dvojitém kliknutí na položku se nabídne dialogové okno, kam zadáme nový název souboru a to včetně přípony, takto dojde k přejmenování položky i samotného souboru ve složce nesoucí název stránky, nakonec se přejmenuje i atribut *name* pro médium v XML
- Smazání – smazaní odstraní položku, příslušný soubor z adresáře a celý uzel z XML
- Přesunutí – pomocí myši chytneme položku, přesuneme myš a pustíme na požadované pozici, ostatní média se posunou dolu a ještě dojde k přesunutí uzlů v XML

8.9.1 Vložení obrázku

Vše se odehrává v metodě *InsertPicture*, která nabídne dialogové okno s filtrem pro výběr obrázku. Vytvoří se objekt, který uchovává název a cestu kam je obrázek následně zkopírován. Dále je objekt vložen do seznamu určenému pro obrázky na aktuální stránce. Obrázek je načten do komponenty *TImage* a zobrazen uživateli. Položka v seznamu posledního načteného obrázku zůstane označená. Následuje uložení informací do XML. Sem se uloží název obrázku a text, který je na začátku nastaven na prázdný znak.

```
<picture name="TCAN2_007EA.png" text="reContent" />
```

Po kliknutí na obrázek je vždy zobrazen obrázek následující, po posledním je to první obrázek.

8.9.2 Vložení audia

Metoda *InsertAudio* otevře dialogové okno s příslušným filtrem pro vybrání souboru. Do vytvořeného objektu se přiřadí jméno a cesta audia. Soubor je zkopírován do příslušné složky. Objekt se přidá do seznamu určenému pro audia. Komponenta *TMediaPlayer* se zviditelní, nastaví se cesta k souboru pro přehrání a audio se otevře do přehrávače, tak aby po kliknutí na zelenou šipku bylo možné audio rovnou přehrát. V poslední části procedura vloží podelement do elementu *audios* do XML.

```
<audio name="TCAN2_001BD_001.wav" text="reLabel" />
```

8.9.3 Vložení a přehrání videa

Metoda *InsertVideo* vyvolá dialog pro výběr video souboru. Vytvoří se objekt a do něj se přiřadí jméno a cesta souboru. Následuje zkopírování souboru do složky aktuální stránky. A konečné uložení části XML pod element video.

```
<video name="Video.avi" text="" />
```

Pro přehrání videa položka v menu nebo ikona na liště. O přehrání se stará metoda *PlayVideo*, která má jako parametr cestu i se jménem videa. Zde se vytvoří objekt přehrávače *TWindowsMediaPlayer*, nastaví se potřebné vlastnosti a začne samotné přehrávání. Stisknutím tlačítka zastavit dojde k ukončení přehrávání, zároveň se zavře přehrávač a uvolní se vytvořený objekt.

8.9.4 Smazání média

Pro smazání média jsou implementovány 3 metody *DeletePicture*, *DeleteAudio* a *DeleteVideo*. Při zavolání některé z metod je otevřeno dialogové okno pro potvrzení smazání, aby nedošlo k nechtěné ztrátě. Postup je u jednotlivých medií obdobný. Nejprve dojde ke smazání všech informací o médiu z XML, dále se smaže příslušný soubor média, odejme a uvolní se objekt ze seznamu, čili zmizí položka. A nakonec drobné nastavení, pro audio to znamená zavření přehrávače pro tuto položku a pro obrázek nastavení metody Picture komponenty *TImage* na *nil*.

8.10 Vložit a smazat list

Pro vkládání šablony v programu slouží metoda *InsertList* a pro smazání metoda *DeleteList* s parametrem index listu, který se má smazat.

8.10.1 Vložení listu

V první části metody *InsertList* se vytvoří objekt typu *TfrTemplate*, přidá se do seznamu listů, nastaví se pozice a parent. Položka dostane počáteční jméno dle názvu projektu a indexu. Následuje přiřazení potřebných událostí pro komponenty. Přidaná položka se nastaví na vybranou a index vkládaného listu se zobrazí do stavového řádku jako číslo stránky, index se inkrementuje. V poslední části metody se realizuje vložení uzlů a atributů do elementu *pages* v XML.

```
<page code="XCan_A001" title="Nadpis" note="Poznámky">
   <pictures />
   <audios />
   <videos />
</page>
```

8.10.2 Smazání listu

Při zavolání metody *DeleteList* pro smazání se objeví dialog pro potvrzení. Dojde ke smazání celého elementu *page* z XML dokumenty v paměti. Metoda seznamu *DeleteFrFrame* s parametrem předaným metodou smazaní, uvolní instanci dané šablony z paměti a odstranění položky ze seznamu. Následuje smazání adresáře i s obsahem do koše.

8.11 Zavření projektu a aplikace

Před zavřením projektu či aplikace se program zeptá na uložení změn, pakliže jsou změny již uloženy, je aplikace rovnou zavřena. Při zavírání projektu je ještě třeba ručně uvolnit objekty a inicializovat proměnné. To obstará procedura *Init*.

8.12 Překlady

Dalším požadavkem na aplikaci je umožnit překlad do jiných jazyků. Překlad je realizován na dvou úrovních. První úrovní je myšleno překlad samotné aplikace a druhou možnost a usnadnění přeložit finální projekt.

8.12.1 Překlad aplikace

Pro překlad aplikace je vybraný způsob ukládat všechny použité texty do INI souboru. Jelikož se jedná o jednoduchý textový dokument, je možné ho snadno zeditovat a tím pádem i překladem vytvořit nový soubor pro nový jazyk. Aplikace obsahuje již vytvořený soubor pro češtinu, který je načítán při spuštění programu. Ten zároveň slouží i jako předloha pro vytváření dalších souborů. Druhým ukázkovým souborem je anglická verze.

Příklad INI souboru:

```
[Aplication_Language]
Aplication_Language=Čeština
[Menu_File]
File=&Soubor
New=Nový
Save=Uložit
ImportZip=Import ZIP
ExportZip=Export ZIP
Close_Project=Zavřít projekt
Close_Aplication=Konec
[Menu_Insert]
…
```

Jak je vidět na příkladu, struktura INI souboru je celkem přehledná. V hranatých závorkách jsou popisky, které napomáhají orientaci v souboru. Pod popisky jsou vypsány jednotlivé texty. Na pravé straně je anglickým názvem popsáno k čemu slouží a na straně levé je samotný text, který se následně zobrazí a používá na příslušných místech v programu.

O ukládání INI souborů se stará metoda *SaveIniFile*, která vytvoří nový soubor a o načítání metoda *LoadFromIni*.

8.12.2 Překlad projektu

Po dohodě se zadavatelem byl schválen způsob překladu projektu přímo zeditováním výsledného XML souboru ve vhodném XML editoru tak, aby se zabránilo případnému porušení validity souboru. K tomuto řešení bylo přihlédnuto i z důvodu pozdějšího využití pro dalšího programátora, kterému tento způsob usnadní práci.

8.13 Okno „O programu“

Okno O programu je malý formulář obsahující data o aplikaci. V tomto případě se jedná o jméno autora, verzi programu a datum vydání programu. Pro uložení textů jsou použity komponenty typu *TLabel*. Stejně jako rozhraní programu i tento malý formulář je nutné přeložit.

Obrázek 5: Okno O programu

8.14 Validace oproti XSD

Jelikož je možné a v případě překladu projektu žádoucí, aby uživatel zasahoval do XML souboru, je zde riziko poškození struktury. Proto je nutné před načítáním souboru nechat soubor zkontrolovat validárotem oproti předdefinovanému XSD. V tomto případě je XSD vygenerováno aplikací dle XML získaného z vhodného scénáře. Pro představu je uveden příklad XSD vytvořeného z jednoduchého XML kódu.

XML:

```
<struct>
   <project name="Projekt"/>
</struct>
```

XDS:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="struct" type="structType"/>
  <xs:complexType name="structType">
    <xs:sequence>
      <xs:element name="project" type="projectType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="projectType">
    <xs:sequence/>
    <xs:attribute name="name" type="xs:string"/>
  </xs:complexType>
</xs:schema>
```

8.15 Komprimace

Pro komprimaci a dekomprimaci v rámci zip archivu je nutné zavést knihovnu do prostředí.

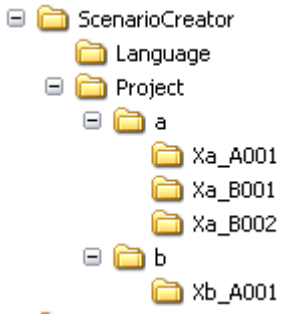
Pro komprimaci nejprve do proměnné přiřadíme adresář, který je třeba zabalit. Nastavíme, že je potřeba použít rekurzivní přidávání souborů a podadresářů do archivu. Nakonec je celý obsah přidán do zipu.

Při rozbalování otevřeme cestu k zipu a nastavíme cestu pro rozbalení a dále se celý obsah rozbalí.

8.16 Struktura programu

Adresářová struktura programu na disku je ukázána na obrázku. Je zde jeden hlavní adresář, ve kterém je celá struktura programu. V hlavním adresáři jsou soubory nutné k plné funkci aplikace. Jedná se o project.exe pro spuštění, schema.xsd pro validaci, zipdll.dll a unzdll.dll potřebné ke komprimaci a dekomprimaci projektu v rámci zip archivu.

Podadresář Language obsahuje INI soubory pro překlad aplikace. Druhým je Project, kam se defaultně ukládá projekt. Ten obsahuje podadresáře nesoucí názvy dle stránek v projektu a je zde soubor xml.xml nesoucí veškeré potřebné informace o vytvořeném projektu. Každý z těchto adresářů obsahuje použitá média na stránce.



Obrázek 6: Struktura adresářů programu na disku

8.17 Automatické přečíslování

Při přejmenovávání položky program prověří, zda již daná položka existuje, pokud ne dokončí přejmenování. Jinak upozorní, že tento název i složka již existují a nedovolí ji znovu vytvořit.

Po každém přejmenování položky program zavolá program metodu *Rename*, která vezme nový název a použije funkce *GetCode*, která vrátí čtyři klíčové znaky určující úroveň ve scénáři, jedná se např. o A001. Následně dojde k přejmenování všech položek médií na této stránce. Program vychází z předem známé struktury pojmenování.

8.17.1 Pojmenování

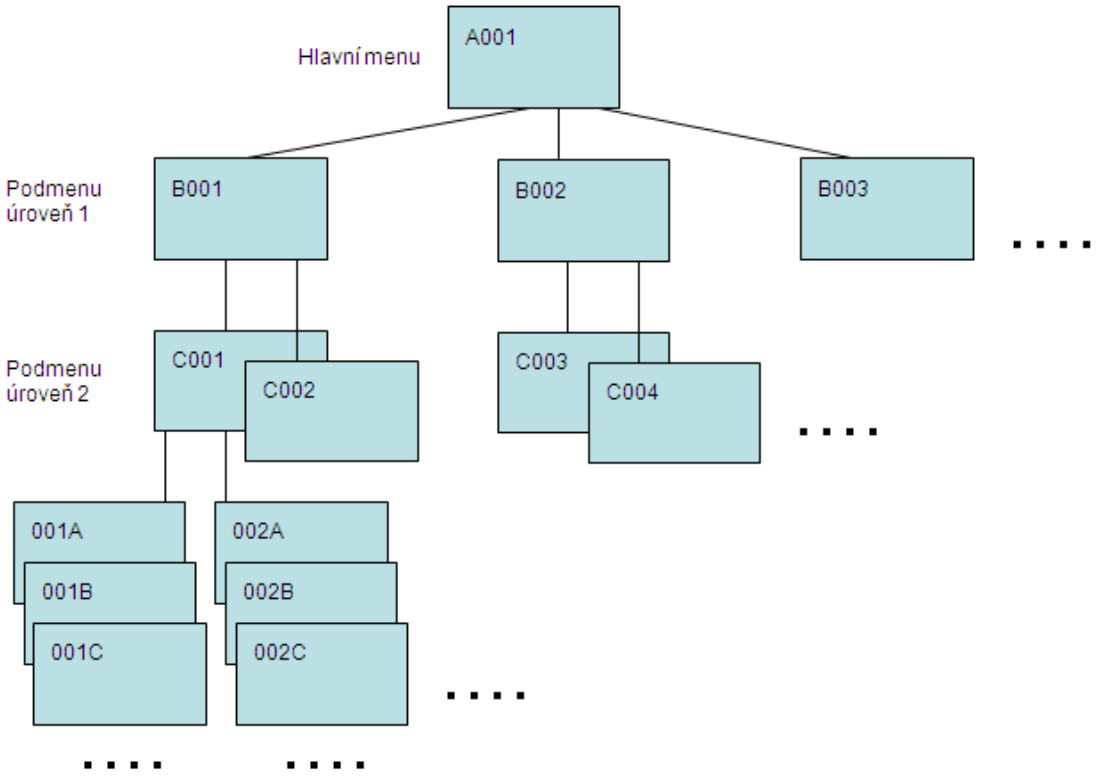
Struktura pojmenování stránek a médií je složená ze znaku určující, zda se jedná o obsahovou stránku (X) či stránku testovací (T). Následuje název projektu, podtržítko a označení úrovně stránky, pro média ještě pokračuje podtržítko a označení, že se jedná o médium a případně další podtržítko a pořadové číslo média.

XProjekt_A001 – kód stránky

XProjekt_A001_001.wav – kód pro audio

XProjekt_A001_001_001.png – kód pro obrázek související s audiem

TProjekt_001AA – první test ke stránce A001



Obrázek 7: Struktura stránkově orientovaného programu

8.18 Práce s adresáři a soubory

Při realizaci práce bylo potřeba manipulovat s adresáři a se soubory. Složky bylo třeba vytvářet, mazat a přejmenovávat. Se soubory bylo třeba provést operace kopírování, mazání a přejmenování.

Vytvoření adresáře obstará metoda *CreateDir* s parametrem jméno adresáře. Před samotným vytvořením je zjištěno, zda již takový adresář existuje, pokud ne, je vytvořen.

Smazání adresáře i s podadresáři zajistí funkce *DelTree* s parametrem jméno adresáře. Metoda používá unitu *ShellApi*. Adresář i se svým obsahem je smazán do koše.

Přejmenování adresáře je realizováno metodou *MoveFile* s parametry starý a nový název adresáře.

Soubor je nutné zkopírovat z původního místa do složky s názvem kódu stránky. *CopyFile* s parametry původní a nová cesta souboru.

Přejmenování souboru provádí metoda *RenameFile*, která má jako vstupní parametry staré a nové jméno souboru.

Smazání souboru se provádí metodou *DeleteFile* s parametrem jména souboru určenému k smazání.

8.19 Podporované formáty

V aplikaci je možné pracovat s různými typy formátů. Vstupní formáty pro obrázky, audia, videa a výstupním formátem je XML dokument.

Obrázky:

Bmp, jpg, jpeg, ico, emf, wmf a png

Všechny formáty krom png jsou podporovány Delphami přímo, ale pro zahrnutí png bylo třeba přidat externí knihovnu.

Audio:

Mp3, wav, mid …

Video:

Mpg, mpeg, wmv, avi a další dle dostupných kodeků v počítači.

Závěr

Vytvořený nástroj spadá do kategorie autorských nástrojů založených na šablonách. Používání šablon je pro uživatele jednodušší na ovládání a výsledný scénář bude mít jednotný, šablonami předdefinovaný vzhled. Aplikace disponuje jednou šablonou, která byla sestavena dle požadavků zadavatele.

Stávajícím způsobem vytváření scénářů bylo použití nástrojů MS Office. Scénář byl vyvíjen v PowerPointu a dále byly potřeba dva soubory vytvořené v Exelu. Jeden udával kompletní strukturu scénáře a druhý obsahoval všechny potřebné informace jako texty a jejich překlady.

Oproti minulému řešení aplikace přináší řadu výhod. Všechny nástroje, které jsou potřeba k vývoji, nejsou nijak zpoplatněné a až na případné použití editoru XML se nemusí instalovat. Aplikace je zcela nezávislá.

Další výhoda je možnost uložení celé struktury projektu do jednoho souboru zip, přenesení a následné načtení projektu. To vše v rámci jediné aplikace.

Oproti PowerPointu nese aplikace výhodu i ve formátování textu. Po jeho vložení do polí zaujme předdefinovaný a neměnný formát.

Nutným požadavkem bylo vyřešit překlad jak aplikace samotné, tak i celého projektu. Aplikace je překládána pomocí INI souborů, které lze snadno přeložit zeditováním. Pro projekt je zvolen způsob editace již výstupního XML dokumentu.

Média se vkládají do seznamů a jako soubory jsou uložené v adresáři s názvem příslušné stránky. Jsou tedy přehledně vidět názvy médií a je zcela jasné, kde je hledat. Tento způsob velice usnadňuje práci programátora, který již nemusí luštit záměr scénáristy a upravovat média, jak tomu bylo v případě PowerPointu.

Aplikace umožňuje dosáhnout jisté standardizace přípravy scénáře před samotným programováním. Zároveň je třeba mít možnost vizuálně vše ukázat zákazníkovi a mít sofistikovaný nástroj pro zapracování připomínek, tak abychom eliminovali chyby vznikající při jejich přenosu do finálního produktu

Jako mnoho aplikací i tato obsahuje okno „O programu“ s informacemi o autorovi, datu vydání a verzi.

Automatické přečíslování je realizováno, tak že po přejmenování dojde k přečíslování médií na stránce.

Aplikace umožňuje provádět různé změny, například úpravy posloupností jednotlivých médií, přesunutí textu, přeskupení stránek či obměnu větvení scénáře. Změny na úrovni scénářů jsou již nutné, aby k programátorovi odcházel jednoznačný a hlavně schválený produkt zákazníkem. Pro programátora z toho vyplívá usnadnění práce a pro další účely použije stávající výstupní XML soubor.

Po shrnutí lze na závěr říci, že zadání diplomové práce s požadavky od zadavatele bylo splněno.

Do budoucna by bylo dobré doplnit ještě podporu obrázků typu TIFF. Používaný seznam by bylo vhodné nahradit jinou komponentou, která umožní zobrazit nelineární strukturu stránek ve scénáři.

Použitá literatura

Knihy:

[1] HAROLD Elliotte Rusty, MEANS W. Scott. *XML v kostce*. Computer Press 2002, ISBN 80-7226-712-4

[2] CANTÚ Marco. *Myslíme v jazyku Delphi 6 1. díl*. Grada 2002, ISBN 80-247-0334-3

[3] CANTÚ Marco. *Myslíme v jazyku Delphi 7*. Grada 2003, ISBN 80-247-0694-6

Internet:

České:

[4] Wikipedie. *ELearning*. [online]. [cit.21.4.2008].
URL: http://cs.wikipedia.org/wiki/ELearning

[5] Wikipedie. *Scénář*. [online]. [cit.21.4.2008].
URL: http://cs.wikipedia.org/wiki/Sc%C3%A9n%C3%A1%C5%99

[6] Kontis s.r.o. *ToolBook II Instructor*. [online]. [cit.21.4.2008].
URL: http://www.e-learn.cz/produkty_instructor.asp?menu=produkty&submenu=vyvoj&subsubmenu=toolbook&subsubsubmenu=instruktor

[7] Mgr. KOPECKÝ Kamil Ph.D. *Základy e-leaningu*. [online]. [cit.21.4.2008].
URL: http://www.net-university.cz/cdrom/demo.swf

[8] AMOS Software. *Macromedia Authorware*. [online]. [cit.21.4.2008].
URL: http://www.amsoft.cz/produkty/adobe/authorware/overview.html

Anglické:

[9] HARRIS Jeff. *An Introdution to Authoring Tools*. [online]. [cit. 20.2.2008].
URL: http://learningcircuits.org/2002/mar2002/harris.html

[10] W3SCHOOLS. *Introduction to XML Schema*. [online]. [cit. 20.4.2008].
URL: http://www.w3schools.com/Schema/schema_intro.asp

[11] KURTUS Ron. *Authoring Tools for eLearning, CBT and WBT*. [online]. [cit. 20.2.2008]. URL: http://school-for-champions.com/elearning/authortools.htm

Příloha A – Metoda NewProject

```
procedure TfMain.NewProject;
var
  dir: String;
begin
  if FTextChanged or (FrameListBox.Count > 0) then
    if MessageDlg(rsChanges, mtWarning, [mbYes, mbCancel], 0) = mrYes
then
    begin
      Save;
      Init;
    end
    else
      Exit;

  FLoadingXML := False;
  //TODO: dotazovat se?
  FIndex := 0;
  if InputQuery(rsNewProject, rsNameOfNewProject, FProject) then
  begin
    fMain.Caption := rsCaptionAplication + ' - ' + FProject;
    dir := ExtractFilePath(Application.ExeName) + cProjects + FProject
             + '\';
    if not DirectoryExists(ExtractFilePath(dir))
      then CreateDir(ExtractFilePath(dir));

    //koren
    FiXmlDOM := msxml2_tlb.CoDOMDocument40.Create;
    FiRoot := FiXmlDOM.appendChild(FiXmlDOM.createElement('struct'));

    //uzel
    FiPages := FiRoot.appendChild(FiXmlDOM.createElement('project'));

    FiAttribute := FiXmlDOM.createAttribute('name');
    FiAttribute.nodeValue := FProject; //jmeno projektu
    FiPages.attributes.setNamedItem(FiAttribute);

    FiPages := FiRoot.appendChild(FiXmlDOM.createElement('pages'));
```

```
   if FrameListBox.Items.Count = 0 then
      InsertList;
  end;
end;
```

Procedura *NewProject* slouží k založení nového projektu. Pokud už je nějaký otevřený, program upozorní na změny, které je třeba uložit. Při odpovědi „*ne“* se metoda ukončí. Pokud je odpověď „*ano*“, program otevřený projekt uloží metodou *Save* a inicializuje zavoláním *Init*.

Další dialogové okno slouží pro zadání jména nového projektu a vytvoří se stejně jmenující adresář pro ukládání celého projektu. Následuje vytvoření DOM struktury v paměti.

```
<struct>
      <project name="name" />
      <pages>
      </pages>
</struct>
```

Nakonec dojde k vložení prvního listu, což znamená i další generování DOM struktury.

Příloha B – Metoda InsertPicture

```
procedure TfMain.InsertPicture;
var
  hlpObject: TAudioPicture;
  dir, item: String;
  hlpNode: IXMLDOMNode;
begin
  OpenPictureDialog1.InitialDir :=
ExtractFilePath(Application.ExeName) + cProjects + FProject + '\';
  if FrameListBox.Items.Count > 0 then  // pokud uz neco je v listu
    if OpenPictureDialog1.Execute then
    begin
      with FrameListBox.FrFrames[FrameListBox.ItemIndex] do
      begin
        hlpObject := TAudioPicture.Create;

        item := FrameListBox.Items[FrameListBox.ItemIndex];
        dir := ExtractFilePath(Application.ExeName)+ cProjects +
                  FProject + '\' + item + '\';
        hlpObject.Path := dir;
        hlpObject.Name := ExtractFileName
                              ( OpenPictureDialog1.FileName);

        CopyFile(PAnsiChar(OpenPictureDialog1.FileName),
                PAnsiChar (dir + hlpObject.Name), true);

        lbPictures.Items.AddObject(hlpObject.FName, hlpObject);
        Image.Picture.LoadFromFile(OpenPictureDialog1.FileName);
        lbPictures.Selected[lbPictures.Count - 1] := True;
       end;
{ XML }
      hlpNode := FiXmlDOM.documentElement.childNodes[1].
                   childNodes[FrameListBox.ItemIndex].childNodes[0];
      // pridat element podelementu audios
      FiPicture := hlpNode.appendChild
                     (FiXmlDOM.createElement('picture'));
      FiAttribute := FiXmlDOM.createAttribute('name');
      with FrameListBox.FrFrames[FrameListBox.ItemIndex].lbPictures do
        FiAttribute.nodeValue := Items[ItemIndex];
```

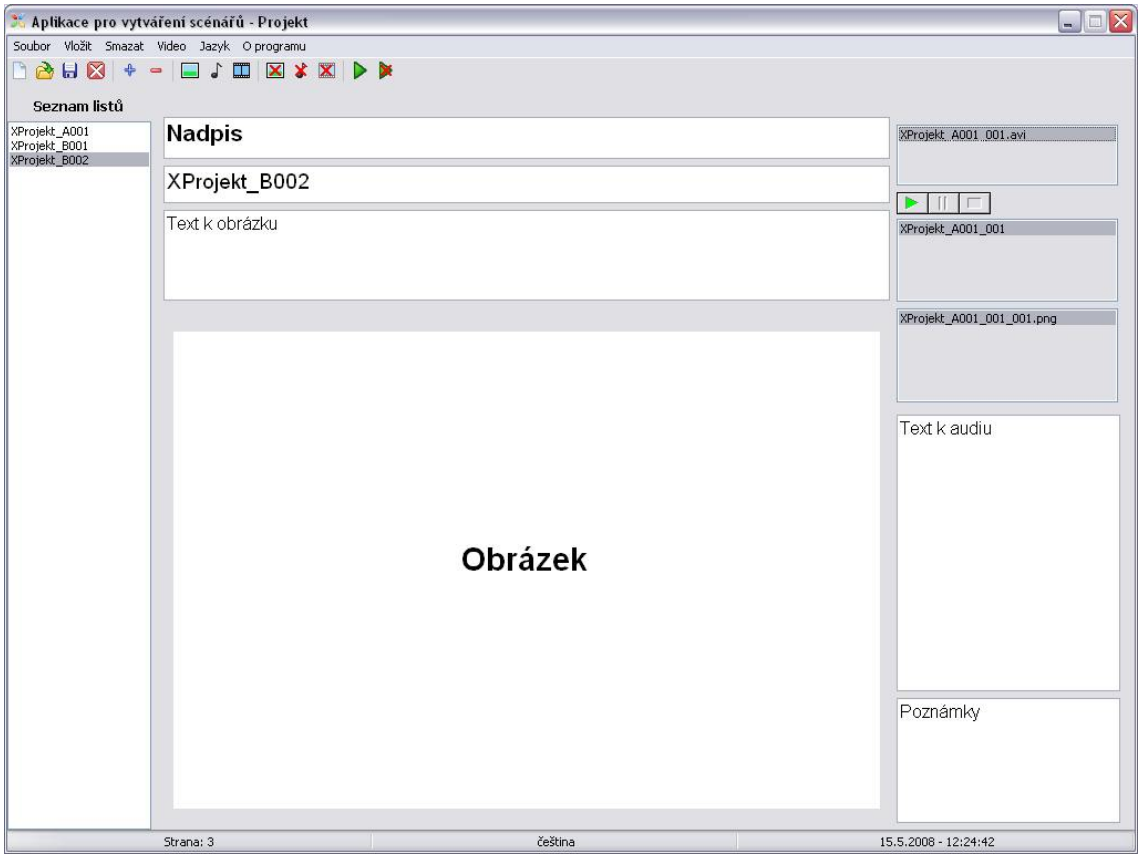
```
      FiPicture.attributes.setNamedItem(FiAttribute);

      // ulozit jen pro aktualni text obrazku ne u vsech
      FiAttribute := FiXmlDOM.createAttribute('text');
      FiAttribute.nodeValue := '';  // inicializace na prazdny znak
      FiPicture.attributes.setNamedItem(FiAttribute);
    end;
end;
```

InsertImage je metoda pro vložení obrázku. Po zavolání metody se otevře dialogové okno pro vybrání média. Vytvoří se objekt typu *TAudioPicture*, do kterého se uloží jméno a nová cesta k souboru, kam bude obrázek následně zkopírován. Objekt je přidán do seznamu. Dále je obrázek načten a zobrazen komponentou *TImage*. Poslední vložená položka je označena jako vybraná.

V poslední části metody se přidá do elementu *pictures* element *picture* s atributy *name* pro jméno souboru a *text*, kde je uložen text, který se váže k tomuto obrázku. Nyní je nastaven na prázdný znak.

Příloha C – Stručný manuál

Aplikace je založená na používání šablon. V popisku aplikace je její název a jméno otevřeného projektu. Pro zjednodušené ovládání obsahuje program menu a lištu s tlačítky pro rychlé spuštění. Tlačítka na liště jsou opatřena ikonami a nápovědou pro lepší přehlednost. Vlevo je umístěn seznam, který zobrazuje jako položky názvy stran. Ve stavovém řádku je zobrazeno číslo aktuální stránky, vybraný jazyk, datum a čas. Na zbylé volné místo se vkládá šablona.



Vložená šablona

Jak je vidět názorně na obrázku prvním textem je nadpis, pod ním je zobrazen kód stránky a dále text, který souvisí s právě zobrazeným obrázkem. Po levé straně jsou vidět tři seznamy, které zobrazují názvy vložených médií i s příslušnou příponou. Nad seznamem sdružujícím audia je zobrazena komponenta umožňující přehrání vybraného audia. Pro přehrání videa slouží menu a tlačítka na liště. Prohlížení obrázků je možné označováním položek v seznamu nebo kliknutím přímo na obrázek. Pod seznamy je místo pro text právě označeného audia. V pravém dolním rohu je ještě možnost zapsání nějakých poznámek.

Menu Soubor

Nový projekt – zadáme název nového projektu, pokud je nějaký otevřen, program vyzve k uložení a provede zavření. Při vytváření projektu se rovnou vytvoří adresář s názvem projektu.

Načtení projektu – otevře se dialogové okno pro výběr XML soboru. Po potvrzení výběru se vybraný projekt načte.

Uložení projektu – dojde k uloží všech informací do souboru xml.xml, který je uložen do adresáře s názvem projektu.

Importovat ZIP – otevře se dialogové okno pro výběr archivu zip. Po vybrání dojde k rozbalení.

Exportovat ZIP – otevře se dialogové okno pro výběr archivu, který se má rozbalit.

Zavřít projekt – uloží a zavře projekt.

Konec – slouží pro zavření aplikace. Před zavřením je uživatel vybídnut k případnému uložení projektu.

Menu Vložit

Vložit list – pro každý vložený list se vytvoří adresář s jeho jménem.

Vložit obrázek, audio a video – objeví se dialogové okno pro výběr média. Po otevření se položka přidá do seznamu médií a zkopíruje se do složky s názvem stránky. Seznamy médií mají nastavený posuvník pro případ, že se nevejdou.

Obrázek se zobrazí na šabloně. Prohlížení je možné buď klikáním na obrázek anebo označováním položek v seznamu.

Po vložení audia se objeví přehrávač, který je rovnou připraven k přehrání aktuálního audia.

Přehrát video lze pomocí tlačítek na liště nebo z menu.

Menu Smazat

Před jakýmkoliv smazáním se objeví varování, aby nedošlo k nechtěné ztrátě.

Smazat list – při smazání listu se automaticky smaže i adresář s veškerým obsahem do koše.

Smazat obrázek, audio a video – mazání média znamená odstranění položky ze seznamu a smazání souboru z adresáře.

Menu Video

Přehrát – otevře se přehrávač Windows Media Player a spustí se video.

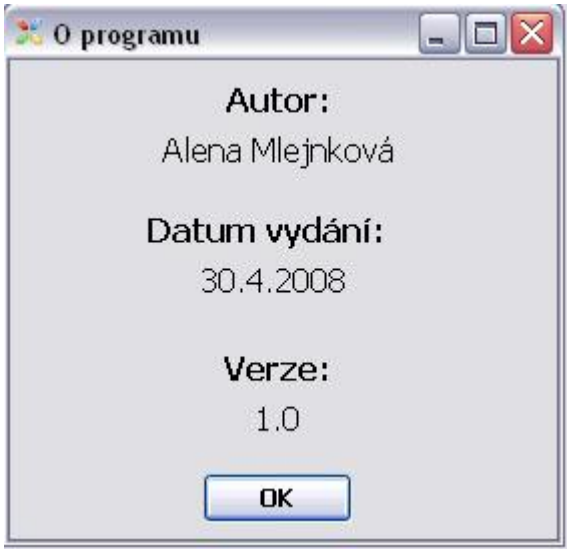
Ukončit – slouží pro zavření přehrávače.

Menu Jazyk

Zde po stisku lze načíst INI soubor s jazykem pro překlad aplikace. Tyto soubory lze jednoduše editovat a tím vytvářet další překlady.

Menu „O aplikaci“

Toto menu po stisku zobrazí formulář s informacemi o programu, které jsou autor, datum vydání a verze.



Automatický formát

Po vložení z externího zdroje zaujme text pevně nastavený formát.

Funkce seznamů

- Označení položky v seznamu:
    - Obrázek – kliknutím na jinou položku se zobrazí příslušný obrázek
    - Audio – označené audio lze přehrát kliknutím na *play* zobrazeného přehrávače
    - List – zobrazí se příslušná stránka
- Přejmenování – po dvojitém kliknutí na položku se nabídne dialogové okno, zadáme nový název stránky nebo souboru včetně přípony, takto dojde k přejmenování položky a adresáře resp. souboru
- Smazání – smazaní odstraní položku ze seznamu a příslušný adresář resp. soubor z adresáře
- Přesunutí – pomocí myši chytneme položku, přesuneme myš a pustíme na požadované pozici, ostatní položky se posunou dolu