OD MODELU K SIMULÁTORU V INTERNETOVÉM PROHLÍŽEČI

Pavol Privitzer, Jan Šilar, Martin Tribula, Jiří Kofránek

Abstrakt

V práci je popsána originálně vyvinutá technologie tvorby simulátorů spustitelných v internetovém prohlížeči. Základem tvorby simulačních modelů je vývojové prostředí akauzálního simulačního jazyka Modelica. Vlastní simulátory jsou vytvářeny v prostředí Microsoft .NET. Animace, ovladatelné modelem na pozadí, jsou vytvářeny pomocí nástroje Microsoft Expression Blend. Simulační jádro modelu je automaticky vygenerováno překladačem jazyka Modelica pomocí námi originálně vytvořeného generátoru kódu C# a propojeno s řešičem algebrodiferenciálních rovnic. Celý simulátor je vytvořen jako aplikace pro platformu Microsoft Silverlight. To umožní v různých operačních systémech spouštět simulátory jako webové aplikace, spustitelné pomocí internetového prohlížeče s nainstalovaným doplňkem Silverlight.

Klíčová slova

e_Learnig, Model, Simulace, Web

1. Úvod

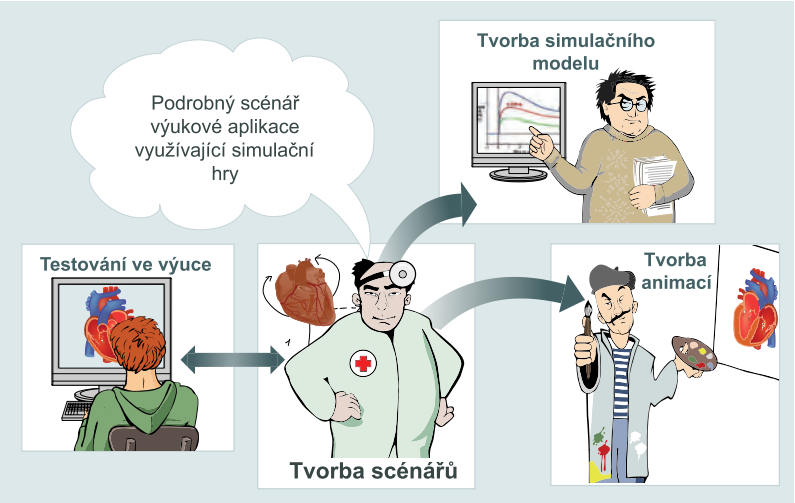
Základem výukového simulátoru je identifikovaný simulační model. Simulační model, implementovaný v tom nejrafinovanějším vývojovém prostředí ale není sám o sobě použitelný jako výuková pomůcka. Je jen implementací formalizovaného popisu modelované reality umožňující testovat chování matematického modelu při nejrůznějších hodnotách vstupů a hledat takové rovnice a parametry modelu, které by ve zvolených mezích přesnosti nakonec zajistily dostatečně dobrou shodu chování modelu s chováním modelovaného systému (identifikace modelu)

I po dosažení tohoto cíle je od identifikovaného modelu k výukovému simulátoru ještě poměrně dlouhá cesta. Díky pokroku v softwarových technologiích se však objevily nové nástroje nejen pro efektivnější vytváření simulačních modelů, ale také i prostředky, usnadňující tvorbu vlastních simulátorů s působivým vizuálním uživatelským rozhraním.

2. Kostra výukové simulační aplikace – scénář

Simulační technologie se zlepšily a zefektivnily, ceny zařízení i softwaru padají. Rozvoj internetu, virtuální 3D prostředí, jakým je např. prostředí Second Life, nebo stále širší nabídka lékařských trenažérů využívajících robotizovanou figurínu pacienta přinášejí nové možnosti pro lékařskou výuku. Výuka pomocí simulátorů se v poslední době velmi rozšiřuje (zejména v USA a v Izraeli).

Výuka se simulátorem je z pedagogického hlediska vysoce účinná, klade však mnohem větší nároky na vyučujícího, než klasická výuka**.** Sebesložitější simulátor s tím nejatraktivnějším uživatelským rozhraním nemusí proto nutně být účinnou výukovou pomůckou. Pedagogická efektivita využití simulátoru



Obrázek 1 - Úloha pedagoga – tvůrce scénáře ve vývoji výukových simulátorů je klíčová. Autor scénáře musí ve spolupráci s tvůrcem modelu přesně definovat, jakým způsobem se bude model ve výukové aplikaci využívat a jak bude vypadat uživatelské rozhraní modelu. Pro výtvarníka musí připravit dostatečně podrobné zadání požadavků na výtvarné ztvárnění všech potřebných grafických prvků a animací. Důležité je testování vytvořeného simulátoru ve výuce, které obvykle přinese cenné podněty pro modifikaci výukové aplikace.

především závisí na pedagogovi, který musí mít jasné představy, jakým způsobem a kde je nejvhodnější využít ve výuce simulační model.

Čím složitější je výukový simulátor, tím důležitější je mít předem jasnou představu o scénáři výukových simulačních her, které s ním budeme provádět. To potvrzují i naše praktické zkušenosti s využitím složitých simulátorů ve výuce (např. se simulátorem Golem [4] nebo se simulátorem QCP [1]). Ukázalo se, že uživatelské rozhraní, které při mnoha větvících se nabídkách a možnostech prohlížení hodnot stovek proměnných, studenty zbytečně rozptyluje. Bez jasného pedagogického vedení, na co a kdy se při dané simulační hře se složitým simulátorem podívat a jak interpretovat výsledky, má jejich využití mizivé výsledky.

O tom, jak může být simulátor pedagogicky využíván, je ovšem třeba přemýšlet především před tím, než se pustíme do jeho vytváření. To platí zejména, chceme-li vytvářet multimediální interaktivní výukové programy, dosažitelné po internetu, využívající simulační hry pro lepší pochopení a procvičení vykládané látky. Klíčem k úspěchu je dobrý scénář (*Obrázek 1*).

První, na kom závisí úspěch vytvářené aplikace, je tedy zkušený pedagog, který musí mít jasno v tom, co a jakými prostředky chce svým studentům pomocí multimediální výukové aplikace vysvětlit, kde a jak využít pro ozřejmění vykládané látky simulační model.

Kostrou výukové aplikace je její scénář. Jeho základem je obvykle nějaký výukový text – skripta, kapitola v učebnici apod. Při tvorbě scénáře pro multimediální výukovou aplikaci však musíme myslet i na to, jak se bude výukový program jevit na obrazovce, jaká bude posloupnost jednotlivých obrazovek, jaké bude jejich výtvarné ztvárnění, kde budou umístěny interaktivní elementy, kde se bude zapojovat zvuk, jak budou vypadat jednotlivé animace, kde se vloží simulační model a jak bude ovládán, kde se vloží test znalostí, jak bude vypadat, jak se bude vyhodnocovat a jak se bude reagovat na jeho výsledek apod.

Finální podobu grafických prvků tvoří profesionální výtvarník. Proto je nesmírně důležitá dobrá komunikace pedagoga - tvůrce scénáře s výtvarníkem. Pedagog nemusí umět dostatečně dobře kreslit, musí mít ale jasnou a promyšlenou představu o tom, co od výtvarníka potřebuje. Kamenem úrazu se zpočátku stávalo neustálé předělávání již nakreslených animací, způsobených obvykle původně ne zcela jasnou představou pedagoga o výtvarném ztvárnění multimediálního prvku ve scénáři. Věnovat pozornost pečlivé přípravě scénáře před počátkem vlastní realizace se proto vyplatí.

Při tvorbě scénáře výukové aplikace se nám osvědčilo využívat postup, který je znám u kresleného filmu – nakreslit (nejlépe ve spolupráci s výtvarníkem) obrazový scénář, tzv. „Story Board" – hrubou posloupnost jednotlivých obrazovek, a ke každé obrazovce pak v klasickém textovém editoru napsat komentář (případně odkaz na příslušnou část textu).

Interaktivní multimediální program však nejsou do počítačové podoby jednoduše přepsaná skripta. Není to ani lineární posloupnost textů, zvuků a pohyblivých obrázků jako kreslený film. Výrazným rysem výukového programu je jeho interaktivita – a s tím spojená možnost větvení a vzájemného propojení jednotlivých částí. Přetvořit lineární textový a obrazový scénář do scénáře větveného, hypertextovými odkazy provázaného interaktivního programu ovšem není jednoduché.

Jedním z metodologických problémů, který bylo nutno při tvorbě scénářů našich výukových aplikací vyřešit, byl způsob jak ve scénáři zobrazit vlastní strukturu výukového programu, zahrnující výklad, interakce s uživatelem, větvení programu apod.

Nejjednodušší je v textovém či obrazovém editoru pomocí klasických blokových diagramů či struktogramů popsat příslušná větvení, rozhodovací bloky apod. s příslušnými odkazy na stránky textu a příslušné obrázky uložené v dalších souborech.

Při psaní scénáře se osvědčilo využít schopností moderních textových editorů a vytvářet příslušné hypertextové odkazy – tím již vlastní scénář dostává jisté rysy budoucí interaktivity.

Pro zápis scénáře výukové aplikace jsme také vyzkoušeli nástroj Adobe Captivate (*http://www.adobe.com/cz/products/captivate/*), který umožňuje rychle vytvářet profesionálně vypadající e-learningový obsah s pokročilou interaktivitou, bez velkých nároků na znalosti programování. Ukázalo se ale, že

pro cíl – zapsat scénář ve formě interaktivně se větvícího storyboardu, je tento nástroj až příliš zbytečně komplikovaný. Naopak, ukázalo se, že pro sdílení postupně vznikajících scénářů mezi členy vývojového týmu stačí jednoduché nástroje. Pro specifikaci zadání vytváření grafických prvků výtvarníkům i sledování dosažených výsledků jejich práce apod., velmi postačoval nástroj Microsoft OneNote.

Moderní interaktivní výukový program není ani do počítačové podoby převedený výukový animovaný film – nejvyspělejším rysem interaktivity, kterou počítač poskytuje, je právě možnost využití simulátoru, který formou simulační hry umožní ve virtuální realitě ozřejmit vysvětlovaný problém. A scénář výukového programu s tím musí počítat. Autor si musí položit otázky – jaké experimenty se simulačním modelem je vhodné nabídnout, jaké má být uživatelské rozhraní simulační hry, a konečně, jaké jsou požadavky na simulační model na pozadí.

Proto je důležité, aby „scénárista" komunikoval i s tvůrcem modelu, aby znal strukturu modelu a mohl navrhnout její případnou modifikaci, či formuloval požadavky, které by model měl splňovat.

Klíčová je i pedagogická zkušenost. Nezřídka se ukazuje, že to, co se při návrhu scénáře výukové aplikace jevilo jako jednoduché a srozumitelné, může být v pedagogické praxi komplikované a obtížně pochopitelné. Kromě toho, právě při využívání simulační hry ve výuce se vyjeví nutnost modifikace ať již uživatelského rozhraní, nebo i simulačního modelu na pozadí aplikace.

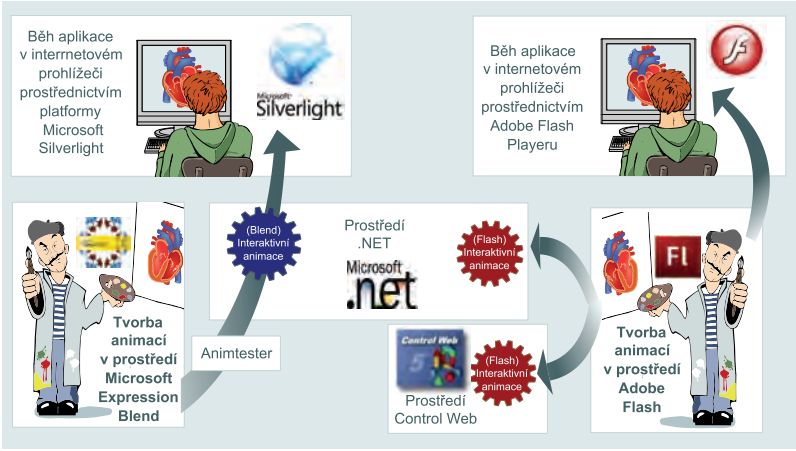
Proto je důležité průběžně vytvářet návrhy scénářů v těsném kontaktu s pedagogickou praxí. Proto se nám také osvědčilo si simulační aplikaci„nejdříve vyzkoušet na studentech" při výuce, a poučeni z této zkušenosti pak vytvářet nebo dotvářet vysvětlující text, modifikovat simulátor a navrhovat konečnou formu uživatelského rozhraní v produkční verzi internetové e-learningové aplikace.

3. Svaly výukové simulační aplikace – interaktivní multimediální komponenty

Pro vytváření uživatelského rozhraní výukového simulátoru je velmi působivé simulátor navenek reprezentovat jako pohyblivé obrázky řízené simulačním modelem. Řízené animace mohou graficky reprezentovat význam číselných hodnot – např. schematický obrázek cévy se může roztahovat nebo komprimovat, srdce může rychle či pomaleji tepat, plíce mohou hlouběji či mělčeji „dýchat", ručička měřicího přístroje se může pohybovat a průběžně zobrazovat hodnotu nějaké výstupní proměnné modelu čtené z běžícího simulačního modelu na pozadí apod. Na druhé straně můžeme přes vizuální prvky (přes nejrůznější tlačítka, knoflíky, táhla apod.) do simulačního modelu zadávat nejrůznější vstupy.

Grafický vzhled v nezanedbatelné míře rozhoduje o tom, jak bude výuková aplikace přijímána potenciálními uživateli.

Pro profesionální výsledný vzhled výukového simulátoru je proto nezbytné,



Obrázek 2 - Na bedrech výtvarníků spočívá odpovědnost za tvorbu multimediálních komponent, zejména interaktivních animací, propojených se simulačním jádrem výukových simulátorů. Animační komponenty vytvářené v prostředí Adobe Flash využíváme v numericky méně náročných simulátorech spustitelných v internetovém prohlížeči nebo jako prvky uživatelského rozhraní simulátorů vytvářených ve vývojových prostředích Control Web nebo Microsoft .NET. V poslední době využíváme též nástroj Microsoft Expression Blend pro tvorbu grafických komponent pro multimediální simulátory spustitelné prostřednictvím platformy Silverlight přímo v internetovém prohlížeči. Pro usnadnění tvorby animací, které budou řízeny simulačním modelem, využíváme námi vytvořený nástroj Animtester.

aby vlastní animace vytvářel výtvarník – výsledky jsou neporovnatelně lepší, než když animace vytváří graficky nadaný programátor. Předpokladem ale je, že výtvarník musí dobře zvládat práci s nástroji pro tvorbu interaktivní grafiky. Takto vzdělaných výtvarníků je ale na trhu práce kritický nedostatek a ti, kteří tyto nástroje ovládají, jsou žádanými (dobře placenými) členy profesionálních týmů vytvářejících počítačové hry, webové portály, reklamní multimediální aplikace apod.

Znamenalo to ovšem věnovat nemalé úsilí výuce výtvarníků, které jsme tyto dovednosti museli naučit. Proto jsme již před lety začali úzce spolupracovat se Střední uměleckou školou Václava Hollara a na této škole jsme otevřeli „laboratoř interaktivní grafiky" jako detašované pracoviště Univerzity Karlovy.

Věnovali jsme značnou část našeho pracovního času tomu, abychom profesionální výtvarníky naučili pracovat s vývojovými nástroji pro tvorbu interaktivních animací (jako je Adobe Flash, Microsoft Expression Blend aj.), s nástroji pro tvorbu 3D grafiky (např. Cinema 3D), s nástroji pro zpracování videa (např. Adobe Premiere), poskládat vše do internetem dostupné aplikace (např. pomocí Adobe Flex) i naučit výtvarníky základům programového ovládání interaktivních animací a tvorby interaktivních webových stránek.

Naše úsilí mělo úspěch. Výtvarníci ztratili ostych před počítačem a záhy

pochopili, že „digitálním štětec" představuje jen další nástroj pro kreativní výtvarné vyjadřování, jehož ovládnutí navíc přináší další velké možnosti jejich pracovního uplatnění.

Iniciovali jsme také založení Vyšší odborné školy, která vyučuje v tříletém studiu předmět „interaktivní grafika". Pracovníci našeho Oddělení biokybernetiky a počítačové podpory výuky se v této Vyšší odborné škole mimo jiné podílejí i na výuce (*http://www.hollarka.cz/*).

Výtvarník musí na jedné straně spolupracovat s pedagogem, vytvářejícím scénář výukové aplikace, a na druhé straně s programátorem, který zajistí adekvátní chování interaktivních animací (např. interaktivní animace propojí se simulačním modelem). Proto i výtvarník musí mít i některé základní programátorské znalosti, aby komunikace s programátorem byla efektivní (*Obrázek 2*).

V minulosti jsme pro vizuální rozhraní simulátorů (např. simulátoru Golem) využívali softwarové prostředí Control Web, původně určené pro vizualizaci a řízení průmyslových procesů, které nabízí řadu předpřipravených prvků – virtuálních přístrojů, z nichž šlo velmi pohodlně a rychle sestavit uživatelské rozhraní. Za rychlost a snadnost sestavení se ale platilo tím, že námi vytvořený simulátor měl spíše tvar řídícího pultu technologického procesu než interaktivní obrázek z lékařské učebnice. S příchodem profesionálních výtvarníků do našeho vývojového týmu se možnosti vizualizace značně posunuly a ve výukových programech jsme již nebyli omezováni nabídkou předpřipravených prvků. Mohli jsme vytvářet libovolné tvary animovaných komponent, které mohly být propojeny se simulačním modelem a jako loutky řízeny hodnotami na výstupech či vstupech simulačního modelu.

Pro vytváření interaktivních multimediálních komponent, propojitelných se simulačním modelem na pozadí, využíváme dva nástroje:

1. Prvním z nich je Adobe Flash, v němž můžeme vytvářet animované interaktivní komponenty, jejichž chování se dá programovat (a v našich aplikacích propojit se simulačním modelem). Vytvořené komponenty se dají na platformách různých operačních systémů (s využitím volně stažitelného doplňku internetového prohlížeče - Adobe Flash Playeru) snadno přehrávat v prohlížeči internetových stránek. Ve Flashi se tak dají vytvářet i numericky méně náročné simulátory spustitelné přímo v prohlížeči webových stránek. Flashové komponenty jsme také využívali jako vizuální rozhraní komunikující s jádrem simulátoru (prostřednictvím ActiveX komponent) v simulátorech vytvořených v prostředí Control Web a v prostředí .NET.
2. Druhým nástrojem, který jsme začali využívat pro tvorbu grafických komponent simulátorů teprve v nedávné době, je vývojové prostředí Microsoft Expression Blend. Toto prostředí umožňuje (s využitím vývojového prostředí Microsoft .NET) vytvářet aplikace přímo spustitelné v internetovém prohlížeči prostřednictvím volně stažitelného doplňku Microsoft Silverlight. Platforma Microsoft Silverlight umožňuje běh rozsáhlých numericky náročných aplikací s interaktivním multimediálním

rozhraním. Tato nová platforma Microsoftu umožňuje pomocí internetu distribuovat numericky náročné multimediální simulátory spustitelné přímo v internetovém prohlížeči.

4. Animační štětec pro výtvarníky - Adobe Flash

Flash prošel poměrně dlouhým vývojem. Původně to byl především program od firmy Macromedia určený pro pouhé vytváření animovaných obrázků. Postupně se rozšiřovala možnost vytvářené animace řídit pomocí skriptů i ve Flashi, jejichž syntaxe se postupně vyvíjela a obohacovala. Od verze 7 (prodávané pod názvem Macromedia Flash MX 2004) byl již do prostředí Flash zabudován již skutečně objektový řídící jazyk (ActionScript), syntaxí velmi podobný jazyku Java, který umožňuje poměrně pohodlné a sofistikované řízení chování vizuálních interaktivních elementů.

Velký úspěch vývojového prostředí Macromedia Flash je mimo jiné založen na tom, že tvůrcům se poměrně úspěšně podařilo definovat rozhraní pro výtvarníky (vytvářející základní animační prvky) a programátory, kteří těmto komponentám mohou pomocí výše zmíněného objektového jazyka vdechnout interaktivnost.

Základní komponentou Flashových aplikací je film (movie). Film je možné dělit na jednotlivé scény, které je možno postupně či programově (na přeskáčku) přehrávat. Scény jsou tvořené sekvencí jednotlivých snímků (frames). Filmy (movies) je možno libovolně řetězit – z každého filmu (movie) je možné programově zavolat zavedení dalšího filmu, a pak spustit jeho přehrávání. To má výhodu zejména v internetových aplikacích, kdy po zavedení a spuštění první části animace se na pozadí z příslušného serveru stahuje její další část.

Při vytváření počítačových animací se vychází z klasického způsobu tvorby kresleného filmu, kde se pro každé políčko kreslí jednotlivé součásti na průhledných průsvitkách vrstvených na sebe. Některé průsvitky se nemusí pro další políčko překreslovat, nebo se jen o málo posunou (např. pozadí), zatímco jiné (např. postavička) se na každém políčku překreslují, buď celé, nebo jen zčásti, aby postupně vznikl animovaný pohyb.

Ve Flashi je každá scéna tvořena několika vrstvami, které fungují obdobně jako průsvitky při ruční tvorbě kresleného filmu.

Každé políčko filmu, resp. scény (frame) má tedy několik vrstev, do nichž se ukládají jednotlivé obrazové prvky. Tyto vizuální prvky se mohou v každé vrstvě samostatně kreslit – Flash obsahuje poměrně výkonný nástroj pro vytváření vektorových obrázků (vektorové obrázky je samozřejmě možné i importovat z mnoha jiných externích kreslících programů). Jinou možností je vybrat si z knihovny příslušný obrázek a vytvořit jeho instanci. Instancí však nemusí být jenom statický obrázek. Instancí může být i tzv. filmový klip (MovieClip), který je vlastně instancovanou třídou již dříve vytvořeného filmu. Tak například při vytváření obrázku letadla můžeme jako jeden jeho element do jedné z vrstev vložit z knihovny instanci filmového klipu s točící se vrtulí. Vlastní MovieClip tak může mít i poměrně složitou hierarchickou strukturu

– film, který ho vytváří, může obsahovat instance dalších filmových klipů. Např. MovieClip auta může v sobě obsahovat MovieClipy jeho točících se kol. Každá instance MovieClipu má své vlastnosti (souřadnice umístění na scéně, velikost, přebarvení, průhlednost apod.), které je možné dynamicky měnit z programu. Kromě toho, třída MovieClip má celou řadu metod, které můžeme využívat (např. metodu pro detekci kolize z jinou instancí MovieClipu na scéně apod.).

Při tvorbě MovieClipu můžeme také naprogramovat specifické metody, které pak můžeme volat u všech jeho instancí. Tak je možné naprogramovat složité chování vizuálních komponent. Poměrně jednoduše je možno vytvářet speciální MovieClipy jako skutečné komponenty, a jejich vlastnosti pak nastavovat ve speciálním editoru komponent a za běhu volat jejich metody. To otevřelo možnost vytváření (a následné distribuci či prodeji) nejrůznějších vizuálních (ale i nevizuálních) komponent od řady tvůrců a přispělo k velkému rozšíření Flashe mezi výtvarnou komunitou.

Ve vývojovém prostředí je možné jednotlivé filmy vytvořit (jak po grafické, tak i po programátorské stránce), otestovat a přeložit do mezijazyka (ve formě .swf souboru), který je možno interpretovat pomocí volně stažitelného interpretu (tzv. Flash Playeru) a přehrávat buď jako samostatně spustitelnou animaci nebo ji prohlížet přímo v internetovém prohlížeči.

Kromě toho je možné vytvořený .swf soubor interpretovat pomocí speciální ActiveX komponenty, kterou je možno zabudovat do jiného programu – např. do aplikace vytvářené ve Control Webu nebo v Microsoft Visual Studiu. Důležité je, že tato komponenta si s aplikací může vyměňovat zprávy a tak můžeme jinou aplikací pohodlně řídit chování interaktivní animace. Aplikace zároveň může od interaktivní animace přijímat zprávy, referující o zásazích uživatele.

Velký úspěch Flashe vedl k tomu, že firma Macromedia byla zakoupena firmou Adobe a Flash se stal jednou z integrálních částí portfolia nástrojů počítačové grafiky tohoto výrobce.

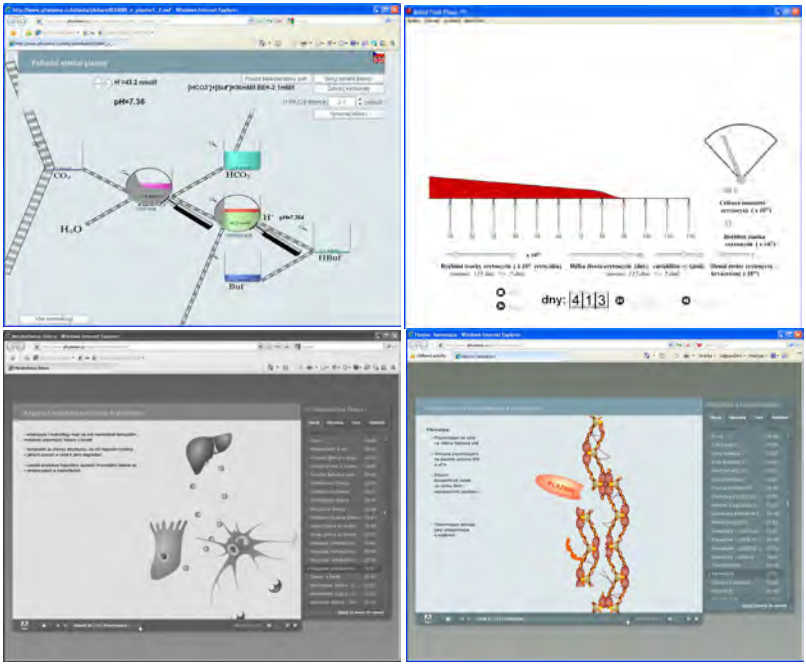
Flashové komponenty se nyní dají využít v tzv. RIA (Rich Internet Application) – nové generaci multiplatformních webových aplikací s propracovaným designem komplexního uživatelského rozhraní, vytvářených pomocí nástroje Adobe Flex či jako desktopové aplikace vytvářené pomocí nástroje Adobe Air.

Rychlost interpretu „.swf" souborů (flash playeru) se zvýšila a v jazyku ActionScript je možné vytvářet i vlastní simulační jádro výukových simulátorů. Výhodou čistě flashových výukových aplikací (které mohou být i složité RIA aplikace, zkomponované v prostředí Adobe Flex***)*** je to, že je možno je spouštět přímo z internetového prohlížeče (který má příslušný plugin) a na všech platformách vypadají stejně.

V tomto prostředí jsme vytvořili některé výukové simulátory a výukové multimediální interaktivní aplikace se simulačními hrami (*Obrázek 3*).

Flash je také současnou platformou, v níž je implementován náš Atlas fyziologie a patofyziologie [7, 8].

Prostředí Flash Playeru je nicméně stále ještě prostředím, založeným na interpretaci flashových .swf souborů. Pro numericky náročnější výpočty



Obrázek 3 - Prostředí Adobe Flash je možné využít pro implementaci simulátorů spustitelných přímo v internetovém prohlížeči - např. model acidobazické rovnováhy plazmy nebo model anémie. Flash byl doposud i hlavním implementačním prostředím pro ozvučené výkladové kapitoly našeho internetového Atlasu fyziologie a patofyziologie. Ve výkladových kapitolách jsou animace plně synchronizovány se zvukem. Pomocí jezdce umožňují výklad libovoně zastavovat, posouvat či navracet a přitom zůstává plná synchronizace animaci.

u složitějších simulátorů zde narážíme na určitou bariéru výkonu. Pro komplikovanější simulátory prostředí Adobe Flash samo o sobě (zatím) nestačí.

Složitější simulátory jsou vytvářeny v prostředí .NET (a v minulosti i v prostředí Control Web). Flashové komponenty do takto vytvářených simulátorů začleňujeme prostřednictvím komponenty Active X. Překlenování dvou nesourodých světů Adobe Flash a .NET ovšem klade další nároky na (ruční) programátorskou práci.

5. Microsoft Expression Blend - nástroj pro tvorbu „grafických loutek" pro simulátory

Pro grafické aplikace je ale možné využít platformu Microsoft .NET přímo (a obejít se bez propojování s komponentami Adobe Flash). Díky technologii WPF - Windows Presentation Foundation [13] je dnes možno přímo v platformě .NET vytvářet složité grafické komponenty obsahující animace, vektorovou

grafiku, 3D prvky apod. Grafické prvky je možné vytvářet obdobně jako v prostředí Adobe Flash, ale s potenciálně většími možnostmi ovládání jejich chování než v prostředí Adobe Flash.

Krom toho, Microsoft reagoval na hojně rozšířený doplněk internetových simulátorů Adobe Flash vytvořením vlastní platformy Silverlight, umožňující, obdobně jako Flash Player, spouštět složité aplikace kombinující text, vektorovou i bitmapovou grafiku, animace a video v internetovém prohlížeči. Aplikace primárně běží v internetovém prohlížeči, aniž by ji bylo nutno zvlášť instalovat (jediná potřebná instalace je samotný Silverlight plugin). Pomocí malé stažitelné komponenty tedy Silverlight umožňuje interaktivní ovládání aplikací ve většině současných webových prohlížečů (Internet Explorer, Firefox, Safari) na různých hardwarových a softwarových platformách. Přímo jsou nyní podporovány operační systémy Windows a Mac pro nejpoužívanější prohlížeče. Pro Linux je vyvíjená plně kompatibilní open source implementace Moonlight. Aplikace vytvořené pro tuto platformu využívají podstatnou část .NET frameworku, který je součástí pluginu (a tudíž mohou provádět i poměrně složité výpočty).

Silverlight je tedy platformou umožňující přes internet distribuovat simulátory, které mohou běžet přímo v internetovém prohlížeči (a to ***i*** na počítačích s různými operačními systémy – stačí aby prohlížeč měl instalován příslušný doplněk).

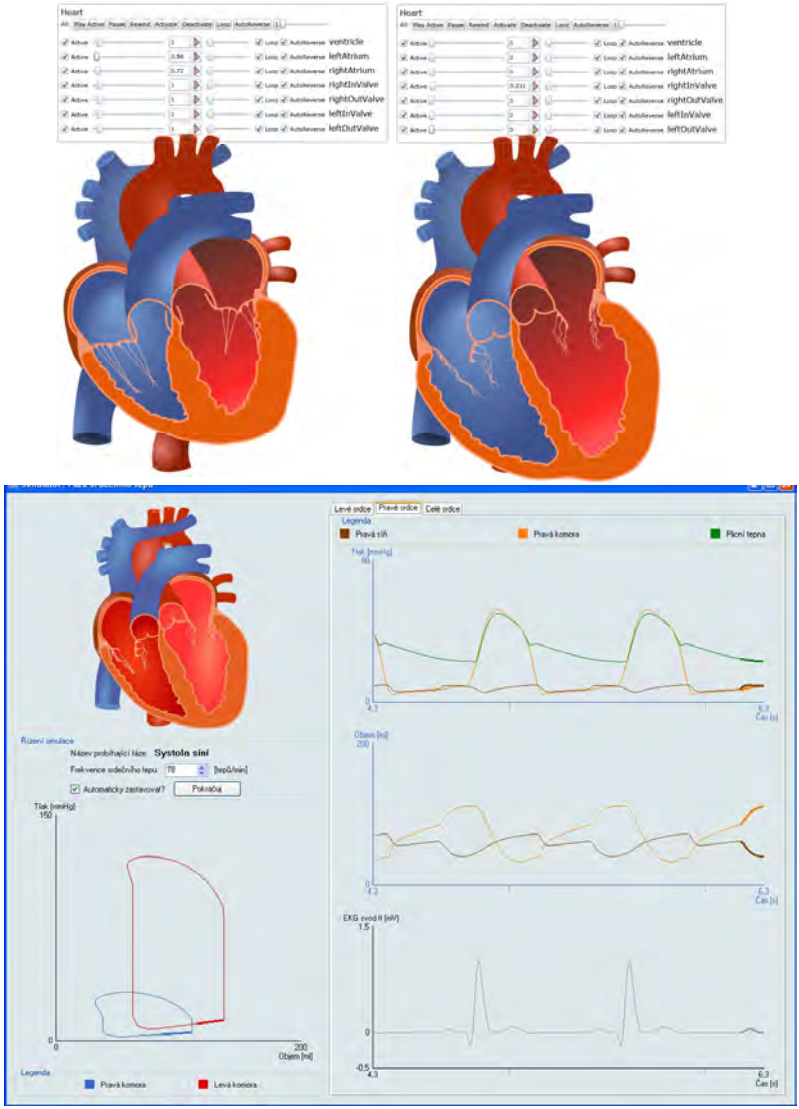
Důležitou vlastností Silverlightu je, že v sobě zahrnuje nativní podporu animací [12]. Animace jsou tedy přímo součástí aplikací a není potřeba pro grafickou vrstvu používat další dodatečnou platformu (např. Adobe Flash).

Pokročilejší je i způsob animace. V prostředí Adobe Flash se animace řídí přehráváním jednotlivých snímků zadanou rychlostí přehrávání. Když se při přehrávání animace na klientském počítači nestačí včas všechny snímky vykreslit, přehrávání některých snímků se přeskočí a animace je „trhaná". V Silverlightu se ale animace řídí přímo časovou osou. To mimo jiné vede k plynulejšímu přehrávání animací, protože snímková rychlost přehrávání se plynule nastaví podle zdrojů na klientském počítači, kde je animace přehrávána.

Microsoft vytvořil i nástroj pro pohodlnou tvorbu grafických prvků a tvorbu animací – **Microsoft Expression Blend**. V tomto nástroji můžeme vytvářet grafické rozhraní pro Silverlight.

Microsoft Expression Blend má rozhraní pro výtvarníka i pro programátora. Microsoft Expression Blend zároveň pracuje přímo nad projektem vytvářené aplikace ve Visual Studiu .NET [15]. Tím eliminuje potřebu převádět návrhy od designéra do projektu aplikace, což podstatně zjednodušuje spolupráci programátora a výtvarníka.

V prostředí Microsoft Expression Blend jsou klíčové snímky animací vytvářeny nikoli po jednotlivých políčkách filmu (jako u Adobe Flash), ale podle časové osy. Nakreslené animace se pak dají vyjádřit jako vlastnosti objektů a tvar animovaného objektu se dá nastavit hodnotou odvozenou z časové hodnoty animace. Tím můžeme zadáním hodnoty vytvořené vlastnosti ovládat tvar animovaného grafického prvku.



Obrázek 4 - Animace tlukoucího srdce. Výstupy modelu ovlivňují fáze tlukotu srdce, otevírání a zavírání chlopní atd. Nad vlastní animací jsou pomocné ovládací prvky Animtesteru umožňující grafikovi ladit jednotlivé subanimace. Grafik je plně odstíněn od programování. Ve výsledném simulátoru pak za příslušná táhlíčka obrazně řečeno „tahá" simulační model naprogramovaný na pozadí.

Vytvořený ovladatelný animovaný grafický objekt můžeme použít i jako komponentu pro další, složitější animovaný objekt a jeho tvar ovládat nastavováním hodnot příslušných vlastností. Můžeme tak vytvořit animační „loutku", jejíž výsledný tvar závisí na momentálním nastavení hodnot jejich jednotlivých komponent.

Pro snadnější komunikaci výtvarníka s programátorem, implementujícím vlastní simulátor, i pro komunikaci výtvarníka s autorem návrhu výukové aplikace, jsme vytvořili pomocný softwarový nástroj Animtester [8], s jejichž využitím mohou designéři-grafici tyto animační „loutky" vytvářet a ladit bez nutnosti dalšího programování. Animtester se vloží jako komponenta do vývojového nástroje Microsoft Expression Blend a umožní z vytvořených animací vygenerovat vlastnosti grafického prvku ovládané zvenčí pomocí ovládacích prvků (šoupátek a tlačítek). V následně vygenerované aplikaci si jak výtvarník, tak i autor návrhu výukové aplikace (pedagog) může ověřit, jak se animovaná komponenta bude chovat.

Výtvarník je tak odstíněn od programátorských detailů aplikace. Obdobně, tvůrce návrhu výukové aplikace se nemusí věnovat detailům implementace grafického návrhu a může v komunikaci s výtvarníkem snadněji dosáhnout realizaci svých představ (viz *Obrázek 4*).

Úlohou programátora, implementujícího vlastní simulátor, je propojit vygenerovaný grafický objekt se simulačním modelem na pozadí. Takto vytvořené animační „loutky" je možné, přes hodnoty ovládající jejich tvar, přímo napojit na výstupy modelů a není potřeba zvlášť přidávat další programovou mezivrstvu pro propagaci dat, jako tomu bylo při použití Flashových animací.

Použití grafických možností platformy Silverlight tedy více než nahrazuje původní přístup s použitím animací založených na platformě Adobe Flash. Při tvorbě animací jako vizuálního rozhraní pro simulátory proto nepotřebujeme platformu Adobe Flash, kterou je možno plně nahradit novými nástroji pro tvorbu animací firmy Microsoft.

6. Mozek výukové aplikace – simulační model

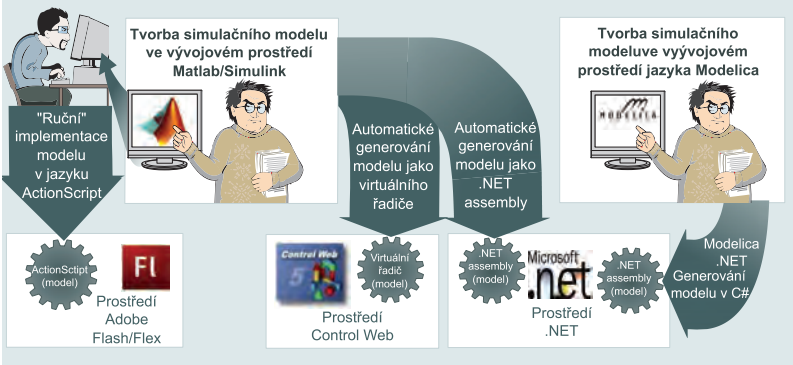
Implementace simulačních modelů do výukového programu není triviální problém (viz *Obrázek 5)*.

Pro tvorbu simulačních modelů využíváme speciální vývojové nástroje, určené pro tvorbu, ladění a verifikaci simulačních modelů (Matlab/Simulink či akauzální vývojové prostředí využívající jazyk Modelica), o kterých jsme pojednávali v předchozích kapitolách.

Odladěné a verifikované modely je ale potřeba převést z vývojového prostředí, v němž jsme vytvořili, odladili a verifikovali, do prostředí, kde je vytvářen vlastní výukový simulátor.

U jednoduchých modelů se to dá udělat „ručně" – tak to často děláme u čistě flashových výukových simulátorů, kde vývojovým prostředím pro tvorbu simulátorů je pouze Adobe Flash.

Pro složitější modely jsme si však raději vytvořili softwarové nástroje, které nám tuto práci zautomatizují.

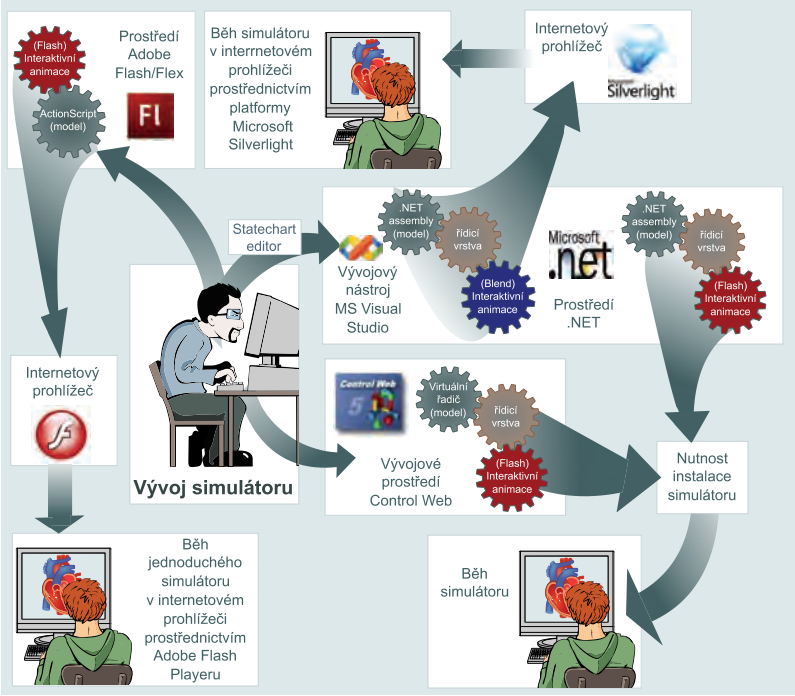


Obrázek 5 - Pro vývoj simulačních modelů využíváme simulační prostředí Matlab/Simulink a v poslední době vývojové prostředí jazyka Modelica (např. Dymola nebo MathModelica). Identifikovaný simulační model můžeme ručně „přepsat" do prostředí, v němž bude vytvářen vlastní simulátor. To se vyplatí jen u jednoduchých simulátorů vytvářených v prostředí AdobeFlash (resp. Adobe Flex). U složitějších simulátorů vytvářených v prostředí .NET (nebo dříve i ControlWeb) jsme vytvořili speciální nástroje automaticky generující simulační jádro modelu ze simulinkového modelu. Pro vývojové prostředí jazyka Modelica jsme vytvořili nástroj, generující model (a příslušný solver algebrodiferenciálních rovnic) v jazyku C# což umožní tvorbu webových simulátorů přímo spustitelných v internetovém prohlížeči (pomocí platformy Silverlight).

V simulátoru Golem, implementovaném v prostředí ControlWeb byl model reprezentován jako ovladač virtuální měřící/řídící karty. Pro automatizaci převodu simulačního modelu z prostředí Matlab/Simulink jsme vytvořili generátor, který vytváří zdrojový text tohoto ovladače v jazyce C přímo ze Simulinkového modelu [5].

Obdobně, abychom si ulehčili vytváření simulátorů, vytvářených ve Visual Studiu .NET (tj. aby nebylo nutné ve Visual Studiu .NET „ručně" programovat již odladěný simulační model) vyvinuli jsme i zde speciální softwarový nástroj [6, 14], který automaticky ze Simulinku vygeneruje simulační model ve formě komponenty pro prostředí .NET.

Výstupem programu v Modelice je vygenerovaný program simulátoru v C++. Pokud bychom vystačili se simulátorem, který se bude následně na počítači klienta vždy instalovat, pak nám program modelu v C++ stačí. Pokud ale chceme využít nové možnosti prostředí .NET umožňující vytvářet pomocí platformy Silverlight webově spustitelné aplikace běžící v internetovém prohlížeči, pak musíme vytvořit nástroj, který umožní z Modeliky generovat zdrojový text modelu a příslušného solveru v C# [3] což je i náš úkol v rámci mezinárodního konsorcia Open Source Modelica Consortium – (viz *http://www.ida.liu.se/labs/pelab/modelica/OpenSourceModelicaConsortium.html*).



Obrázek 6 - Programátor je zodpovědný za vývoj vlastního výukového simulátoru. Na základě znalostí struktury identifikovaného simulačního modelu může v prostředí Adobe Flash (nebo pomocí nástroje Adobe Flex) naprogramovat simulační jádro a příslušné uživatelské rozhraní výukového simulátoru, využívajícího vytvořené Flashové animace. Celá výuková aplikace pak může být distribuována prostřednictvím internetu a pomocí Flash Playeru provozována v internetovém prohlížeči. Složitější simulátory jsou však vytvářeny v prostředí .NET (dříve jsme simulátory také vytvářeli ve vývojovém prostředí Control Web). Tvorba simulátorů předpokládá, že programátor propojí animace se simulační komponentou modelu (realizovanou jako .NET assembly). Pro usnadnění programování tohoto propojení (a vytvoření propojovací řídicí vrstvy) jsme vyvinuly nástroj Statechart Editor, založený na využití hierarchických stavových automatů. Simulátory využívající flashové animace (propojené se svým okolím přes rozhraní Active X) vyžadují instalaci vytvořené aplikace v počítači. Vytvoříme-li animace v prostředí Microsoft Expression Blend pak můžeme vybudovat aplikaci spustitelnou v internetovém prohlížeči s nainstalovaným doplňkem Silverlight. Tímto způsobem můžeme vytvářet poměrně numericky náročné výukové simulátory distribuované jako webové aplikace spouštěné v internetovém prohlížeči klienta.

7. Tělo výukového simulátoru – instalovatelný program nebo webová aplikace

Tvorba výukové simulační aplikace je poměrně náročná programátorská práce, spočívající ve vytvoření simulačního jádra vyvíjené aplikace (pokud toto jádro již nebylo z příslušného simulačního vývojového nástroje automaticky vygenerováno) a jejího propojení s grafickými prvky vizuálního uživatelského rozhraní (viz Obrázek 6).

Pro výukové aplikace s jednoduchými modely vystačíme s prostředím Flashového přehrávače a jazykem Action Script, v němž naprogramujeme vlastní simulátor a celou aplikaci můžeme spouštět přímo v internetovém prohlížeči. Pro složitější aplikace to ale nestačí.

Simulátory jsme v minulosti vytvářeli ve vývojovém prostředí Control Web české firmy Moravské přístroje. Vytvořená aplikace vyžadovala instalaci do počítače uživatele, nebo (u webově šiřitelných aplikací) alespoň instalaci prostředí runtime Control Web.

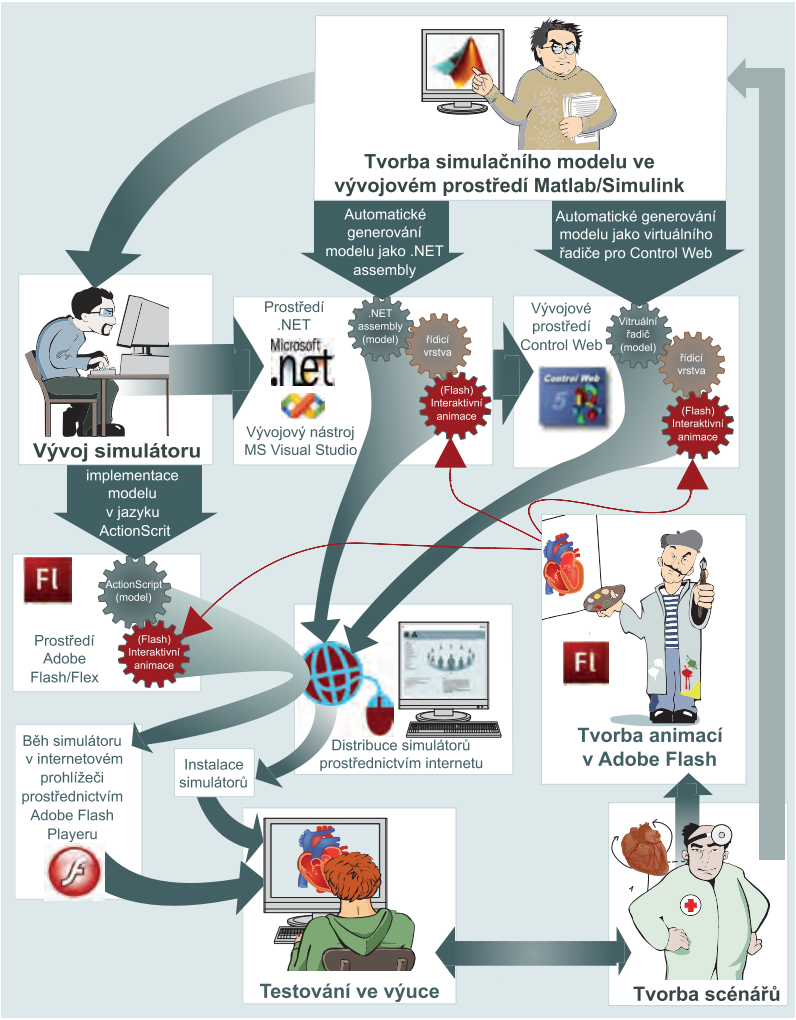
Platformou pro vývoj simulátorů, kterou využíváme nyní, je platforma Microsoft .NET a programovací prostředí Microsoft Visual Studio .NET, které, poskytuje velké možnosti pro programátorskou práci. Můžeme využít grafické komponenty uživatelského rozhraní, vytvořené v Adobe Flash, které můžeme propojit (přes ActiveX) s jádrem simulátoru, kterým je simulační model a grafické komponenty se pak mohou chovat jako loutky řízené simulačním modelem. Tímto způsobem byla například realizována kapitola našeho Atlasu fyziologie a patofyziologie, věnovaná základních dynamických vlastnostech fyziologických regulačních systémů - *http://physiome.cz/atlas/sim/RegulaceSys/* nebo výukový simulátor přenosu krevních plynů - *http://physiome.cz/atlas/sim/BloodyMary_cs/*.

Nevýhodou tohoto přístupu je nutnost instalace programu (nabízeného přes internetové rozhraní) do počítače klienta. Vyžaduje to ale, aby klient měl příslušná instalační práva na počítači, na němž pracuje. To ale obvykle neplatí zejména v počítačových učebnách, kde bývají počítače chráněny před instalací nevhodného softwaru, a uživatel musí o instalaci výukového programu nejprve požádat příslušného správce.

Proto je vhodné mít možnost spouštět a ovládat i složité modely přímo z webového prohlížeče. Tato cesta je možná, pokud celý simulátor vytvoříme tak, aby byl spustitelný v prostředí – Silverlight, tj. simulační jádro je vytvořeno formě řízeného kódu pro prostředí .NET (ve formě .NET assembly) a grafické komponenty jsou vytvořeny v prostředí Microsoft Expression Blend.

8. Struktura simulátoru – MVC architektura

V případě složitější architektury může být logika propojení vizuálního uživatelského rozhraní a simulačního modelu poměrně složitá, proto je vhodnější mezi vrstvu vizuálních elementů a vrstvu simulačního modelu vložit řídící vrstvu, která na jednom místě řeší veškerou logiku komunikace uživatelského rozhraní s modelem a kde je ukládán i příslušný kontext.



Obrázek 7 - Původní řešení kreativního propojení nástrojů a aplikací pro tvorbu simulátorů a výukových programů využívajících simulační hry. Základem e-learningového programu je kvalitní scénář, vytvořený zkušeným pedagogem. Tvorba animovaných obrázků je odpovědnost výtvarníků, kteří vytvářejí interaktivní animace v prostředí Adobe Flash. Jádrem simulátorů je simulační model, vytvářený v prostředí speciálních vývojových nástrojů, určených pro tvorbu simulačních modelů. Dlouho jsme zde využívali prostředí Matlab/Simulink od firmy Matworks. Vývoj simulátoru je náročná programátorská práce, pro jejíž usnadnění jsme vyvinuli speciální programy, usnadňující automatický převod vytvořeného simulačního modelu z prostředí Matlab/Simulink do prostředí Control Web a do prostředí Microsoft .NET.

V literatuře [1, 11] se hovoří o tzv. MVC architektuře výstavby simulátorů (Model – View – Controller).

Toto uspořádání je nezbytné zejména při složitějších modelech a simulátorech, jejichž uživatelské zobrazení je reprezentováno mnoha virtuálními přístroji na více propojených obrazovkách. Výhody tohoto uspořádání zvláště vyniknou při modifikacích jak modelu, tak i uživatelského rozhraní.

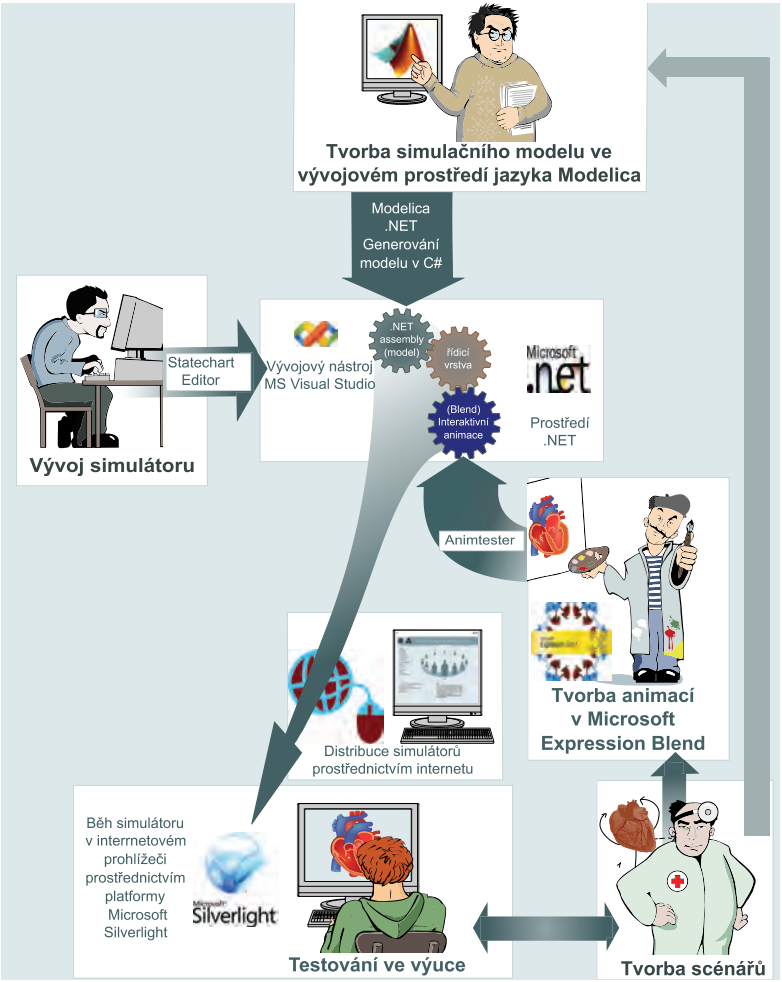
Při návrhu řídící vrstvy, propojující vrstvu simulačního modelu s uživatelským rozhraním, se nám velmi osvědčilo využít propojené stavové automaty (jejichž pomocí je možno zapamatovat příslušný kontext modelu a kontext uživatelského rozhraní).

Vytvořili jsme proto speciální softwarový nástroj, Statechart Editor pomocí kterého můžeme propojené stavové automaty vizuálně navrhovat, interaktivně testovat jejich chování a automaticky generovat zdrojový kód programu pro prostředí Microsoft .NET [9]. Tento nástroj umožňuje zefektivnit programování propojek simulačního modelu s vizuálními objekty uživatelského rozhraní ve výukovém simulátoru.

9. Propojení platforem pro tvorbu modelů, simulátorů i animací

Při tvorbě simulátorů jsme nuceni pracovat se třemi typy rozdílných softwarových nástrojů:

1. Softwarové nástroje pro tvorbu a odlaďování matematických modelů, které jsou podkladem simulátoru – Matlab/Simulink a v poslední době s akauzálními nástroji využívajícími jazyk Modelica. V tomto prostředí je výhodné a efektivní simulační modely vyvíjet, problematické je ale v tomto prostředí simulátory provozovat.
2. Softwarové nástroje pro vývoj vlastního simulátoru – zde především využíváme prostředí Microsoft Visual Studio .NET. Dříve jsme také využívali vývojové prostředí Control Web, české firmy Moravské přístroje, zejména proto, že má vynikající možnosti pro rychlé vytváření uživatelského rozhraní simulátoru – toto rozhraní má ale příliš „technicistní" charakter. Jednoduché modely jsme implementovali v jazyce ActionScript a vystačíme proto pouze s prostředím Adobe Flash, doplněném případně o prostředí Adobe Flex.
3. Softwarové nástroje pro tvorbu interaktivní multimediální grafiky – uživatelského rozhraní pro simulátory. Zde jsme dlouhodobě využívali nástroj Adobe Flash (dříve Macromedia Flash). V tomto nástroji je možné vytvářet interaktivní animace, které ale lze zároveň programovat pomocí speciálního programového jazyka ActionScript. Důležité je, že animace mohou (díky výše zmíněné možnosti programování v jazyce ActionScript) softwarově komunikovat se svým okolím prostřednictvím komponenty Active X. Nyní při vývoji grafických aplikací dáváme před nástrojem firmy Adobe přednost vývojovému prostředí Microsoft Expression Blend jehož pomocí můžeme vytvářet grafické komponenty pro platformu Siverlight.



Obrázek 8 - Nové řešení kreativního propojení nástrojů a aplikací pro tvorbu simulátorů a výukových programů využívajících simulační hry. Základem e-learningového programu nadále zůstává kvalitní scénář, vytvořený zkušeným pedagogem. Tvorba animovaných obrázků je odpovědnost výtvarníků, kteří vytvářejí interaktivní animacev prostředí Expression Blend. Pro vytváření a testování animací, které budou posléze řízeny simulačním modelem výtvarník využívá námi vyvinutý softwarový nástroj Animtester. Jádrem simulátorů je simulační model, vytvářený ve vývojovém prostředí simulačního jazyka Modelica. V rámci Open Modelica Source Consortia jsme vytvořili nástroj, který z Modeliky generuje zdrojový text modelu a příslušného solveru algebrodiferenciálních rovnic v C#. To umožní z modelu vygenerovat .NET komponentu simulačního jádra pro výslednou aplikaci v platformě Silverlight, umožňující distribuovat simulátor jako webovou aplikaci běžící přímo internetovém prohlížeči (i na počítačích s různými operačními systémy).

Protože pro tvorbu simulačních modelů a pro vytváření vlastního simulátoru používáme odlišné vývojové nástroje, museli jsme zajistit dostatečně flexibilní přenos výsledků z jednoho vývojového prostředí do druhého – tj. např. zautomatizovat převod modelu z prostředí Matlab/Simulink do prostředí Visual studia Microsoft .NET (a v minulosti i do prostředí Control Web).

Tyto „propojovací" nástroje umožnily vyvíjet a průběžně aktualizovat matematický model v nejvhodnějším prostředí určeném pro vývoj matematických modelů a zároveň vyvíjet vlastní simulátor ve Visual Studiu .NET (případně ControlWeb), aniž bylo nutné matematický model „ručně" přeprogramovávat.

Umožnily snadnou multidisciplinární spolupráci členů řešitelského týmu – systémových analytiků, vytvářejících matematické modely a programátorů, implementujících simulátor (viz *Obrázek 7*).

Na druhé straně to ovšem znamenalo práci ve třech softwarových prostředích a při každé inovaci jednotlivého prostředí bylo nezřídka nutno inovovat příslušné propojovací nástroje.

Simulátory je z hlediska uživatele nejvýhodnější distribuovat přes webové internetové rozhraní, které může sloužit i pro příslušný interaktivní výklad. Webovou výkladovou aplikaci lze snadno propojit s jednoduchými modely implementovanými přímo v Action Scriptu na pozadí flashových animací (tímto způsobem jsme např. vytvořili výukovou aplikaci „mechanické vlastnosti kosterního svalu" - *http://www.physiome.cz/atlas/sval/svalCZ/svalCZ.html*).

Složitější modely, např. komplexní model přenosu krevních plynů (*http://physiome.cz/atlas/sim/ BloodyMary_cs/*) již před vlastním spuštěním ale vyžadovaly instalaci modelu na počítači klienta (a také i přítomnost platformy .NET, která, pokud nebyla na počítači klienta nainstalována, byla automaticky stažena ze serveru Microsoftu).

Instalace programů ale vyžaduje mít k počítači příslušná administrátorská práva. Kromě toho se model, který běží jako samostatná aplikace, jen nepřímo se propojuje s webovým rozhraním, kde je realizován interaktivní multimediální výklad.

Tento problém řeší naše nová technologie tvorby simulátorů, využívající platformu Silverlight (viz *Obrázek 8*).

Grafické prvky jsou vytvářeny v prostředí Microsoft Expression Blend.

Simulační jádro ale vyžaduje být realizováno jako řízený kód v prostředí .NET – to zajišťuje námi vyvinutá aplikace Modelica .NET generující zdrojový kód modelu a příslušného solveru algebrodiferenciálních rovnic v C#.

Pro návrh vnitřní logiky aplikace používáme hierarchické stavové automaty (jejichž pomocí je možno zapamatovat příslušný kontext modelu a kontext uživatelského rozhraní). Námi vyvinuté vizuální prostředí (Statecharts editor) umožňuje graficky automaty navrhnout, vygenerovat jejich kód a také je ladit.

Výhodou je že jak grafické interaktivní prvky tak i simulační jádro jsou vytvářeny na jedné platformě - odpadá tedy nutnost složitého přemosťování prostředí .NET a Adobe Flash přes ActiveX komponenty.

Simulátor být snadno kombinován s výkladovou kapitolou. Výsledná aplikace (jak simulátor, tak i výkladová kapitola) může být realizována jako webová aplikace spustitelná přímo v internetovém prohlížeči, bez nutnosti její instalace na počítači klienta. Může běžet v různých operačních systémech - jedinou podmínkou je instalovaný plugin Silverlight v internetovém prohlížeči.

10. Zabalení simulačních her do multimediálního výkladu

Simulátor bez výkladové části vyžaduje zkušeného pedagoga při jeho využívání. Simulátory je proto vhodné kombinovat s výkladovými lekcemi. Velký pedagogický efekt mají interaktivní výukové programy, dosažitelné po internetu, které kombinují výukový text, doprovázený kreslenými animovanými obrázky se simulační hrou, ozřejmující vykládanou látku pomocí modelu ve virtuální realitě.

V naší technologii jsou simulační hry součástí e-learningových multimediálních výukových lekcí, jejichž podkladem je scénář vytvořený zkušeným pedagogem. Pedagog navrhuje vysvětlující text a s textem propojené doprovodné obrázky a animace.

Animace jsou vytvářeny výtvarníkem v úzké spolupráci s pedagogem v prostředí Adobe Flash pro Flash Player v internetovém prohlížeči nebo (v novější technologii) v prostředí Microsoft Expression Blend pro platformu Siverlight.

Text je poté namluven a synchronizován se spouštěním jednotlivých animací a s odkazy na simulační hry. Jednotlivé komponenty jsou kompletovány do výukových lekcí.

Vytvořit synchronizaci animací se zvukem v prostředí Adobe Flash není ovšem zcela jednoduchý problém.

Protože se animace v Adobe Flash se vytvářejí jako v kresleném filmu – po jednotlivých políčkách, pak časová posloupnost vizualizace animací závisí na tom, kdy se „přehrávací hlava" (při nastavené rychlosti přehrávání) dostane k příslušnému klíčovému snímku.

Do vrstvy filmových políček je možné uložit i zvukovou stopu. Pro synchronizaci přehrávání animací se zvukem je nutné zajistit (pomocí příkazu v jazyce ActionScript), aby se vložený filmový klip přehrávající danou animaci spustil ve správném v okamžiku, kdy se přehrávací hlava dostane do příslušné pozice.

Pro usnadnění této synchronizace jsme vytvořili speciální nástroj PlayDirector jako knihovní prvek vkládaný do vytvářeného flashového filmového klipu. Při jeho přehrávání ve FlashPlayeru se dají podle zvukové stopy interaktivně nastavit příslušné zarážky, podle kterých se pak vygenerují data pro vložený skript, který zajistí spouštění příslušných animací v požadovaném čase.

Pro kompletaci multimediálních výukových lekcí vytvořených v prostředí Adobe Flash jsme pak využívali vývojový nástroj Adobe Presenter, dodávaný jako softwarového prostředí serveru Adobe Connect (nyní možné Adobe Presenter zakoupit i separátně – viz *http://www.adobe.com/products/presenter/*).

Synchronizace animací se zvukem je ovšem v prostředí Silverlight mnohem jednodušší. Protože animační přístup v Silverlightu je založen na animaci pomocí změny některých vlastností grafických objektů v čase (oproti systému založeném na snímcích v Adobe Flash) je synchronizace animací se zvukovou stopou přímočará, pro tuto synchronizaci nám vývojové prostředí Microsoft Expression Blend zcela postačí a není zapotřebí vytvářet nový pomocný nástroj.

Závěr

Naše současná technologie, založená na tvorbě modelů v akauzálním prostředí jazyka Modelica, zefektivňuje tvorbu simulačních modelů, které jsou teoretickým jádrem výukových simulátorů [9]. Propojení Modeliky s prostředím platformy Silverlight a s tvorbou grafických komponent v nástroji Microsoft Expression Blend umožňuje vytvářet simulační jádro i grafické interaktivní prvky na jedné vývojové platformě, což zjednodušuje vývoj numericky náročných simulačních aplikací přímo spustitelných v internetovém prohlížeči.

Poděkování

Tvorba výukových simulátorů a vývoj příslušných vývojových nástrojů byly podporovány granty MŠMT č. 2C06031 „e-Golem", výzkumným záměrem MSM 0021620806 a společností Creative Connections s. r. o.

Literatura

[1.] Abram, S. R., Hodnett, B. L., Summers, R. L., Coleman, T. G., & Hester, R. L. (2007). Quantitative circulatory physiology. An integrative mathematical model of human mathematical model of human physiology for medical education. Advanced Physiology Education , 31, stránky 202-210.

[2.] Collins, D. (1995). Designing object-oriented user interfaces. Redwood City, CA: Benjamin Cummings (ISBN: 0-8053-5350-X).

[3.] Fritzson, P., Privitzer, P., Sjölund, M., Pop, A. (2009). Towards a text generation template language for modelica, Proceedings 7th Modelica Conference, Como, Italy, Sep. 20-22, 2009, (editor: Francesco Casella), Linköping University Electronic Press, 2009. ISBN 978-91-7393-513-5, stránky 193-207, 2009. Internet Proceedings: http://www.ep.liu.se/ecp/043/021/

[4.] Kofránek, J., Anh Vu, L. D., Snášelová, H., Kerekeš, R., & Velan, T. (2001). GOLEM – Multimedia simulator for medical education. In L. Patel, R. Rogers, & R. Haux (Editor), MEDINFO 2001, Proceedings of the 10th World Congress on Medical Informatics. 1042-1046. London: IOS Press. Práce je dostupná na adrese http://www.physiome.cz/references/MEDINFO2001.pdf

[5.] Kofránek, J., Kripner, T., Andrlík, M., & Mašek, J. (2003). Creative connection between multimedia, simulation and software development tools in the design and development of biomedical educational simulators. Simulation Interoperability Workshop, Position papers, Volume II, paper 03F-SIW-102., stránky 677-687.

[6.] Kofránek, J., Andrlík, M., Kripner, T., & Stodulka, P. (2005). From Art to Industry: Development of Biomedical Simulators. The IPSI BgD Transactions on Advanced Research , 1 #2(Special Issue on the Research with Elements of Multidisciplinary, Interdisciplinary, and

Transdisciplinary: The Best Paper Selection for 2005), stránky 62-67. Práce je dostupná na adrese http://www.physiome.cz/references/IPSI2005.pdf

[7.] Kofránek, J., Matoušek, S., Andrlík, M., Stodulka, P., Wünsch, Z., Privitzer, P., a další. (2007). Atlas of physiology - internet simulation playground. In B. Zupanic, R. Karba, & s. Blažič (Editor), Proceedings of the 6th EUROSIM Congress on Modeling and Simulation, Vol. 2. Full Papers (CD ROM) (stránky MO-2-P7-5: 1-9). Ljubljana: University of Ljubljana. Práce je dostupná na adrese http://ww.physiome.cz/references/EUROSIM2007.pdf

[8.] Kofránek, J., Mateják, M., & Privitzer, P. (2009). School as a (multimedia simulation) game. The use of object tools for designing multimedia applications for biomedical teaching. In C. Moler, A. Procházka, R. Bartko, M. Folin, J. Houška, & P. Byron (Editor), Technical Compuning Prague 2009, 17th Annual Conference Proceedings. CD ROM, ISBN 978-80-7080-733-0, internetový sborník: http://dsp.vscht.cz/konference_matlab/MATLAB09/ (stránky 55: 1-27). Praha: Humusoft s.r.o. Práce je dostupná na adrese http://www.physiome.cz/references/TCP09.pdf

[9.] Kofránek, J., Mateják, M., & Privitzer, P. (2009). Leaving toil to machines - building simulation kernel of educational software in modern software environments. CD ROM. In L. Dušek, D. Schwarz, & S. Štípek (Editor), Mefanet 2009, Conference Proceedings (stránky kofranek.pdf: 1-39). Brno: Masarykova Univerzita. Práce je dostupná na adrese http://www.physiome.cz/references/MEFANET2009.pdf

[10.] Kofránek, J., Privitzer, P., Matoušek, S., Vacek, O., & Tribula, M. (2009). Schola Ludus in modern garment: use of web multimedia simulation in biomedical teaching. Proceedings of the 7th IFAC Symposium on Modelling and Control in Biomedical Systems, Aalborg, Denmark, August 12-14, 2009, stránky 425-430. Práce je dostupná na adrese http://www.physiome.cz/references/IFAC2009.pdf

[11.] Leff, A., & Rayfield, J. T. (2007). Web-application development using the Model/View/Controller design patterns. Fifth IEEE International Enterprise Distributed Object Computing Conference, ISBN: 0-7695-1345-X (str. 118). Seattle: IEE International.

[12.] Little, J. A., Beres, J., Hinkson, G., Rader, D., & Croney, J. (2009). Silverlight 3 programer's reference (Wronx programer to programer). Indianopolis: Wronx Wiley

[13.] Sells, C., & Griffins, I. (2007). Programming WPF. Beijing, Cabridge,Farhan, Köln, Sebastopol, Taipe, Tokyo: O'Reilly

[14.] Stodulka, P., Privitzer, P., Kofránek, J., & Vacek, O. (2007). Development of WEB accessible medical educational simulators. In B. Zupanic, R. Karba, & S. Blažič (Editor), Proceedings of the 6th EUROSIM Congress on Modeling and Simulation, Vol. 2. Full Papers (CD). (stránky MO-3-P4-2, 1-6). Ljubljana: University of Ljubljana. Práce je dostupná na adrese http://www.physiome.cz/references/EUROSIM2007Stod.pdf

[15.] Williams, B. (2008). Microsoft Expression Blend unleashed, First edition. Indeanopolis: Sams

Kontakt:

MUDr., Mgr. Pavol Privitzer
Jan Šilar
Ing. Martin Tribula
MUDr. Jiří Kofránek, CSc.

Oddělení biokybernetiky a počítačové podpory výuky,

Ústav patologické fyziologie 1.LF UK
U nemocnice 5, 121 53 Praha 2
tel: +420 22496 5912
e-mail: *ppriv@lf1.cuni.cz*
http://physiome.cz