TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika

Studijní obor: 1802R022 – Informatika a logistika

Webová aplikace pro e-learning na téma UML a Objektový návrh

Web application for e-learning on the topic of UML and Object-oriented design

Bakalářská práce

Autor: **Jan Paulík**

Vedoucí práce: Ing. Přemysl Svoboda

Konzultant: Ing. Roman Špánek, Ph.D.

V Liberci 16. 5. 2009

Originální zadání

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou 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 BP a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědom(a) toho, že užít své bakalářské 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).

Bakalářskou práci jsem vypracoval(a) samostatně, s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Abstrakt

Bakalářská práce popisuje realizaci e-learningové aplikace na téma UML a Objektový návrh, který je vyučován na Fakultě mechatroniky, informatiky a mezioborových studií Technické univerzity v Liberci.

V úvodní části práce jsou vysvětleny pojmy webová aplikace a e-learning. Dále je popsán návrhový software Adobe Flash CS3, pomocí kterého byl e-learning vytvořen. Jsou zde popsány základní prvky programu a skriptovací jazyk ActionScript využívaný při návrhu aplikace. Nejsou opomenuty programy jiných společností, kterými se může webová aplikace vytvořit.

Následující úsek je rozdělen na témata, která představují jednotlivé kapitoly e-learningu a charakterizují základní složky tvorby grafického jazyka pro objektově orientovanou analýzu a návrh softwaru. Z počátku se kapitoly zaměřují na teorii, kde jsou pro lepší srozumitelnost použity praktické příklady. Závěrečná kapitola je věnována praktickým návrhům, kde je za pomoci softwaru Enterprise Architect znázorněn postup tvorby základních diagramů zachycující různé aspekty modelovaného systému.

Samotnou realizací e-learningu se zabývá zbylá část bakalářské práce, kde je nastíněn postup vytvořené aplikace pomocí všech prostředků, které dodávají aplikaci interaktivitu a možnost využití návrhu při zpracování jiného předmětu.

Klíčová slova: e-learing, webová aplikace, UML, Flash, ActionScript

Abstract

The Bachelor’s Thesis describes the implementation or realization of the e-learning application on the UML topic and Objective concept, which is taught at the Faculty of mechatronics, information science and interdisciplinary studies of the Technical university of Liberec.

The introduction explains the terms of web application and e-learning. Further the design software Adobe Flash CS3, withwhich the e-learning was created, is described. Here the basic elements and scripture languages ActionScript used during the design of the applicaion are described. Programs of other corporations, withwhich the

web applicaion is also created, are not left out. The following section is divided into the topic which presents individual chapters of e-learning and characterizes the fundamental creating components of a graphical language for object oriented analysis and software design. From the beginning the chapters focus on the theory where practical examples are used for a better simplicity and understanding. The conclusion chapter is devoted to practical designs where the method of creating basic diagrams intercepting different aspects of the modeled system are illustrated by means of the Enterprise Architect software.

The remaining part of the Bachelor's Thesis is concerned with the singular realisation of e-learning where the method of the created aplication is outlined by means of all resources and devices that provide the aplication interactivity and the ability to use a design during the processing of other items.

Keywords: e-learing, web application, UML, Flash, ActionScript

ObsahAbstrakt **4**

Obsah **6**

1 Úvod **9**

2 Webové aplikace a e-learning **10**

2.1 Formy e-learningu 10

2.2 Výhody a nevýhody 11

3 Seznámení s prostředím Adobe Flash CS3 ..... **12**

4 Nástroje a funkce Adobe Flash CS3 ..... **13**

4.1 Panel nástrojů 13

4.2 Časová osa, vrstvy a snímky 13

4.2.1 Vrstvy – Layer 13

4.2.2 Snímky – Frames 14

4.3 Scéna 14

4.4 Knihovna 14

4.5 Panel akce a ActionScript 15

4.5.1 ActionScript 1.0 16

4.5.2 ActionScript 2.0 16

4.5.3 AcitonScript 3.0 16

5 Flash player **17**

6 Konkurenční software **18**

6.1 SilverLight 18

6.2 JavaFX 18

7 Rozdělení kapitol pro e-learning **19**

7.1 Co je to UML a modelování 19

7.2 Požadavky 19

7.3 Use Case (případ užití) 19

7.4 Modelování tříd a objektů 20

7.4.1 Asociace 20
7.4.2 Agregace a kompozice 20
7.4.3 Generalizace a specializace 20
7.4.4 Zapouzdření 21
7.4.5 Polymorfismus (mnohotvarost) 21
7.5 Relace 21
7.6 Dědičnost a polymorfismus 22
7.6.1 Dědičnost (generalizace/specializace) 22
7.6.2 Polymorfismus 22
7.7 Stavové a sekvenční diagramy 22
7.7.1 Stavové diagramy 22
7.7.2 Sekvenční diagramy 23
7.8 Diagramy aktiv 24
7.9 Datové modelování 24
7.9.1 Logický datový model 24
7.9.2 Fyzický datový model 25
7.10 Praktické postupy 27
8 Tvorba aplikace v Adobe Flash CS3 27
8.1 Načítání aplikace 28
8.2 Práce na časové ose 29
8.3 Navigační menu 29
8.4 Tvorba a úprava kapitol 30
8.5 Instruktážní videa 31
8.6 Testový modul 32
9 Publikace webové aplikace 33
10 Využití e-learningu pro jiný předmět 35
Závěr 37

Přílohy **38**

Příloha A 38

Literatura **39**

1 Úvod

V dnešní době neustálého rozvoje informačních technologií a rozšiřujících se možností Internetu se studentům naskýtá možnost využít webové rozhraní i ke svému vzdělávání. Pomocí interaktivních nástrojů lze vytvořit webové aplikace vhodné pro studium z domova či zaměstnání.

Tato bakalářské práce se zabývá realizací e-learningové webové aplikace, která bude sloužit jako podpora při výuce na Technické univerzitě v Liberci. Aplikace bude zaměřena na grafický jazyk UML (Unified Modeling Language), který je využíván pro objektově orientovanou analýzu a návrh softwaru. Za využití grafického programu Flash CS3 od společnosti Adobe bude navrhnut e-learning, kde si student může pročíst teoretickou část a následně vyzkoušet své znalosti v modelu testů. Tento software byl zvolen pro svou kompatibilitu s webovými prohlížeči a malé velikosti výstupních souborů.

Pro samotný návrh se využije časová osa, kde se za pomoci vrstev a snímků řízených ActionScriptem odehrává veškerá činnost aplikace, která bude publikována pomocí přehrávače Adobe Flash Player v internetovém prohlížeči.

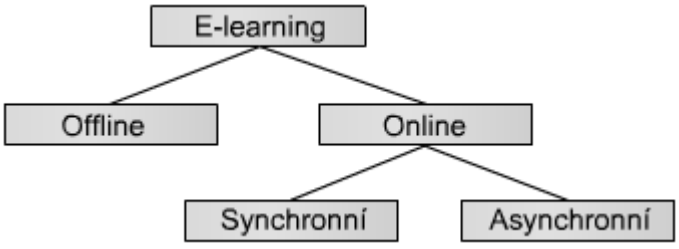
2 Webové aplikace a e-learning

Při začínajícím rozvoji Internetu se webové stránky vytvářely pouze ve statické formě, která značně omezovala interaktivní práci. Na počátku 90. let se začalo s postupným odstraňováním omezení, webové servery umožnily komunikaci s uživatelskými skripty, jejichž mohli návrháři vytvářet jednoduché webové aplikace. Postupem času byly vyvinuty skriptovací jazyky a prostředky, kterými je možné navrhnout plnohodnotný software (ICQ, e-mail klient, grafické editory), který pracuje v internetovém prohlížeči. Webové aplikace se dají využít v mnoha odvětvích, jako prezentace firmy, určitého produktu nebo jako prostředek pro vzdělávání, jenž se nazývá e-learning.

E-learning poskytuje uživatelům možnost vzdělání bez pomoci učitele nebo mnohastránkových knih, prostřednictvím internetu z domova nebo kanceláře. Podstatnou výhodou této formy studia je rychlý přehled o daném tématu v čase, který si uživatel zvolí sám a možnost vyzkoušet si naučené znalostí v testovém modulu.

2.1 Formy e-learningu

Internetová výuka se využívá v několika různých formách, které se dělí podle požadavků na připojení k Internetu, popřípadě k intranetu, na offline a online e-learning. Oba způsoby výuky umožňují obsah výuky prezentovat v elektronické podobě. Při online e-learningu musí být uživatel připojen k počítačové síti, prostřednictvím které získává studijní materiály. Online e-learning se dále může rozdělit na synchronní a asynchronní.



Obr. 2.1: Formy e-learningu

U **synchronní formy** online e-learningu je zapotřebí udržovat stálé připojení k síti. Výuka probíhá v reálném čase prostřednictvím webového rozhraní, za pomoci interaktivních nástrojů, které umožňují komunikaci a spolupráci studentů a vyučujících

ve virtuální třídě v reálném čase. K tomuto druhu výuky se využívají audio a videokonference, chat, instant messaging (ICQ).

Asynchronní podoba e-learningu využívá pro komunikaci e-maily, instant messaging a diskusní fóra, kde mohou komunikovat nejen studenti s vyučujícím, ale i studenti mezi sebou. Výhodou této formy studia je její přizpůsobivost, student si může studijní materiály uložit do počítače a pokračovat ve studiu v offline režimu.

2.2 Výhody a nevýhody

Mezi hlavní výhody e-learningu se řadí:

+ **flexibilita studia** – student má možnost studovat v jakémkoli prostředí a čase, který mu nejvíce vyhovuje
+ **dostupnost** – student má nepřetržitý přístup k materiálům, ze kterých může studovat z jakéhokoli počítače připojeného k Internetu
+ **úspora výdajů** – odpadají výdaje na dopravu při cestě za studiem
+ **individuální tempo** – student si sám volí rychlost studia, k probrané látce se může kdykoli vrátit
+ **interaktivita** – studijní materiály jsou navrženy za pomoci animací, multimediálních textů, které ve studentovi vyvolávají dojem komunikace
+ **testování znalosti** – student si může průběžně ověřovat získané vlastnosti

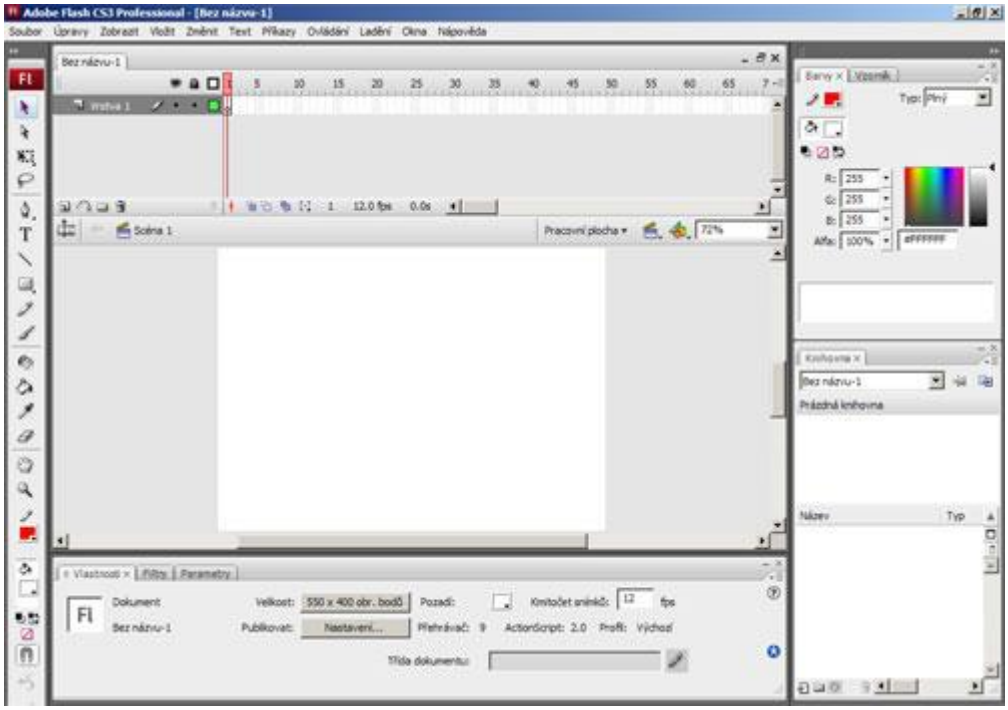
K nevýhodám využívání e-learningu patří:

- **nutnost technologického vybavení** – připojení k Internetu při online výuce
- **tvorba obsahu** – většina studijních materiálů je v tištěné formě, kterou je zapotřebí přepsat do elektronické podoby
- **nepřímá komunikace** – chybí osobní kontakt s ostatními studenty nebo vyučujícím
- **nevhodnost pro některé oblasti vzdělávání**

3 Seznámení s prostředím Adobe Flash CS3

Adobe Flash CS3 je grafický vektorový software, který se převážně používá pro tvorbu interaktivních animací, prezentací, her a v poslední době i aplikací do mobilních zařízení. Díky velmi malé velikosti výsledných souborů (*.swf) nahradil ostatní formáty, které vytváří animaci (*.gif). Jako jeden z mála grafických programů může přímo spolupracovat s ostatními programy firmy Adobe, což urychluje práci nejen při tvorbě webových animací.

Pracovní prostředí Flashe je vytvořeno tak, aby splňovalo požadavky každého uživatele. Skládá se z *hlavního panelu*, *panelu nástrojů*, *uživatelského panelu*, kde si každý uživatel může navolit nejvíce používaná okna pro zrychlení tvorby, a z *časové osy*, na které se odehrává veškerá činnost při tvorbě animace. Všechny panely, které nejsou nastavené pro výchozí zobrazení, lze vyvolat v menu pomocí položky *Okna* (Window).



Obr. 3.1: Uživatelské prostředí Flashe

4 Nástroje a funkce Adobe Flash CS3

4.1 Panel nástrojů

V levé části základní pracovní plochy se nachází panel s jednotlivými kreslícími nástroji. Mezi základní patří šipka, která se používá k výběru a posunu jednotlivých objektů. Laso slouží k přesnějšímu označování výběru objektů i jejich částí. Textový nástroj umožňuje vybrat font, velikost, typ a zarovnání písma. Ke kresbě využívá Flash několik nástrojů. Může vytvářet předem definované tvary (kruh, elipsa, čtverec, mnohoúhelník), každému z objektů lze nastavit barvu výplně, tloušťku a barvu ohraničení. Při kresbě tužkou lze nastavit typ, tloušťku čáry a tvar tuhy (zaoblená, hranatá). Nástroj štětec umožňuje malovat několika způsoby - přes objekt, za objekt, výplň a vyplnit označené.

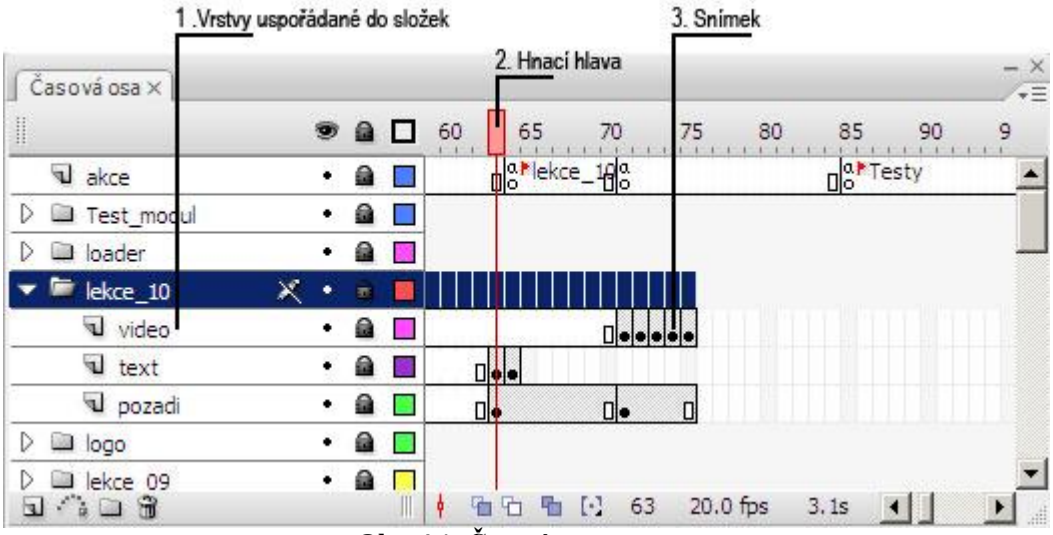
Každému nástroji v panelu lze nadefinovat vlastnosti dle potřeby návrháře.

4.2 Časová osa, vrstvy a snímky

Hlavní částí programu je *časová osa* (Obrázek 4.1), která je uspořádána ve vrstvách a po sobě jdoucích snímcích (frames), které při spuštění animace rozpohybuje hnací hlava (Obrázek 4.1) a vyvolá dojem pohybu. Na časové ose (Time line) se zobrazuje aktuální snímek a rychlost přehrávání (Frame rate), kterou můžeme nastavovat (optimální nastavení je 12 nebo 24 fps).

4.2.1 Vrstvy – Layer

Vrstvy (Obrázek 4.1) slouží pro lepší orientaci v *časové ose*, lze si je představit jako průhledné vrstvy položené na sobě a dohromady vytváří jeden celek. Vrstvy jsou na sobě nezávislé (objekty v různých vrstvách na sebe nemají žádný vliv) a při běhu animace lze objekty v jednotlivých vrstvách programově přesouvat do jiných vrstev.



Obr. 4.1: Časová osa

4.2.2 Snímky – Frames

Snímky jsou hlavní částí pro vytváření děje animace. Pro vytvoření objektů nebo vložení skriptu se používá *Klíčový snímek* (Keyframe). Tento snímek může být plný (obsahuje nějakou grafiku) nebo prázdný. Velikost souboru výsledné animace závisí právě na počtu snímků.

4.3 Scéna

Scéna je plocha, ve které se odehrává průběh animace. Výsledná animace se může skládat z několika scén. Dělení do scén se používá především pro větší přehlednost a také pro oddělení náročnějších animačních prvků.

4.4 Knihovna

Každá animace se skládá z různých objektů. Všechny tyto objekty jsou umístěny v *knihovně* jako symboly. Symboly mohou být trojího typu:

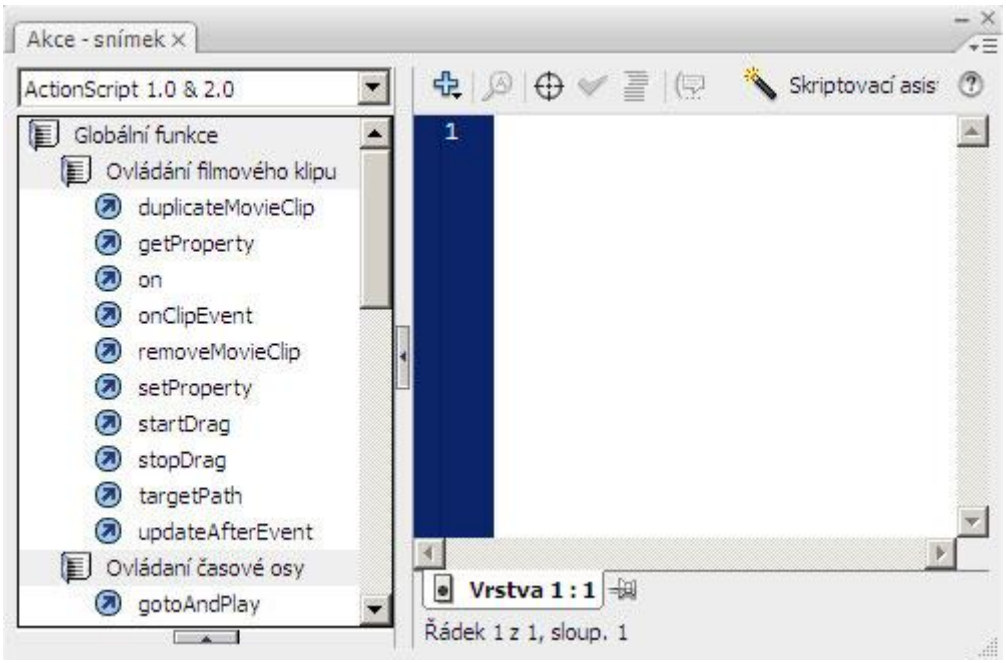
- Movie clip (filmový klip)
- Button (vlastní tlačítka)
- Graphic (vytvořená grafika)

Každý typ pod sebou skrývá velké možnosti dalšího využití a editace. Filmové klipy jsou dynamické, vnořené do animace, kde jejich časová osa je nezávislá na ostatních a chová se jako nová animace. Dají se ovládat pomocí ActionSriptu.

Grafické symboly jsou shodné s filmovými klipy, ale nejsou dynamické a nejdou ovládat pomocí ActionSriptu. Grafické symboly lze umisťovat do filmových klipů a do tlačítek, kde mohou mít vlastní snímky a vrstvy.

Tlačítka mají čtyři stavy (Over, Up, Down a Hit), které jsou znázorněny jako čtyři snímky v časové ose symbolu. Aby tlačítko bylo funkční, musí se k němu přiřadit ActionScript.

4.5 Panel akce a ActionScript



Obr. 4.2: Panel akce

ActionScript je skriptovací jazyk, pomocí kterého můžeme dynamicky měnit obsah scén, reagovat na události, měnit vlastnosti objektů. Všechny skripty se píší v panelu *akce* (Obrázek 4.2), který je aktivní u klíčových snímků, instancí tlačítek nebo instancí filmových klipů. Skript připojený k filmovému klipu reaguje po spuštění události (např. Load, Unload a EnterFrame) a skript připojený k tlačítku reaguje na událost myši (např. Press, Release a Roll Over).

Programování v ActionScriptu se výrazně neliší od programování v jiných jazycích. Prostředí je přizpůsobeno jak pro profesionální programátory, tak pro úplné začátečníky, kterým slouží skript asistent. V tomto modulu jsou připraveny všechny akce, funkce a vlastnosti, které se přiřazují k jednotlivým objektům.

Protože se ActionScript jako ostatní jazyky stále vyvíjí, je důležité si před začátkem programování zvolit správnou verzi.

4.5.1 ActionScript 1.0

Vznikl s vydáním Macromedia Flash 5, kde vznikl nový panel *Action*, pro psaní základních skriptů. Jazyk původně vycházel z JavaScriptu a standartu ECMA-262 [5], od něhož se liší tím, že nepodporuje specifické objekty browseru (Document, Window), nepodporuje kompletně všechny předdefinované objekty JavaScriptu. Vyšší vydání ActionScriptu verze 1.0 přinesla podporu objektového programování.

4.5.2 ActionScript 2.0

Objevil se ve verzi Macromedia Flash MX 2004, kde je podstatně vyvinutější v oblasti objektového orientovaného programování (OOP) a svou syntaxí a zpracováním se podobá JScript.NET a JavaScript 2.0. Plná podpora objektově orientovaného programování umožnila využívat ActionScript ke komplexním aplikacím.

Změny oproti předchozí verzi:

- striktně definované proměnné a vylepšená varování při kompilaci
- „case sensitive“ – rozlišování velkých a malých písmen
- balíčky tříd (packages)
- nový způsob dědění
- get/set metody, které zefektivňují práci s vlastnostmi tříd
- nově zavedeno rozhraní - interface
- nové metody/příkazy pro práci s grafikou

4.5.3 AcitonScript 3.0

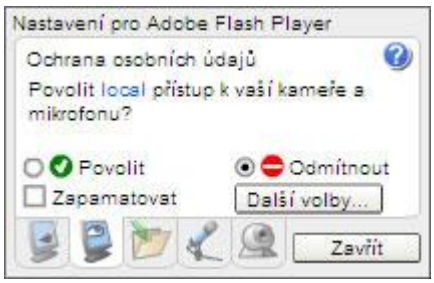
Je to poslední verze ActionScriptu, která prošla kompletním přepracováním. Oproti předchozím se kód zpracovává mnohem rychleji a plně vyhovuje specifikaci ECMAScript. Je podporována od verze Adobe Flash CS3. Změnou prošlo i zapisování Actionscriptu, který se musí psát přímo do snímku na časové ose a nelze jej připojovat přímo k vytvořeným symbolům, důvodem je větší srozumitelnost a lepší údržba psaného kódu. Pro větší přehlednost a rychlejší práci byl celý jazyk upraven do pojmenovaných balíčků, kde uživatel snadno najde požadované funkce a vlastnosti.

Změny oproti předchozí verzi:

- přesná definice co, kdy a jak se bude používat
- větší kontrola nad případným výskytem chyb
- nové události a primitivní typy proměnných
- upravené Flash player API, organizace do balíčků
- vylepšená práce s XML
- vylepšená práce zobrazujících se objektů

5 Flash player

Adobe Flash Player [7] je přehrávač pomocí kterého lze na internetu i mimo něj spouštět interaktivní animace a multimediální obsah. Jeho velikost je minimální a po instalaci lze prohlížet webové stránky a aplikace tvořené v programu Adobe Flash. Aplikace vytvořené pro Flash Player ve většině případů svoji dynamičností a grafickým zpracováním nahradily stránky, vytvořené pomocí html kódu.



Obr. 5.1: Nastavení pro Adobe Flash Player

V současné době je na trhu nejnovější verze Adobe Flash Player 10, která je oproti starším verzím obohacena o Runtime modul, který slouží k vykreslování bitmap a vektorových objektů s mnoha novými vlastnostmi. Do animace a SWF souborů lze za pomoci grafické karty vložit složitější 2D a 3D objekty s texturou. Na tyto objekty lze využít ActionScript, který je uvede do pohybu po křivkách nebo aplikuje různé 3D efekty. Nejedná se však o klasické 3D prostředí, jako v počítačových hrách, protože Flash Player 10 pouze převede 2D objekty do 3D prostředí. Podstatnou výhodou je hardwarová akcelerace prostřednictvím GPU grafické karty, která pracuje nejen u zobrazování 2D a 3D objektů ale také u videa. To vše vede k malé zátěži procesoru počítače. Volný rychlostní prostor pak využívají ostatní aplikace jako webové prohlížeče, které se s přibývajícími otevřenými okny s Flashovými aplikacemi nezpomalují. V této verzi přehrávače jsou k dispozici také 2D efekty realizované

pomocí filtrů podporujících prolínání módů a barev, které využívají technologie Adobe Pixel Blender (výsledná velikost upraveného efektu pouze několik kB).

Zobrazování a vykreslování textů již nezávisí na funkci TextField, jak tomu bylo v předchozích verzích přehrávače a nabízí vlastnosti textových objektů, které zobrazují znaky zprava doleva, svisle orientované texty a nové typografické prvky zahrnující i ligaturu.

Modernizováno:

- Dynamický streaming, který umožňuje dynamicky změnit tok dat u streamů podle aktuálního nastavení a vlastností sítě.
- Správa barev přináší novou možnost konverze SWF souborů do barevného prostoru sRGB, podporu IEC a ICC a možnost online zapnutí a vypnutí správy barev.

6 Konkurenční software

6.1 SilverLight

SilverLight od společnosti Microsoft je stejně jako Flash webový plugin, který umožňuje designérům stránek využívat interaktivní prvky s vektorovou grafikou ve svých webových aplikacích. Nejnovější verze 2.0 byla napsána pomocí jazyků .NET (C#, VB.NET), což umožňuje využívat stejných prvků (proudy, ovládací prvky,kolekce), jak na straně klienta, tak na straně serveru.

Pro vytváření aplikací se využívá standardní programovací jazyk JavaScript, který podporuje každý webový prohlížeč. SilverLight byl původně navržen pro streamování videa v nejvyšší kvalitě HD, která za pomoci kodeku VC-1 disponuje 720 řádky. Aby se svou interaktivností vyrovnal konkurenci, nabízí k práci dvojrozměrné kreslení, kdy obsah je definován jako tvar a cesta, se kterými se může manipulovat na straně klienta.

6.2 JavaFX

JavaFX je nový skriptovací jazyk, který je inovací platformy Java od společnosti Sun Microsystems. Jako ostatní software pro tvorbu interaktivních animací nabízí podporu pro webové prohlížeče i desktopová prostředí. Verze JavaFX 1.0 obsahuje

podporu videa a tři klíčové komponenty, JavaFX Development Environment, JavaFX Production Suite a JavaFX Deskop, které usnadňují práci při návrhu mobilní nebo webové aplikace. Výhodou by mělo být to, že u systémů, které mají nainstalovanou platformou Java, stačí doinstalovat jen doplňkovou knihovnu, a vše by mělo fungovat.

7 Rozdělení kapitol pro e-learning

7.1 Co je to UML a modelování

UML (unifikovaný jazyk pro tvorbu diagramů) je grafický jazyk pro objektově orientovanou analýzu a návrh softwaru. Pro jeho vytvoření se používají grafické prostředky, diagramy. Standardně se používají předdefinované grafické prvky jazyka. Jsou vytvořeny diagramy srozumitelné každému, kdo zná syntaxi a sémantiku. Strukturou tohoto jazyka jsou stavební bloky, které představují základní prvky modelu, relace a diagramy. Dále pak společné mechanismy, což jsou obecné způsoby, jimiž dosáhneme specifických cílů, a architektura, která udává pohled na architekturu systému

7.2 Požadavky

Před začátkem každého grafického návrhu se sestavuje soupis požadavků (inženýrské požadavky), aby bylo jasné, co má daný projekt představovat. Zde se stanoví služby, které systém bude poskytovat uživatelům a omezení, za jakých bude pracovat. Rozlišují se dva druhy požadavků, funkční a nefunkční. Funkční požadavky určují chování systému a nefunkční specifikují nebo omezují podmínky daného systému.

7.3 Use Case (případ užití)

Use Case je sadou scénářů popisujících funkce, které systém provádí prostřednictvím aktérů. Aktér představuje určitou roli, ve které vystupuje uživatel, jiný systém nebo čas. Scénáře popisují spolupráci mezi jednotlivými aktéry a systémem.

Pro určování případů užití neexistuje žádný standard, proto se projde seznam aktérů a stanový se, jak je systém bude používat. Aktéři, kteří spouštějí případ užití po splnění vstupních podmínek, které se týkají omezení systému, se nazývají primárními aktéry. Nejčastěji to jsou uživatelé systému nebo jiný systém.

7.4 Modelování tříd a objektů

Modelování slouží jako nástroj pro zkoumání reálných systémů, pomocí vytvoření obrazu dané reality brané z určitého úhlu pohledu. Základem modelování jsou třídy a objekty, které zjednodušují představu daného jevu.

Objekt představuje seskupení dat a funkcionality. Každý má svou identitu, vlastnosti chování a zodpovědnost. Všechny objekty vycházejí z nějaké třídy, která je pro ně šablonou. Všechny třídy jsou definovány atributy, kterým jsou při vzniku instancí objektu přiřazeny skutečné hodnoty, a operacemi. Atributy mají název (pojmenování dané vlastnosti objektu), formát a nastavení viditelnosti (veřejný přístup, soukromý přístup, chráněný přístup). Operace s jednoznačnou a unikátní signaturou určují statickou strukturu střídy.

Mezi jednotlivými třídami existují vzájemné vztahy, které blíže vysvětlují jejich chování.

7.4.1 Asociace

Asociace reprezentuje vztahy mezi jednou a více třídami. Jejichž pomocí může jeden objekt využívat služeb jiného objektu. V návrhových diagramech se znázorňuje jako rovná čára, která spojuje dané třídy. Asociace je pojmenována slovesem (třídí, vyhledává). Pro určení počtu objektů, které mezi sebou mohou komunikovat, slouží multiplicita.

1:1 … jeden objekt komunikuje právě s jedním objektem

1:N … jeden objekt komunikuje s více objekty

M:N … více objektů komunikuje s více objekty

7.4.2 Agregace a kompozice

Agregace vyjadřuje vztah, kdy jedna třída nebo objekt je součástí druhé třídy nebo objektu. Speciálním typem agregace je kompozice, při které podřízený objekt nemůže existovat bez nadřízeného objektu.

7.4.3 Generalizace a specializace

Generalizace neboli zobecňování umožňuje objektovým třídám sdílet jejich charakteristiky včetně hierarchie dědění a uchovat jejich rozdíly. Používá se především

z důvodu opětovného využití prvků. Generalizace je vztah mezi objektovou třídou (super class nebo parent) a jejími konkrétními potomky (subclass nebo child). Pro všechny třídy nebo objekty platí, že podřízené třídy a objekty dědí všechny vlastnosti svého předka. Specializace označuje opačnou vazbu.

7.4.4 Zapouzdření

Zapouzdření představuje vztah, kdy data a operace objektu tvoří nedělitelný celek a jsou na sobě závislé. Zajišťují soudržnost objektů (objekty nejsou ve stavech, v kterých neodpovídají realitě). Velké množství objektů je pro zjednodušení skryto.

7.4.5 Polymorfismus (mnohotvarost)

Polymorfismus představuje odlišnou reakci stejné zprávy zaslané různým objektům.

7.5 Relace

V návrhovém modelu relace slouží k významovému propojení prvků mezi sebou. Mezi jednotlivými prvky modelu jsou různé vazby.

- Asociace – relace mezi aktéry a případy užití
- Generalizace, <<include>>, <<extend>> - relace mezi případy užití
- Zobecnění – relace mezi aktéry

Vazby mezi objekty se označují jako **spojení** (link), které umožňuje vzájemné předávání zpráv. V různých objektově orientovaných jazycích jsou spojení znázorňována odlišně. Při existenci spojení mezi dvěma objekty musí existovat i nějaká spojitost mezi jejich třídami. Pro jejich spojení se využívá asociace.

Asociace slučuje jednotlivé instance a asociace třídy. Její syntaxe obsahuje název asociace, název role, násobnost a průchodnost. Názvy asociací jsou popsány slovesnými frázemi s příponou nebo předponou, která určuje směr asociace. Proti tomu názvy rolí představují, jaké role objekty jednotlivých tříd mají, jestliže jsou spojeny s instancemi asociace. Mezi asociacemi existují určitá omezení, nejrozšířenějším typem je násobnost, která omezuje počet objektů dané třídy, které se dané relace účastní v určitém okamžiku.

Závislost je relace mezi dvěma prvky modelu, kdy specifikace jednoho prvku (dodavatele) může ovlivnit jiný prvek (klient), který jej používá. Vyskytuje se mezi balíčky a balíčky, mezi objekty a třídami a mohou se vyskytnout i mezi operací a třídou.

7.6 Dědičnost a polymorfismus

7.6.1 Dědičnost (generalizace/specializace)

Dědičnost je vztah, kdy jedna třída sdílí část struktury a chování v jiné třídě a zároveň má vlastnosti které se liší. Potomci dané třídy dědí všechny vlastnosti svého předka (atributy, operace, relace, omezení), ale mohou přidat i novou charakteristiku a předefinovat operace svých předků. Jestliže chce potomek předefinovat operaci svého předka, musí mít jeho operace zcela stejnou signaturu, jako má příslušná operace jeho předchůdce.

7.6.2 Polymorfismus

Polymorfismus umožňuje předávat objektům různých tříd tu samou zprávu a příslušné objekty se chovají, tak jak se od nich čeká. Polymorfní operace představují operace s mnoha implementacemi, proto objekty reagují na zprávy, které jsou volány operacemi specifikované jejich třídou. Výsledek volání závisí na stavu příslušného objektu v daném okamžiku.

7.7 Stavové a sekvenční diagramy

7.7.1 Stavové diagramy

Stavové diagramy popisují stavy, ve kterých se může systém v určitých okamžicích nacházet. Změnu stavu vyvolávají události, které se jich dotýkají. Především se používají pro znázornění chování objektů v Use Case, pro tvorbu tříd se složitým chováním a lze je kombinovat s jinými technikami modelování (diagramy interakce).

Každý stavový diagram musí začínat počátečním stavem Start a končit stavem Stop. Stavy znázorňují situaci, která se stala a vyhovuje určité podmínce v období života objektu, mohou mít jakýkoli počet akcí a interních přechodů. Zahrnují vstupní a výstupní akce, které jsou spojeny s událostmi entry a exit.

- Entry – proběhne po spuštění při přechodu do daného stavu objektu a současně spustí asociovanou vstupní akci
- Exit – proběhne jako poslední událost určitého stavu objektu a spustí asociovanou vstupní akci

Jednotlivé stavy jsou propojeny přechody, akce a aktivita. Akce jsou nepřerušitelné rychle probíhající procesy, které slouží pro přechod mezi jednotlivými stavy. Proti tomu aktivita je spojena se stavem, může se přerušit a trvá určitou dobu. Důsledkem přechodů vznikají události, které se dělí do několika skupin.

- Událost volání – odpovídá metodě dané třídy (požadavek spuštění) a spouští sérii akcí
- Signální událost – přijímá asynchronně předávané zprávy mezi objekty, signály se modelují jako samostatné třídy se stereotypem, informace uloženy v atributech
- Události změny – skládá se z klíčového slova when, podmínky a akce, událost aktivována po splnění logické podmínky
- Časové události – uvozeny klíčovými slovy when a after, událost vygenerována po určité době (např.: after(10minut))

7.7.2 Sekvenční diagramy

Sekvenční diagram popisuje zasílání zpráv mezi jednotlivými objekty systému v závislosti na čase. Každý objekt má vlastní časovou posloupnost. V diagramu jsou znázorněny třídy objektů a jejich instance z různých logických úrovní návrhu.

K ohraničení množiny zpráv se speciálním významem se využívají fragmenty, které se mohou skládat z více částí. Existují různé typy fragmentů.

- Alt (alternativa)
- Opt (podmíněný fragment)
- Par (paralelní fragmenty)
- Loop (opakování)
- Ref (reference, odkaz)

7.8 Diagramy aktiv

Diagramy aktiv jsou varianty stavového diagramu, které v sobě kombinují různé modelovací techniky a znázorňují sekvenci aktiv, které podporují sekvenční a paralelní chování. Diagramy se skládají z elementů: akce, přechody, hodnocení přechodů, větvení a spojení a plaveckých drah, které zodpovídají za konkrétní třídy, osoby nebo oddělení.

Akce s jedním vstupním a výstupním přechodem se nemohou dále dělit a jsou jádrem aktivit (jakákoli činnost), které musí být dokončeny. Přechod mezi jednotlivými stavy, jenž reprezentuje vlákno, nastane hned po skončení akce. K výstupním vláknům lez přidat podmínku (podmíněné vlákno). Ta se vyhodnocuje při aktivaci vlákna.

Diagramy aktiv se používají především pro schopnost modelovat paralelní procesy a podporu paralelního chování. Jejich nedostatkem je nedostatečné znázornění vazeb mezi akcemi a objekty.

Používají se:

- pro analyzování případů užití
- pro porozumění business procesům
- pro modelování paralelních vláken a k synchronizaci

Nepoužívají se:

- pro znázornění spolupráce objektů
- pro znázornění chování objektů v průběhu jejich životního cyklu

7.9 Datové modelování

7.9.1 Logický datový model

Při navrhování logického datového modelu se nebere v úvahu konkrétní typ relační databáze. Model se skládá z entit (datový objekt), které představují logické seskupení dat zkoumané v rámci analýzy (např.:dokument faktura). Entity jsou mezi sebou propojeny vazbami 1:1, 1:N nebo M:N. Každá entita má své atributy určeny formátem a velikostí (např.: Character 10), které ve fyzickém návrhu znázorňují sloupce tabulky. Některé atributy mají významnější úlohu než ostatní, nazývají se klíče. V návrhu se využívají primární klíče, alternativní klíče a cizí klíče. Primární klíče musí být jednoznačné, většinou to je typ číslo nebo krátký řetězec. Cizí klíče se v jiných

entitách vyskytují jako primární klíče. Při správném určení klíčů, běžící systém zpracovává data mnohem rychleji. Pro sestavení datového návrhu se postupuje podle následujících kroků:

- identifikace entit
- identifikování klíčů
- určování vazebních vztahů
- vytvoření datového modelu
- odstranění redundancí a duplicit

7.9.2 Fyzický datový model

Při převodu z logického datového modelu do fyzického návrhu databáze se využívá proces normalizace, jejíž technika je založena na přesných matematických pravidlech navržených tak, aby chránila data a tvořila pružnější databázi. Normalizaci si lze představit jako relační datovou analýzu, tabulku, která je uspořádána do řádků a sloupců, kde sloupec (doména) znázorňuje kategorii dat a řádky hodnoty. Sloupce musí mít jednoznačné pojmenování a alespoň jeden musí být definován jako primární klíč.

Pro normalizaci databáze existuje několik pravidel, které se nazývají, normální forma. Formy postupují od nižších k vyšším, kdy každá vyšší v sobě zahrnuje formy nižší. Při převodu dat do 1. normální formy se nejprve najde jednoznačný klíč, který se převede s opakující se skupinou dat do nové tabulky, kde se stane cizím klíčem. Tabulka převedená do 1. normální formy nesmí obsahovat opakující se skupiny dat.

Tab. 1: Tabulka pro převod do 1. normální formy

Jméno	**Adresa**
Petr Novák	Havlíčkova 256, 293 01 Mladá Boleslav

Tab. 2: Tabulka převedená do 1. normální formy

Jméno	**Příjmení**	**Ulice**	**Město**	**PSČ**
Petr	Novák	Havlíčkova 256	Mladá Boleslav	293 01

U převodu do 2. normální formy se vytvoří tabulky pro data, kam se přesunou sloupce, které jsou závislé pouze na části klíče. Zde se stane klíčem ta část složeného klíče, na které je sloupec závislý. Nové tabulky nesmí obsahovat sloupce závislé jen na části složeného klíče, nejlépe když jsou závislé na primárním klíči nebo kompletním složeném klíči tabulky, *Tabulka 5* (atribut Telefonní_linka není závislý na celém klíči, ale pouze na oddělení)

Tab. 3: Tabulka pro převod do 2. normální formy

Příjmení	**Oddělení**	**Plat**	**Věk**	**Telefonní_linka**
Novák	Příjem zboží	25.000	33	2213

Tab. 4: Tabulka převedená do 2. normální formy

Příjmení	**Oddělení**	**Plat**	**Věk**
Novák	Nákup	25.000	33

Tab. 5: Tabulka převedená do 2. normální formy

Oddělení	**Telefonní_linka**
Příjem zboží	2213

Tabulka je v 3. normální formě, jestliže každý neklíčový atribut není přímo závislý na žádném klíči. Je nutné odstranit závislosti mezi datovými sloupci a vzájemné závislosti v rámci klíče. Po převedení tabulky je nutné zjistit vzájemné závislosti, popřípadě vytvořit ze závislých sloupců novou tabulku, *Tabulka 4* a *5* (atribut Patro a Počet pracovníků nejsou funkčně závislé na klíčových atributech Jménu a Příjmení, ale na atributu Oddělení)

Tab. 6: Tabulka pro převod do 3. normální formy

Jméno	**Příjmení**	**Telefon**	**Oddělení**	**Patro**	**Počet pracovníků**
Petr	Novák	606 342 454	Doprava	33	10

Tab. 7: Tabulka převedená do 3. normální formy

Jméno	**Příjmení**	**Telefon**	**Oddělení_ID**
Petr	Novák	606 342 454	1

Tab. 8: Tabulka převedená do 3. normální formy

Oddělení_ID	Oddělení	Patro	Počet pracovníků
1	Doprava	33	10

Pro urychlení načítání dat do tabulek se používají indexy, které se využívají v dotazech, kde se provádí spojení tabulek nebo dotazy. Indexy se vytváří na sloupcích, které jsou primárním klíčem, cizím klíčem nebo často používané a na sloupcích podle kterých se třídí.

Ve vytvořených databázích jsou uloženy programy a procedury, které jsou spustitelné externími programy a vykonávají funkce nad daty (kopírování, mazání, výpočty). Procedury se starají o údržbu databáze a poskytují výsledky na základě dotazů do několika tabulek. Jsou spouštěny pomocí triggerů, což je softwarový prostředek, který vykonává příkazy na úrovni tabulky.

Pro návrh samotného datového modelu z diagramu tříd se využívá mapování tříd, kde se atributy tříd stanou sloupci a instance objektů odpovídají řádkům tabulky.

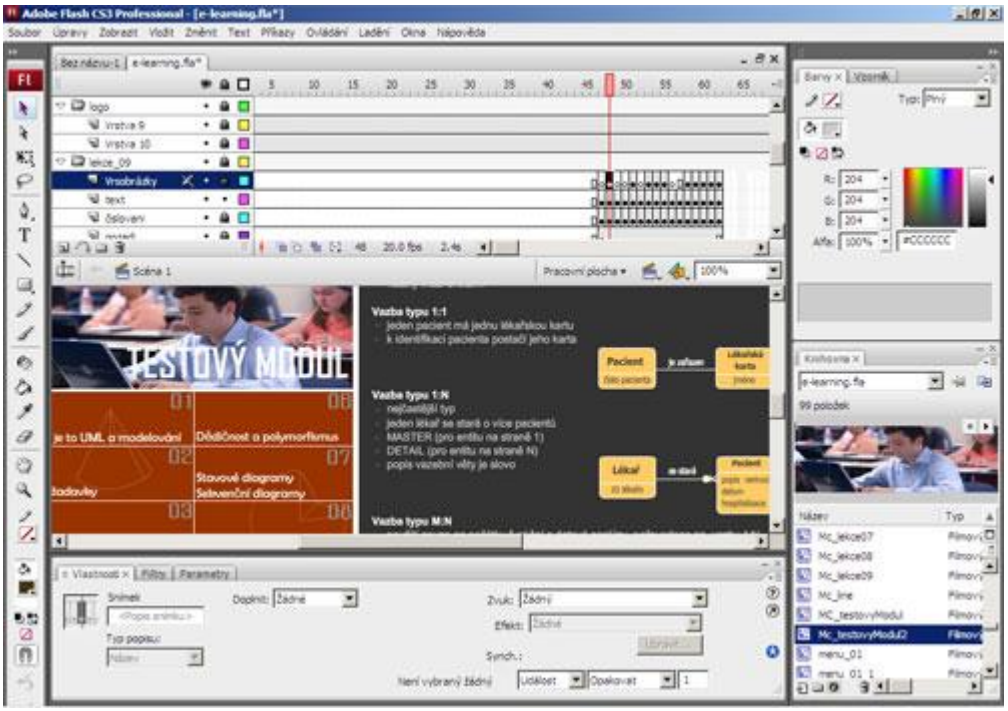
7.10 Praktické postupy

Pro realizaci praktický postupů byl použit návrhový software Enterprise Architect od společnosti SPARX Systems. Pomocí tohoto softwaru byla vytvořena instruktážní videa, která zachycují základní postupy návrhu modelu.

Všechny video soubory byly uloženy ve formátu FLV, který v sobě kóduje data zvuku a videa. Tyto soubory nejsou součástí aplikace, ale jsou uloženy pro rychlejší načítání samostatně. Při spuštění videa se obsah nahraje do vyrovnávací paměti, která umožňuje u větších souborů plynulý přechod snímků.

8 Tvorba aplikace v Adobe Flash CS3

Celá aplikace e-learningu byla vytvořena tak, aby se dále mohla vyvíjet. Proto byla pro její realizaci použita jedna scéna, kde je každá kapitola znázorněna jako jedna složková vrstva na *časové ose*. Aby aplikace byla interaktivní, byly všechny skripty psány pomocí ActionScriptu 2.0, který lze přiřazovat přímo jednotlivým snímkům nebo objektům.



Obr. 8.1: Tvorba aplikace

8.1 Načítání aplikace

Přestože výsledná velikost výstupního souboru swf je velmi malá, při publikaci větších souborů na Internet dochází k jejich sekání a trhání. Swf soubor se načítá snímek po snímku a může se stát, že v aplikaci na úvodním snímku bude tlačítko, které bude odkazovat na ještě nenačtený snímek. Při kliknutí na toto tlačítko se aplikace zastaví nebo zbortí. Aby se zabránilo těmto potížím, byl vytvořen ***preloader***, který zajišťuje plynulejší práci v aplikaci.

Preloader je objekt, který porovnává stav načtených dat s celkovým datovým objemem publikovaného swf souboru. Do doby načtení celého souboru informuje o stavu načtených dat a pro udržení pozornosti obsahuje jednoduchou animaci, která udrží pozornost uživatele.

Načtení celé e-learningové aplikace se provádí v prvním snímku *časové osy*. Aby se hnací hlava zastavila, musel se k prvnímu snímku přiřadit příkaz stop(), který se zapsal do panelu *akcí*. Po té se na scéně vytvořil nový symbol (filmový klip) „loader“, jenž představuje animaci načítání. Do vytvořeného symbolu se vložilo dynamické textové pole, kde se procentuelně znázorňuje stav načtené aplikace. Nakonec se v panelu *akcí* k symbolu „loader“ zapíše skript pro načtení aplikace.

```
onClipEvent (load) {                    /* po načtení se preloader zmenší */
 preloadbar._xscale = 0;
}
onClipEvent (enterFrame) {              /* pohyb preloaderu */
 celkem = _root.getBytesTotal();
 nahrano = _root.getBytesLoaded();
 procNahrano = nahrano/(celkem/100);
 procent = Math.round(procNahrano);
 preloadstatus.text = ""+procent+"%";
 preloadbar._xscale = procNahrano;
 if (procent == 100) {                  /* po načtení celé aplikace přeskočí */
 _root.gotoAndStop(2);                  /* hnací hlava na úvodní stránku */
 }                                       /* e-learningu na snímku 2 */
}
```

8.2 Práce na časové ose

Celá aplikace e-learningu je uspořádána do 18 složkových vrstev a 101 snímků, které představují grafické prostředí. Pohyb po ose znázorňuje hnací hlava, která se plynule pohybuje po ose, pokud není do snímku vložen ActionScript, který změní její pohyb.

Použité příkazy ActinScriptu, které mění pohyb po ose:

- gotoAndPlay(1); hnací hlava přejede na snímek 1 a spustí animaci
- gotoAndStop(2); hnací hlava přejede na snímek 2 a zastaví animaci
- nextFrame(); hnací hlava přejede na následující snímek
- play(); spuštění animace
- prevFrame(); hnací hlava přejede na předchozí snímek
- stop(); zastavení animace

8.3 Navigační menu

Navigační menu je tvořeno deseti tlačítky, která představují jednotlivé kapitoly. Tlačítka byla vytvořena jako symbol tlačítko (button), která mají vlastní *časovou osu* (Obrázek 9.1). Jejich časová osa je vytvořena ze čtyř základních snímků reagujících na

pohyb a akci vyvolanou ukazatelem myši tím, že přeskočí na daný snímek. Každý ze základních snímků má specifickou funkci.

- Up (Nahoře) - znázorňuje vzhled tlačítka, když na něm není ukazatel myši
- Over (Přes) - znázorňuje vzhled tlačítka, když na něm je ukazatel myši
- Down (Dole) - znázorňuje vzhled tlačítka, když je stisknuto
- Hit (Zásah) - definuje oblast, která bude reagovat na kliknutí myši

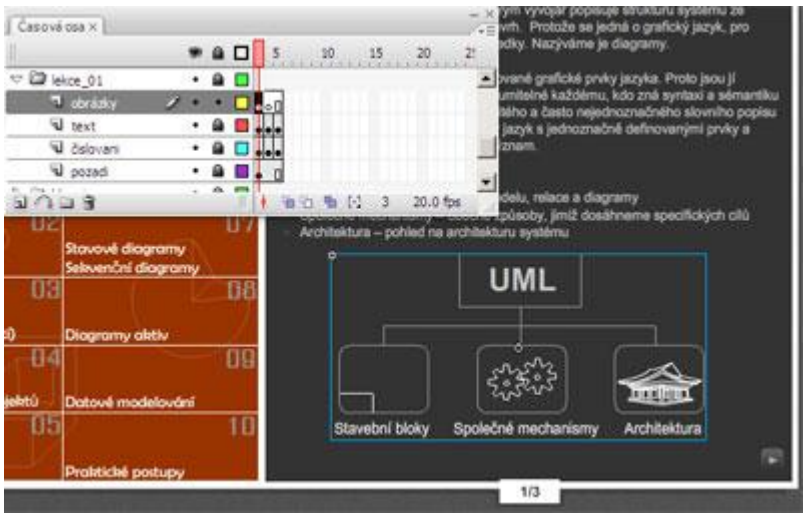
Pro všechna tlačítka byl vložen do panelu *akce* skript, který představuje pohyb po časové ose, kdy po stisknutí tlačítka on(press) se hnací hlava převine a zastaví se na definovaném snímku gotoAndStop().

Například:

```
on (press) {
        gotoAndStop(30);
}
```

8.4 Tvorba a úprava kapitol

Tvorba jednotlivých kapitol v e-learningu se od sebe příliš neliší. Pro návrh první kapitoly byly použity čtyři vrstvy (pozadí, číslování, text, obrázky). Jednotlivé vrstvy jsou uspořádány tak, aby se navzájem nepřekrývaly. Kapitolu lze vytvořit i v jedné vrstvě, ale úpravy by byly zdlouhavé a nepřehledné.



Obr. 8.2: Návrh 1. kapitoly

V nejnižší vrstvě bylo vytvořeno pozadí kapitoly, které je pro všechny snímky stejné. K jeho tvorbě bylo zapotřebí označit klíčový snímek (klávesa F6), ve kterém se pomocí nástroje *obdélník* nakreslily plné obdélníky bez okrajů představující pozadí. Následující vrstva „číslování“ znázorňuje číslo stránky, na které se návštěvník nachází. V případě první kapitoly byl obsah rozdělen do tří jednotlivých snímků, kde statické textové pole obsahuje číslo (např.: 2/3).

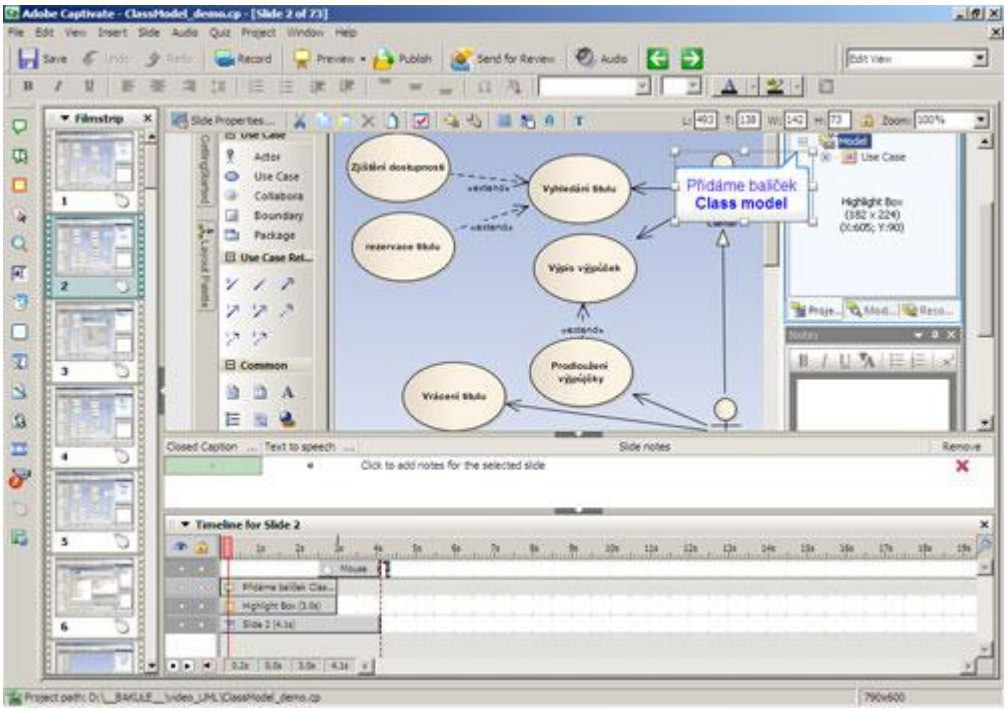
Hlavní vrstvou každé kapitoly je vrstva „text“, obsahující výukový text a grafickou úpravu stránky (odrážky a navigaci). Do této vrstvy bylo dále vloženo statické textové pole s názvem a číslem kapitoly. Snímky v poslední vrstvě slouží pro kreslení a vkládání obrázků nebo animací. Pro znázornění struktury jazyka UML byla v úvodní kapitole vytvořena jednoduchá animace, která využívá vlastností symbolu tlačítka.

Ve vrstvě „obrázky“ byl na klíčovém snímku vytvořen symbol (klávesa F8) s názvem MC_struktura, uvnitř kterého se nachází čtyři předdefinované snímky (Obrázek 9.1). V prvním snímku je zobrazena statická část animace. Druhý snímek, který představuje pohyblivé části animace, se spustí po najetí kurzoru na obrázek struktury.

8.5 Instruktážní videa

Instruktážní videa byla vytvořena za pomoci několika softwarů. Hlavním byl Enterprise Architect, který slouží pro návrh vývojových diagramů a schémat pro vývoj aplikací. Software využívá jazyka UML a podporuje generování zdrojových kódů programovacích jazyků C++, C#, Java, Delphi, ActionScript. Pro zachycení obrazovky při tvorbě diagramů byla použita trial verze programu Adobe Captivate 4. Pracovní prostředí Adobe Captivate 4 je založeno na platformě Adobe Flash. Díky tomu je možné vytvořený interaktivní obsah přehrávat v Adobe Flash Playeru.

Nahrávání obrazovky probíhalo v reálném čase, to znamená, že při vytváření návrhu diagramu se zachycovaly jednotlivé pohyby kurzoru do animací a nově vyvolané události se uložily jako slidy. Jednotlivé slidy jsou stejně jako v Adobe Flash CS3 rozděleny do snímků a vrstev na *časové ose* (Obrázek 8.2), kde se dají dále upravovat. Výsledkem je prezentace, která vytváří dojem videa.



Obr. 8.3: Návrhové prostředí Adobe Captivate

Jednotlivé prezentace byly pomoci výstupního formátu videa (MSScrean 9 encoder DMO) vyexportovány do video souborů avi, ze kterých byly převedeny do FLV souborů, jež je možné vkládat do Flash aplikací.

8.6 Testový modul

Testový modul byl rozdělen podle výukových kapitol. Pro všechny kapitoly byl vytvořen souhrn otázek, které představují základní informace z daných kapitol. Každý test byl v návrhu rozdělen do několika snímků, kde každý snímek představuje jednu otázku a po jejím zodpovězení hnací hlava přeskočí na další snímek. Přitom se správné odpovědi ukládají do proměnné, ze které se pak vytvoří výsledky. Pro označení správné odpovědi bylo vytvořeno interaktivní tlačítko, které se při kliknutí zaškrtne a do proměnné se přičte jedna.

První snímek:

```
stop();                 /* pozastavení hnací hlavy */
odpoved=0;              /* nadefinování proměnné */
```

Poslední snímek:

```
procento= (odpoved/5)*100;          /* procento úspěšnosti */
pocet_dobrych= odpoved;             /* počest správných odpovědí */
pocet_spatnych= 5-odpoved;          /* počet špatných odpovědí */
stop();                             /* pozastavení hnací hlavy */
```

9 Publikace webové aplikace

Publikování aplikace na Internet bylo provedeno ve dvou fázích. Nejprve byly nastaveny parametry pro publikování html a swf souborů, které se zobrazí výběrem:

- *Soubor (File) > Nastavení publikování (Publish Settings)*

Zde bylo možné nastavit různé vlastnosti, které informují o použitých prostředcích při realizaci aplikace (např.:verze Flash Playeru, verze ActionScriptu) nebo nějakým způsobem ovlivňují chod aplikace (např.:kvalitu výstupního souboru, rozměry aplikace).

Vytvořený html soubor využívá automatické zjištění verze Flash Playeru, které bylo nastaveno při publikaci. Zjišťování je přímo začleněno do obsahu html stránky a v případě, že uživatel používá starší verzi, informuje o tom a nabídne mu stažení novější verze flashplayeru. Po nastavení publikace byla aplikace vyexportována do html stránky:

- *Soubor (File) > Publikovat (Publish)*

Příkaz „publikovat“ vyvolal spuštění internetového prohlížeče, do kterého byla načtena vytvořená webová aplikace. Zdrojový kód stránky obsahuje dvě hlavní části. Část pro zobrazení aplikace ohraničenou tagy *OBJECT*, které obsahují nastavené atributy (HEIGHT, WIDTH, CLASSID a CODEBASE), a část pro aktivaci Flash aplikace, která je tvořena javascriptem uloženém v externím souboru.

HTML kód pro načtení aplikace:

```
<object
        classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
        codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab
        #version=9,0,0,0"
         width="800"
        height="600"
        id="e-learning"
        align="middle">
```

```
        <param name="allowScriptAccess" value="sameDomain" />
        <param name="allowFullScreen" value="false" />
        <param name="movie" value="e-learning.swf" />
        <param name="quality" value="high" />
        <param name="bgcolor" value="#cccccc"/>
    <embed src="e-learning.swf" quality="high" bgcolor="#cccccc" width="800"
  height="600" name="e-learning" align="middle" allowScriptAccess="sameDomain"
  allowFullScreen="false" type="application/x-shockwave-flash"
  pluginspage="http://www.macromedia.com/go/getflashplayer" />
    </object>
```

Atribut CLASSID slouží pro identifikaci ovládání ActiveX v prohlížeči. ActiveX je ovládací prvek, který je spustitelný pouze z hostujících aplikací (Internet Explorer), pomocí něhož lze přistupovat k místnímu systému souborů a měnit nastavení registru operačního systémů. Pokud nastane situace, kdy ActiveX ještě není nainstalován, může ho internetový prohlížeč pomocí atributu CODEBASE automaticky stáhnout. Samostatné tagy PARAM představují vlastnosti nastavené ve Flashi při publikování aplikace.

Protože v posledních verzích Internet Exploreru se musí aplikace do webové stránky ručně aktivovat, generuje se spolu s exportovaným souborem swf další soubor, *AC_RunActiveContent.js*, který obsahuje javascript pro zjištění prohlížeče a verze flashplayeru. Ve zdrojovém kódu stránky jsou zobrazeny všechny atributy a odkazy výsledné aplikace.

```
    <!--url's used in the movie-->
    <a href="http://www.sparxsystems.com/"></a>
    <!--text used in the movie-->
    <!-- saved from url=(0013)about:internet -->
    <script language="javascript">
        if (AC_FL_RunContent == 0) {
            alert("This page requires AC_RunActiveContent.js.");
        } else {
            AC_FL_RunContent(
        'codebase',
'http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=9,0,0,0',
        'width', '800',
        'height', '600',
        'src', 'e-learning',
        'quality', 'high',
        'pluginspage', 'http://www.macromedia.com/go/getflashplayer',
        'align', 'middle',
        'play', 'true',
        'loop', 'true',
        'scale', 'showall',
        'wmode', 'window',
```

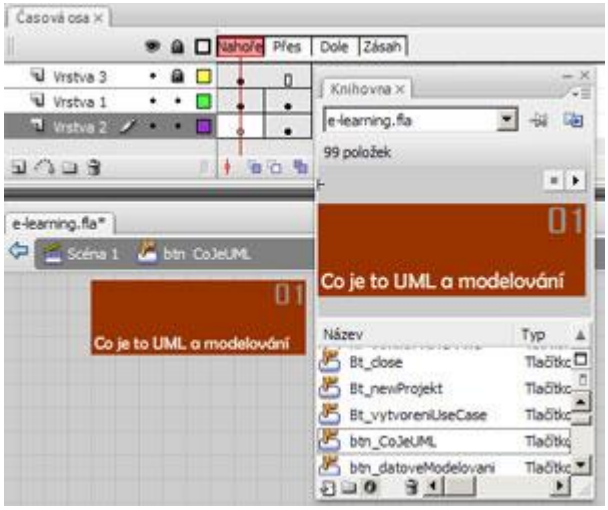
```
            'devicefont', 'false',
            'id', 'e-learning',
            'bgcolor', '#cccccc',
            'name', 'e-learning',
            'menu', 'true',
            'allowFullScreen', 'false',
            'allowScriptAccess','sameDomain',
            'movie', 'e-learning',
            'salign', ''
            );
            }
</script>
```

Aby výsledná aplikace byla plně funkční, musí se do hlavičky html stránky přidat skript pro načtení *AC_RunActiveContent.js* souboru.

```
<script language="javascript">AC_FL_RunContent = 0;</script>
<script src="AC_RunActiveContent.js" language="javascript"></script>
```

U ostatních internetových prohlížečů jako jsou Firefox nebo Opera je nutno nainstalovat plug-in, který zajistí správné zobrazení flashové aplikace. Pomocí OBSERVERU (monitorovací systém) lze zjistit verzi nainstalovaného plug-in Flash v prohlížeči, popřípadě nainstalovat novější verzi.

10 Využití e-learningu pro jiný předmět



Obr. 10.1: Úprava tlačítka

Vytvořená webová aplikace může sloužit i jako template (šablona), který lze využít pro prezentaci jiného předmětu. Template je nástroj, který uchovává základní vzhled a vrstvy. Přizpůsobuje se požadavkům návrháře bez větších znalostí kódu, v jakém je vytvořen.

V případě využití šablony e-learningu pro jiný předmět se musí nejprve upravit název tlačítek. Toho se docílí otevřením panelu knihovna, kde jsou zobrazeny všechny vytvořené symboly, a upravením textu tlačítka na daném snímku (Obrázek 10.1).

Při vkládání nového textu se upravuje v příslušné kapitole vrstva text, popřípadě vrstva obrázek, kde je znázorněn obsah. Veškerá úprava se pak provádí v jednotlivých snímcích.

Závěr

Cílem bakalářské práce bylo seznámit se s unifikovaným jazykem pro tvorbu diagramů a vytvořit e-learning na téma UML a Objektový návrh, který bude prostřednictvím webové aplikace publikován na webu.

Pro samotný návrh aplikace bylo nutné se seznámit s programovacím jazykem ActionScript a grafickým softwarem Adobe Flash CS3. Pomocí těchto prostředků bylo vytvořeno interaktivní prostředí e-learningu, které obsahuje několik kapitol vysvětlujících základní prvky UML, instruktážní videa a testový modul pro ověření získaných znalostí.

Výsledná aplikace byla uspořádána do 18 složkových vrstev, které vytvářejí výukové prostředí prostřednictvím 101 snímků. Pro lepší pochopení probrané látky bylo v programu Enterprise Architect vytvořeno pět instruktážních videí, které obsahují základní principy tvorby návrhových diagramů grafickým jazykem UML. Pro dobrou kvalitu videí bylo potřeba vyzkoušet několik výstupních video formátů a zvolit ten nejvhodnější. Vyexportovaná videa byla uložena jako externí soubory, které nejsou součástí aplikace a při jejich volání jsou postupně načítány do vyrovnávací paměti.

Výsledkem praktické části je funkční webová aplikace, která by se mohla dále rozvíjet. Za použití stejných prostředků lze přidávat nové kapitoly a praktické příklady, ale mnohem praktičtější bude využívat aplikaci jako šablonu pro návrh e-learnigu jiného předmětu.

Přílohy

Příloha A

A.1: Grafická podoba e-learningu

Literatura

[1] *LUPA: Také E-learning se vyvíjí* [online]. [2009-01-05] URL:<http://www.lupa.cz/clanky/konference-o-e-learningu/>

[2] *Formy E-learningu* [online]. [2009-01-05] URL:<http://h41156.www4.hp.com/education/article.aspx?cc=cz&ll=cs&id=959>

[3] *Adobe Flash CS3* [online]. [2009-01-10] URL:<http://www.adobe.com/products/flash/>

[4] *Voxcafe* [online]. [2009-01-10] URL:<http://www.voxcafe.cz/clanky/webcasting/video-format-flv.html>

[5] *Standard ECMA-262* [online]. [2009-05-20] URL:< http://www.ecma-international.org/publications/standards/Ecma-262.htm>

[6] *Adobe ActionScript 3.0* [online]. [2009-01-20] URL:<http://help.adobe.com/cs_CZ/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7d48.html>

[7] *Adobe – Flash Player: Nejčastější dotazy* [online]. [2009-02-15] URL:<http://www.adobe.com/cz/products/flashplayer/faq/>

[8] Kanisová, Hana; Müller, Miroslav: *UML srozumitlně.* Computer Press Brno, 2004

[9] Arlow, Jim; Neustadt, Ila: *UML 2 a unifikovaný proces vývoje aplikací*, Computer Press, a. s. Brno, 2008