JEN SIMULINK NESTAČÍ: VYUŽITÍ SIMULINKU PŘI TVORBĚ VÝUKOVÝCH MULTIMEDIÁLNÍCH SIMULÁTORŮ

Jiří Kofránek, Michal Andrlík, Pavol Privitzer, Petr Stodulka, Jan Mašek

Laboratoř biokybernetiky, Ústav patologické fyziologie 1. Lékařské fakulty UK, Praha

1. Úvod

Navzdory tomu, že se využití počítačů ve výuce stalo tématem řady konferencí, odborných i popularizačních článků, přesto, že hardwarové možnosti i softwarové nástroje dnes umožňují vytvářet náročná interaktivní multimedia, k výraznému rozšíření multimediálních výukových programů ve výuce zatím nedošlo. Příčin je několik.

Ukazuje se, že tvorba výukových programů je podstatně ***náročnější*** na čas, lidské i materiální zdroje, než je obvykle plánováno (jestliže například v rámci grantů Fondu rozvoje vysokých škol je na vytvoření jednoho multimediálního výukového programu plánováno maximálně 120 tisíc Kč, pak výsledkem nemůže být nic jiného, než po technické a výtvarné stránce nedotažený produkt).

Tvorba kvalitních výukových programů vyžaduje týmovou ***multidisciplinární*** spolupráci zkušených pedagogů, výtvarníků i programátorů.

Nároky stoupají, pokud na pozadí výukového programu má běžet ***simulační program***, umožňující interaktivní simulační hry - ve vývojovém týmu pak musí být i odborníci, kteří jsou schopni navrhnout, formalizovat a odladit příslušné modely.

Konečně, pro kreativní propojení různých profesí, podílejících se na tvorbě výukové multimediální aplikace, musí být k dispozici ***vhodně zvolené vývojové nástroje*** (jejichž cena není malá a jejichž ovládnutí vyžaduje určité úsilí a čas).

Nástroje vhodné pro vytváření simulačních modelů (Simulink apod.) jsou výborné pro vytváření a identifikaci modelu. Vlastní tvorba výukového simulátoru však vyžaduje využití ještě jiných vývojových nástrojů než je prostředí Matlab/Simulink (např. objektové programové prostředí Microsoft Visual Studio .NET, nebo prostředí Control Web původně určené pro vizualizaci průmyslových řídících a měřících systémů či prostředí pro tvorbu programovatelných interaktivních animací – Macromedia Flash). Podmínkou pro efektivní využití různých vývojových nástrojů je ale jejich ***vhodné propojení.***

2. Klíč k úspěchu a kostra výukové aplikace – kvalitní scénář

Klíčem k úspěchu jakéhokoli výukového programu je dobrý scénář. 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. Základem scénáře 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 případně vloží simulační model a jak bude ovládán, kam se vloží test znalostí, jak bude vypadat, jak se bude vyhodnocovat a jak se bude reagovat na jeho výsledek apod.

Zde se nám osvědčilo využívat postupu, který je znám u kresleného filmu – nakreslit (nejlépe ve spolupráci s výtvarníkem) ***obrazový scénář***, tzv. ***"Story Board"*** – hrubou posloupnost

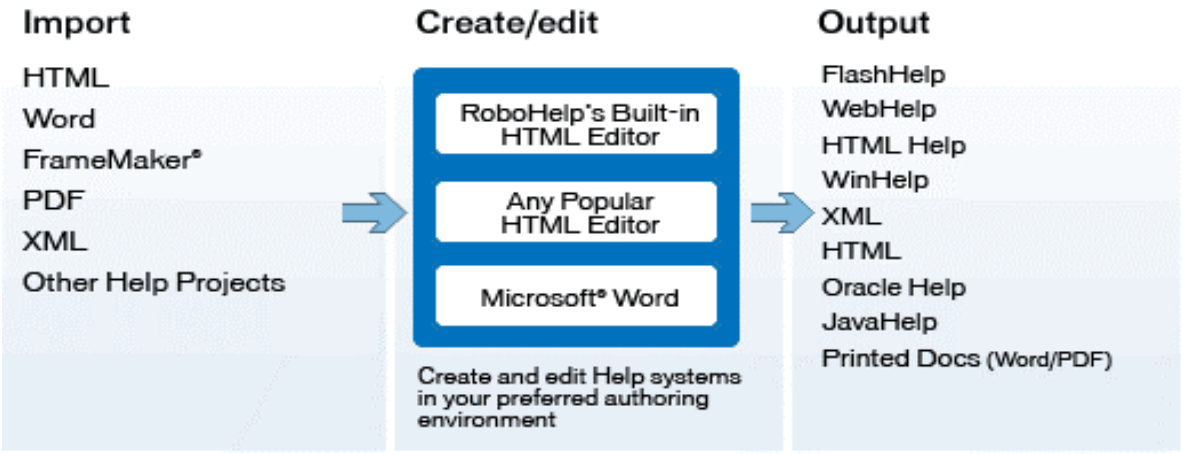
jednotlivých obrazovek, a ke každé obrazovce pak napsat komentář (případně odkaz na příslušnou část textu vytvářeného v klasickém textovém editoru).

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.

Prvním problémem, který je nutno vyřešit, je 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 flowchartů či struktogramů popsat větvení, rozhodovací bloky apod. s odkazy na příslušné stránky textu a obrázky uložené v dalších souborech.

Při psaní scénáře se osvědčilo využít schopností moderních textových editorů vytvářet hypertextové odkazy – tím již vlastní scénář dostává jisté rysy budoucí interaktivity. Pro detailnější hypertextové propojení textové části scénáře je vhodné využít i vývojový nástroj původně určený na tvorbu nápověd ("helpů") – podle našich zkušeností nejvhodnějším vývojovým prostředím, který je v současné době na trhu, je program ***RoboHelp Office*** (nyní ve verzi *X5*), vyvinutý firmou E-help, kterou nedávno koupila společnost ***Macromedia***.

RoboHelp je velmi výkonný nástroj, v němž můžeme vytvořit textovou část scénáře včetně hypertextových propojení a větvení. Dobře spolupracuje s Microsoft Office, takže do prostředí RoboHelpu je snadné inkorporovat texty napsané ve Wordu. V prostředí RoboHelp je snadné vytvořit klasickou strukturu "elektronické knihy" včetně rozbalovacího stromu kapitol a rejstříku. V poslední verzi RoboHelpu již byly odstraněny problémy s diakritickými znaménky českého jazyka, které v předchozích verzích při určité neopatrnosti byly příčinou mnoha svízelných okamžiků.



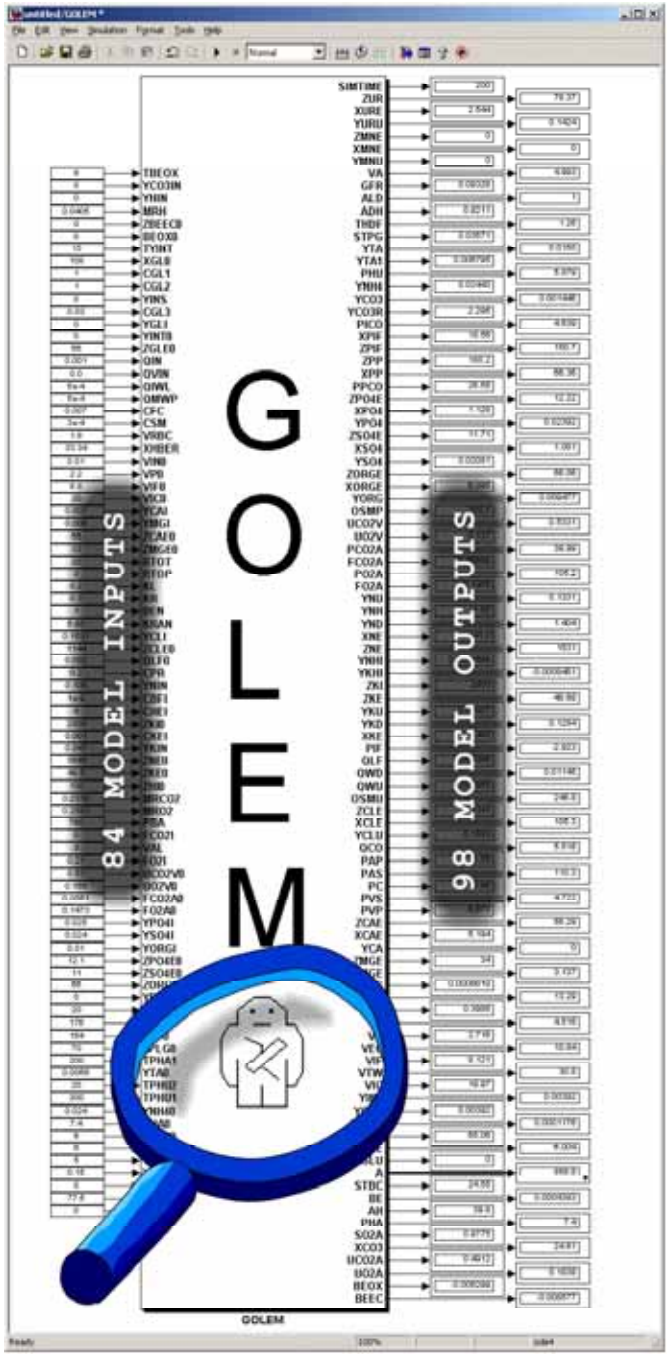
Obr. 1 *Vstupy a výstupy programu RoboHelp*

RoboHelp můžeme využít nejen pro tvorbu scénáře, ale i jako ***kostru pro vytváření výkladové části vlastní výukové aplikace*** – vstupem je textový soubor s vloženými obrázky ve Wordu či ve formátu PDF, vstupem může však být i HTML nebo XML soubor, popřípadě i jiný soubor formátu RoboHelp. Samotný RoboHelp je možno snadno propojit s Microsoft Wordem pro tvorbu a editaci textů a zároveň i s jakýmkoliv HTML editorem. RoboHelp dobře spolupracuje se všemi produkty firmy Macromedia – můžeme do něj například snadno vkládat interaktivní animace vytvořené v prostředí Macromedia Flash, nebo naopak celý hypertextovými odkazy provázaný výstup z RoboHelpu uložit jako komponentu do Flashe. RoboHelp může také vygenerovat klasický HTML či XML soubor, který můžeme dále využít

(obr.1). Vynikající vlastností poslední verze RoboHelpu je také jeho schopnost spolupracovat s prostředím Microsoft .NET, v němž je např. možné implementovat simulační program, který běží na pozadí výukové aplikace.

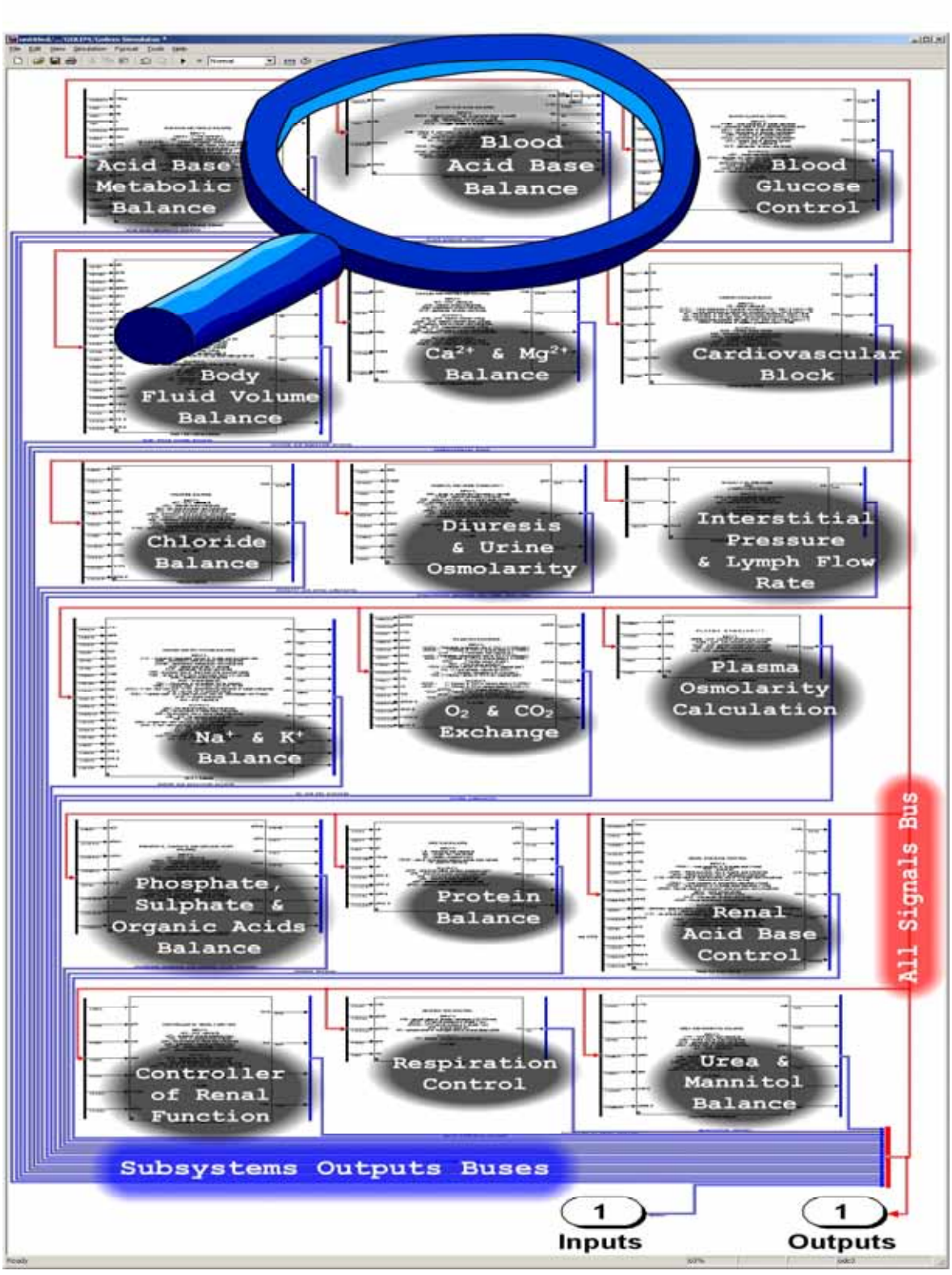
2. Mozek výukové aplikace – simulační modely

Moderní výukové programy by neměly být jen multimediální náhradou klasických učebnic, jsou ale zcela novou výukovou pomůckou. Obsahují totiž ***simulační komponenty***, které umožňují pomocí simulačních her si názorně "osahat" vykládaný problém ve virtuální realitě a přinášejí tak zcela nové možnosti pro vysvětlování složitých problémů. ***Simulační hrou*** je možné bez rizika zkoumat chování simulovaného objektu – přistávat virtuálním letadlem, léčit virtuálního pacienta apod. Ale nejenom to. Modelovaný objekt můžeme rozdělit na subsystémy a testovat jejich chování odděleně i jako součást vyššího celku. Tak např. při studiu složitých fyziologických regulací můžeme dočasně odpojit vybrané regulační smyčky a umožnit studentům sledovat reakce těchto subsystémů na změny vstupních veličin (které ale v reálném organismu nejsou nezávislé a jsou samy regulovány).



Obr. 2 *Ukázka simulačního čipu (v daném případě reprezentujícího simulační model, který je základem simulátor GOLEM). V prostředí Simulinku je možno snadno otestovat jeho chování – k jednotlivým "vstupním pinům" lze přivést vstupní hodnoty (nebo průběhy hodnot) a od "výstupních pinů" na virtuálních displejích či osciloskopech odečítat výstupy, resp. časové průběhy výstupů. Na dalším obrázku se podíváme dovnitř tohoto čipu*

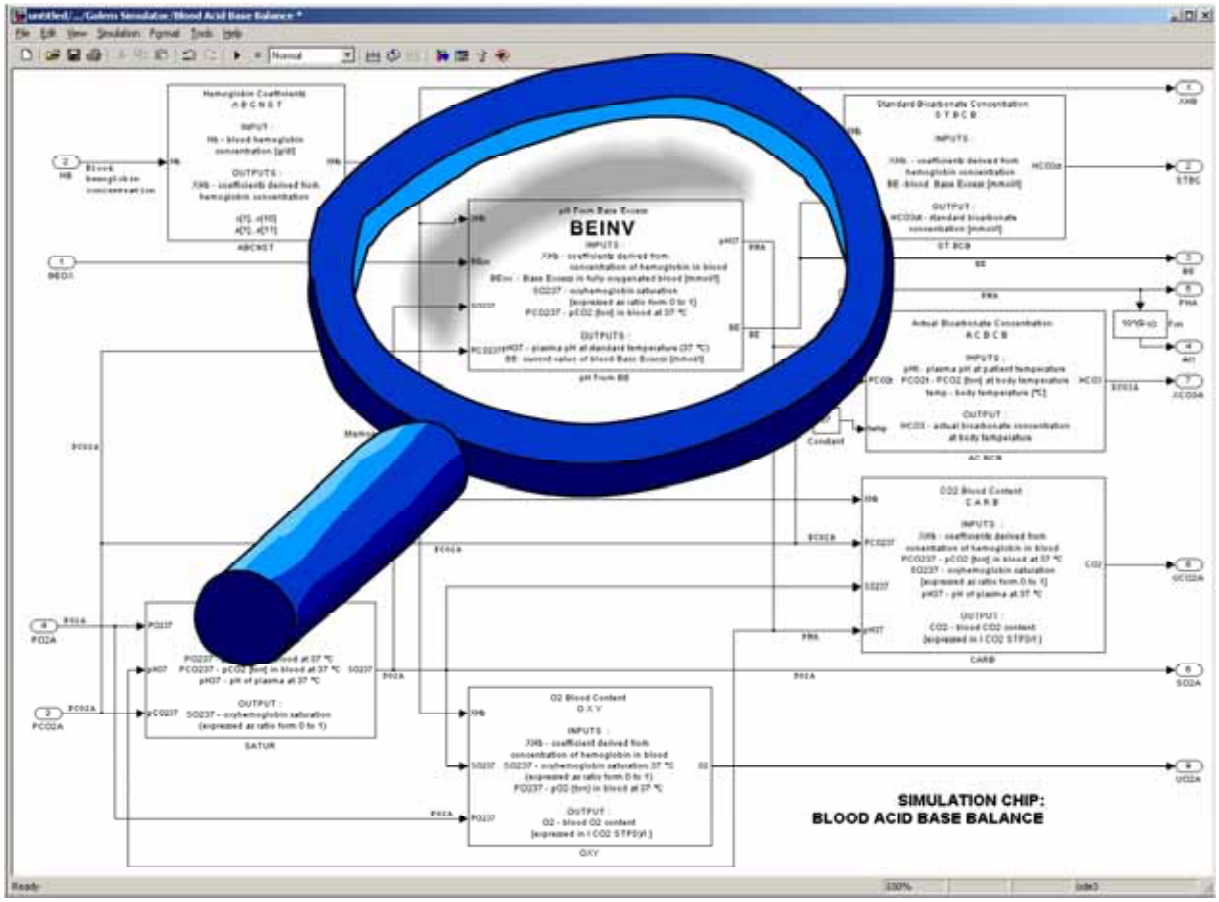
Tím dovolíme sledovat dynamiku chování jednotlivých subsystémů při postupných změnách pouze jediného vstupu, zatímco jiné vstupy jsou nastaveny na zvolenou konstantní hodnotu (tzv. princip "ceteris paribus"). Postupně pak můžeme jednotlivé dočasně rozpojené regulační vazby opět zapojovat a studovat jejich vliv na chování organismu při nejrůznějších patologických poruchách a reakcích na příslušnou terapii. Podle našich zkušeností právě tento přístup vede ***k lepšímu pochopení*** složitých dynamických jevů v patogenezi nejrůznějších onemocnění a porozumění patofyziologickým principům příslušných léčebných zásahů.



Obr. 3 *"Vnitřek" simulačního čipu z obr. 2. Struktura připomíná elektrickou síť s propojenými integrovanými obvody, které v daném případě reprezentují simulační čipy nižší hierarchické úrovně. Na následujícím obrázku je znázorněn obsah čipu "Blood Acid Base Balkance"*

Implementace simulačních her do výukového programu, zvláště v oblasti biomedicíny, ale není triviální problém. Zde nestačí jen namalovat návrhy obrázků. Je zapotřebí nejprve ***vytvořit vlastní simulační model***. Obdobně jako je teoretickým podkladem letového

simulátoru více méně realistický model letadla, je v pozadí lékařského simulátoru model lidského organismu (resp. nějakého jeho subsystému).



Obr. 4 *Simulační čipy mají hierarchické uspořádání. Na obrázku je znázorněn "vnitřek" jednoho ze simulačních čipů z obr. 3. Vzhledem k tomu, že každý jednotlivý simulační čip obsahuje dostatečně podrobnou dokumentaci o svých vstupech a výstupech, může být struktura vztahů uvnitř simulačního čipu (reprezentující fyziologické vztahy v reálném organismu) srozumitelná fyziologům. Na dalším obrázku je zobrazen obsah čipu "BEINV".*

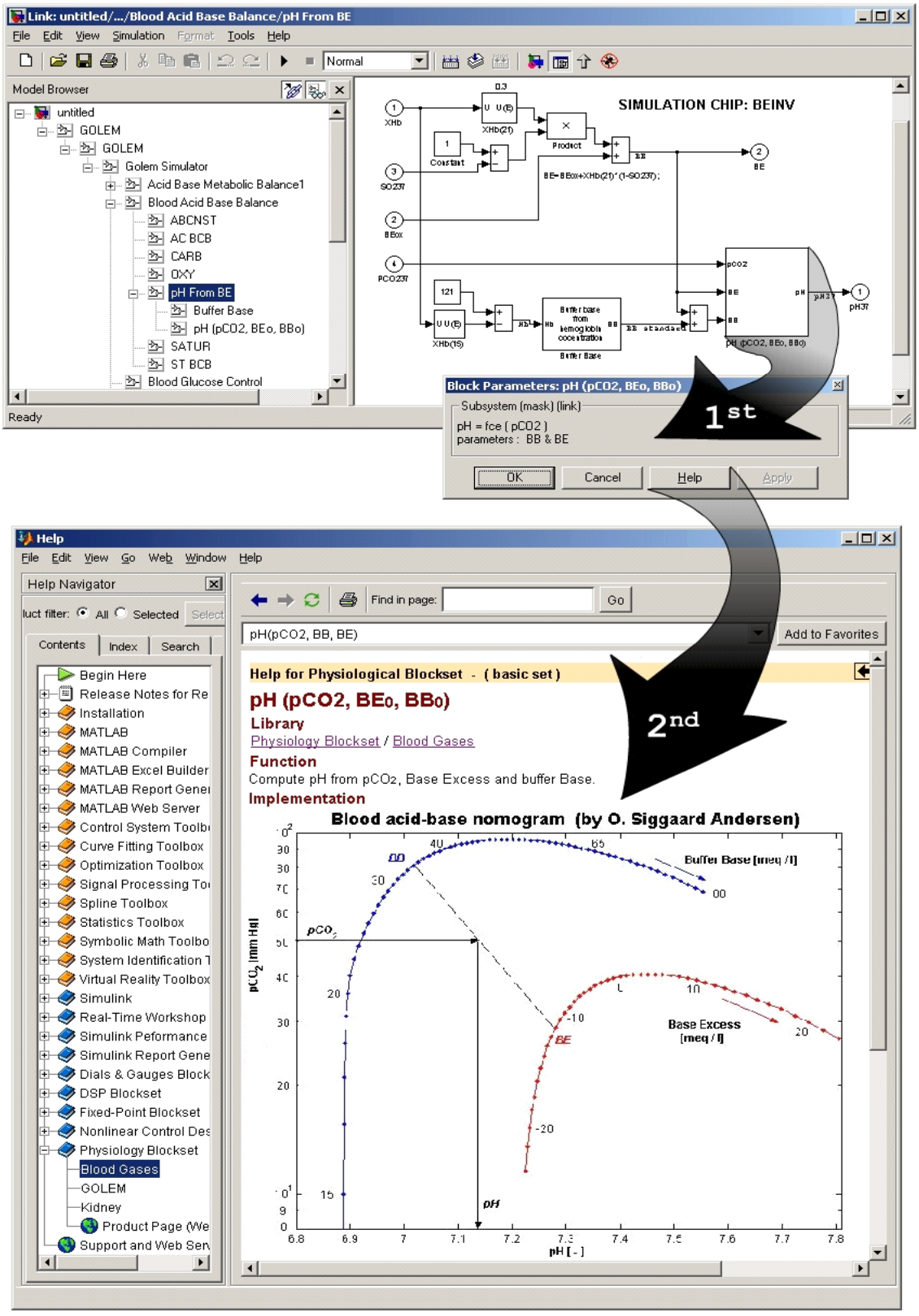
Vytváření simulačních modelů úzce souvisí s problematikou ***formalizace*** – tj. převedení představ o reálném světě z čistě verbálního popisu do popisu vyjádřeného formalizovaným jazykem (např. pomocí matematických výrazů).

Tento proces, který ve fyzice začal již 16. a 17. století a v chemii v 19. století, je v biologických a lékařských vědách opožděn a začal se rozvíjet až v druhé polovině dvacátého století zejména v souvislosti s bouřlivým rozvojem výpočetní techniky.

Mezinárodní úsilí v této oblasti našlo své vyjádření v novém mezinárodním projektu *PHYSIOME* (viz www.physiome.org), který je nástupcem úspěšně řešeného projektu GENOME. Zatímco cílem projektu GENOME bylo vytvořit mapu lidského genomu, úkolem projektu PHYSIOME je vytvoření kvantitativního, formalizovaného popisu biologického organismu – tj. formalizovaným způsobem popsat *jak* organismus funguje.

Tvorba simulačních modelů v biomedicínských vědách je tedy spíše ***výzkumná*** než vývojová práce, která má často multidisciplinární charakter – na jedné straně stojí ***systémový analytik*** - expert na formalizaci a tvorbu simulačních modelů (teoretický fyziolog vytvářející formalizovaný popis fyziologického systému a testující jeho chování pomocí simulačního modelu). Na druhé straně stojí klasický ***experimentální fyziolog*** či ***klinik***, pro kterého může být popis fyziologického systému pomocí diferenciálních rovnic španělskou vesnicí, ale který

při vlastní práci s modelem dokáže snadno rozpoznat, nakolik odpovídá chování počítačového simulačního modelu biologické realitě.

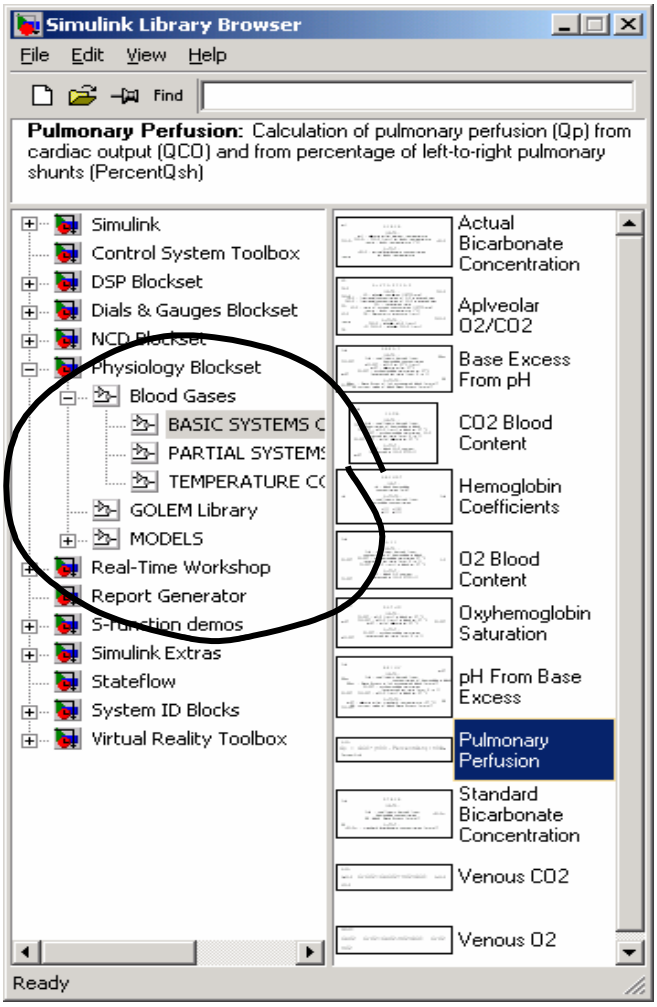


Obr. 5 *Simulační čipy na nejnižší hierarchické rovni jsou tvořeny propojenými základními komponentami vývojového systému Simulink Tyto propojené prvky reprezentují jednotlivé matematické vztahy. Ke každému čipu je dynamicky připojena příslušná dokumentační stránka, která obsahuje věcný popis funkce čipu včetně popisu matematických vztahů, které jsou jejím podkladem.*

3. Simulační čipy – nástroj interdisciplinární komunikace

Prostředí Simulinku umožňuje jednoduchým způsobem komunikaci mezi oběma skupinami specialistů zásadně usnadnit, pokud simulační model vhodně uspořádáme do jednotlivých hierarchicky propojených subsystémů s tím, že na levé straně umísťujeme vstupy a na pravé straně výstupy. Do masky subsystému píšeme seznam vstupů a výstupů a v helpové stránce podrobně popíšeme funkce subsystému tak, aby ho uživatel mohl snadno používat.

Simulační čipy svým chováním ***připomínají elektronické čipy***: analogicky jako se v elektronických obvodech ve vodičích připojených k jednotlivým pinům elektronických čipů rozvádí elektrický proud, tak se i v Simulinkových schématech k jednotlivým vstupním a výstupním "pinům" simulačních čipů rozvádí informace. Chování "čipu" lze *snadno testovat,* stačí k jeho vstupům přivést příslušné definované průběhy hodnot vstupních veličin a na jeho výstupy připojit příslušné virtuální osciloskopy či displeje. Obdobně, jako se v elektrických obvodech dá průběžně měřit napětí a proud pomocí měřících přístrojů, tak i v obvodech tvořených simulačními čipy v grafickém prostředí Simulinku se dají zobrazovat informace pomocí virtuálních displejů a osciloskopů připojených k jednotlivým propojkám. Obdobně jako v elektronických obvodech uživatele čipu zajímá chování čipu a nikoli jeho vnitřek, tak i v simulačních čipech při zkoumání chování je možné testovat pouze chování čipu a nikoli jeho nitro, které graficky reprezentuje simulinková síť použitých vztahů (graficky vyjádřené příslušné rovnice).



Obr. 6 *Simulační čipy ve vývojovém prostředí Simulink soustřeďujeme do hierarchicky uspořádaných knihoven. Vytvořili jsme tak knihovny popisující fyziologické vztahy při regulaci acidobazické rovnováhy, objemu a osmolarity vnitřního prostředí, výměnu krevních plynů, funkce respiračního systému, cirkulace a ledvin.*

Každý čip může v sobě obsahovat další propojené "simulační čipy". Obsáhlý simulační čip tak může mít poměrně složitou hierarchii (viz příklad na obr. 2-5).

Použitý "čip" může být objektem bez vazby (resp. s přerušenou vazbou) na knihovnu, nebo může být tvořen instancí čipu z knihovny – pak "čip" v knihovně představuje třídu a použitý čip je její instance – se všemi z toho plynoucími výhodami týkající se znovupoužitelnosti a využití dědičnosti (při modifikaci knihovního čipu).

Podle našich zkušeností důsledné využívání simulačních čipů při výstavbě simulačního modelu zásadně usnadňuje multidisciplinární spolupráci. Znamená to ale v prvé řadě věnovat důslednou pozornost dokumentaci. ***Simulační čipy*** samy o sobě mohou být i *aktuální elektronickou dokumentací* k vytvářeným modelům. Nejenže v čelní masce simulačního čipu je stručný popis všech vstupů a výstupů a "vnitřek" simulačního čipu graficky reprezentuje síť použitých vztahů, na kliknutí myší lze u každého softwarového simulačního čipu také

otevřít nápovědní okno s dalším podrobnějším popisem (viz obr. 5). ***Simulační čipy tedy umožňují pečlivě vést aktuální dokumentaci v elektronické podobě ke každému použitému subsystému při zachování veškeré funkčnosti.***

Výhodné je využívat toho, že Simulink umožňuje ***podrobnou dokumentaci přímo generovat***. Aby mohl jednotlivé komponenty využívat i ten, kdo danou část modelu nevytvářel, je podrobná a zároveň přehledná dokumentace nezbytná a čas strávený nad vypisováním spousty informací do masek jednotlivých subsystémů reprezentujících simulační čipy se pak zaručeně vyplatí.

Odměnou za trochu té dřiny je porozumění – experimentální fyziolog nemusí rozumět vnitřnímu uspořádání simulačního čipu, porozumí ale tomu, jaké chování má od fyziologického subsystému, který čip reprezentuje, očekávat. Fyziolog je nadto schopen porozumět i struktuře složené z propojených simulačních čipů – ze struktury modelu přímo vidí, které veličiny spolu vzájemně souvisejí (a ze znalosti fyziologie i dovede odhadnout, co a na jaké úrovni bylo zanedbáno).

Tak například fyziolog je schopen porozumět vnitřní struktuře simulátoru GOLEM [1-5], jehož jádro tvoří 18 propojených simulačních čipů (viz www.physiome.cz a obr. 3). Pokud by někdo chtěl jít i hlouběji do struktury modelu, nic mu v tom nebrání – na úrovni počítacích bloků se dostane i k příslušným rovnicím modelu, který je podstatou simulátoru GOLEM (36 diferenciálních rovnic, téměř 200 proměnných).

Ukazuje se, že idea simulačních čipů je vysoce přínosná jako nástroj vzájemného dorozumění i v mezinárodním projektu PHYSIOME. Pro unifikaci komunikace se v současné době rozpracovává metodika popisu simulačních čipů (včetně matematických výrazů) v jazyce XML (viz www.physiome.org).

4. Trnitá cesta od simulačních čipů k simulátorům

Simulační nástroje firmy Mathworks jsou určeny pro specialisty, ale pro běžného uživatele, který si chce se simulačním modelem jen "pohrát", se příliš nehodí. I když v prostředí těchto nástrojů je možné naprogramovat poměrně příjemné uživatelské rozhraní k ovládání vytvořeného modelu, ***pro účely uplatnění simulačního modelu ve výuce medicíny je nevýhodná nutnost spuštění licencovaného prostředí Simulinku/Matlabu***. Navíc, za pohodlí prostředí, určeného především pro vytváření (a nikoli provozování) simulačních modelů se platí tím, že u rozsáhlých modelů (a námi vytvářené modely mezi ně zpravidla patří) jsou nároky na výpočetní výkon počítače poměrně vysoké. Na méně výkonných počítačích pak simulace probíhá neúměrně pomalu.

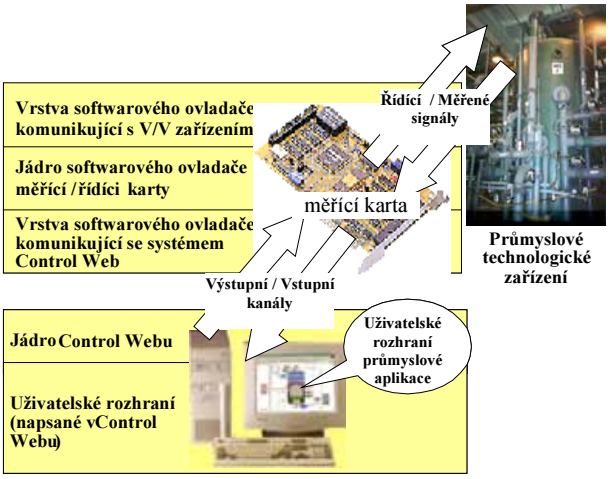
Pro vývoj simulátorů ve výukových programech bylo proto nutné poohlédnout se po jiném vývojovém prostředí než je Matlab a Simulink. Po určitém váhání jsme pro tvorbu uživatelského rozhraní simulátorů, raději než po obecných softwarových nástrojích (typu Delphi, Visual C++ aj.), sáhli po ***nástrojích využívaných při tvorbě průmyslových aplikací*** (měřících ústředen a velínů). Vedly nás k tomu především dva důvody:

Se simulačním modelem chceme v simulátoru fyziologických funkcí zacházet obdobně, jako se v průmyslu z velínu řídí složité technologické zařízení: chceme číst (a v nejrůznější grafické či číselné podobě zobrazovat) množství nejrůznějších měřených dat (jako v průmyslové měřící ústředně) a zároveň chceme jednoduchým způsobem (stiskem tlačítek, otáčením knoflíků, popotahováním táhel apod.) simulační model ovládat (obdobně jako se z velínu řídí nějaká technologie).

Druhým důvodem, proč jsme sáhli po softwarovém nástroji z průmyslu, je ***spolehlivost***. Požadavky spolehlivosti, kladené na nástroje, jejichž pomocí se vyvíjejí průmyslové řídící aplikace jsou obvykle řádově vyšší, než u obecných programovacích nástrojů.

Nástrojů pro design průmyslových aplikací je na světovém trhu nemálo. Jejich ceny jsou ovšem, na rozdíl od obecných softwarových nástrojů, zpravidla velmi vysoké.

Paradoxem je, že velmi dobrý nástroj lze získat velmi levně, a to přímo od domácího výrobce. Zlínská akciová společnost ***"Moravské přístroje"*** již léta vyvíjí systém pro tvorbu průmyslových aplikací s názvem ***"Control Web".*** Jde přitom o kvalitní vývojový systém: na letáčku jejich skandinávského distributora stojí "Svensk kvalitet (till Tjeckiska priser)" – "Švédská kvalita (za české ceny)". Švédové jsou velmi domýšliví na kvalitu svých výrobků a pokud o vývojovém nástroji ***Control Web*** z Valašska veřejně prohlašují, že dosahuje "švédské kvality", je to velké ocenění skupiny tvůrců, kteří na tomto produktu usilovně pracují již od počátku devadesátých let. Nyní je jejich úsilí korunováno tím, že jejich programové dílo (jehož zdrojový text obsahuje více než 2 milióny řádek) nachází úspěšnou cestu na zahraniční trhy. Poslední verzi programu je tak možno vidět v nejen v anglické či německé verzi, ale i v "rozsypaném čaji" japonských znaků.



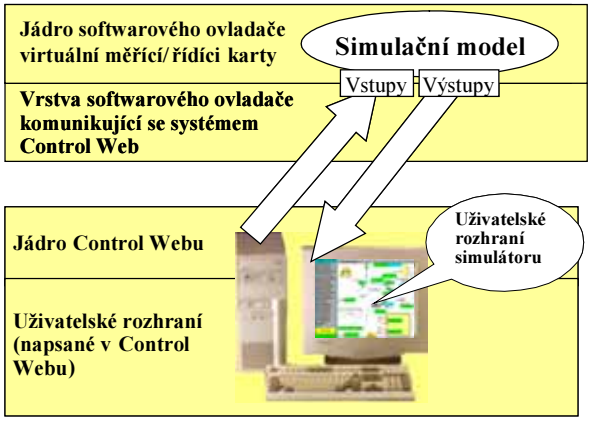
Obr. 7 *Komunikace systému Control Web s ovladačem řídící/měřící karty při tvorbě průmyslových aplikací.*

Control Web je především určen pro ***vývoj průmyslových vizualizačních a řídících aplikací na platformě WIN32*** - sběr, ukládání a vyhodnocování dat, tvorba rozhraní člověk-stroj aj. (viz http://www.mii.cz). Základními stavebními kameny uživatelské aplikace vytvářené v prostředí Control Web jsou ***virtuální přístroje*** (komunikující mezi sebou pomocí proměnných a zpráv). ***Měřené hodnoty*** z vnějšího světa jsou v průmyslových aplikacích virtuálním přístrojům zprostředkovány přes ***vstupní kanály***, ***řídící signály*** mohou virtuální přístroje posílat do okolí pomocí ***výstupních kanálů.***

Pro vývoj uživatelského rozhraní poskytuje systém Control Web velmi výkonné prostředky. Tak např. z palety virtuálních přístrojů je možno snadno tažením myši vytáhnout potřebný přístroj a umístit ho na příslušný panel a v interaktivním dialogu mu nastavit hodnoty jeho příslušných atributů, nadefinovat jeho lokální proměnné, či individuální procedury (metody objektu) apod.

5. Kontejnery pro simulační modely

Abychom mohli využít vývojářské pohodlí systému Control Web, museli jsme použít vcelku jednoduchý trik. V průmyslových aplikacích Control Web komunikuje (prostřednictvím příslušných softwarových kanálů) přes ovladač příslušné měřící/řídící karty s průmyslovým technologickým zařízením.



Obr. 8 *Začlenění simulačního modelu do ovladače "virtuální karty" při tvorbě simulátoru v prostředí Control Web.*

Je ovšem možné napsat ***speciální ovladač***, jehož interní součástí je simulační model. Tento ovladač je schopen komunikovat (přes softwarové kanály) s objekty systému Control Web, ale na rozdíl od ovladačů ke skutečným měřícím a řídícím kartám nekomunikuje s hardwarem ale

komunikuje se simulačním modelem. Pokud se ovladač napíše dobře, je systém Control Web "ošálen": vstupní kanály (k zobrazovacím prvkům na monitoru) považuje za skutečné měřené signály někde z technologického okolí počítače, zatímco ve skutečnosti to jsou výstupní proměnné simulačního modelu. A také v opačném směru je Control Web přesvědčen, že výstupní kanály, které odcházejí od řídících prvků systému Control Web, nastavují přes příslušný ovladač nějaké aktivní prvky průmyslového zařízení, ale ony namísto toho mění vstupy simulačního modelu.

Abychom usnadnili vývoj ovladačů jakési "virtuální měřící/řídící karty", které obsahují simulační model a nemuseli tento ovladač pro každý model psát v C++ "ručně", vyvinuli jsme speciální program (tzv. průvodce), který nám ***umožní vývoj tohoto ovladače automatizovat***. Máme tedy nyní možnost ***bezprostředně ze simulinkového schématu generovat zdrojový text příslušného virtuálního ovladače v C++.*** Tím je možné jednoduše a rychle modifikovat ovladač pro prostředí Control Web při nejrůznějších úpravách simulačního modelu v prostředí Simulink. To nám umožňuje vyvíjet (aktualizovat, modifikovat) simulační modely v prostředí Matlab a Simulink a ze simulačního modelu jednoduše vygenerovat (a okamžitě nechat přeložit) aktuální verzi ovladače pro Control Web se zapouzdřeným simulačním modelem.

Další nespornou výhodou pro tvorbu simulátorů v prostředí systému Control Web je to, že (jak ostatně naznačuje i jeho název) systém ***umožňuje tvorbu skutečně distribuovaných řešení*** v intranetových/internetových sítích. To dává možnosti vytvářet distribuovanou výukovou aplikaci, kdy vlastní numericky náročné výpočty simulátoru probíhají na serveru (Control Web serveru) a vizualizace probíhá na klientech.

Dalším prostředím, které využíváme pro tvorbu multimediálních interaktivních simulátorů je prostředí ***Microsoft .NET.*** Prostředí Microstoft.NET je vhodným kontejnerem pro simulační model i pro vložené interaktivní grafické komponenty (propojitelné s modelem). Simulační model pak může komunikovat s interaktivními grafickými komponentami (vytvořenými např. pomocí nástrojů firmy Macromedia), jejichž chování může být řízeno na základě výstupů ze simulačního modelu. Konec konců, v prostředí Microsoft .NET můžeme snadno zabudovat aplikaci vytvořenou v Control Webu do komplexnějšího celku.

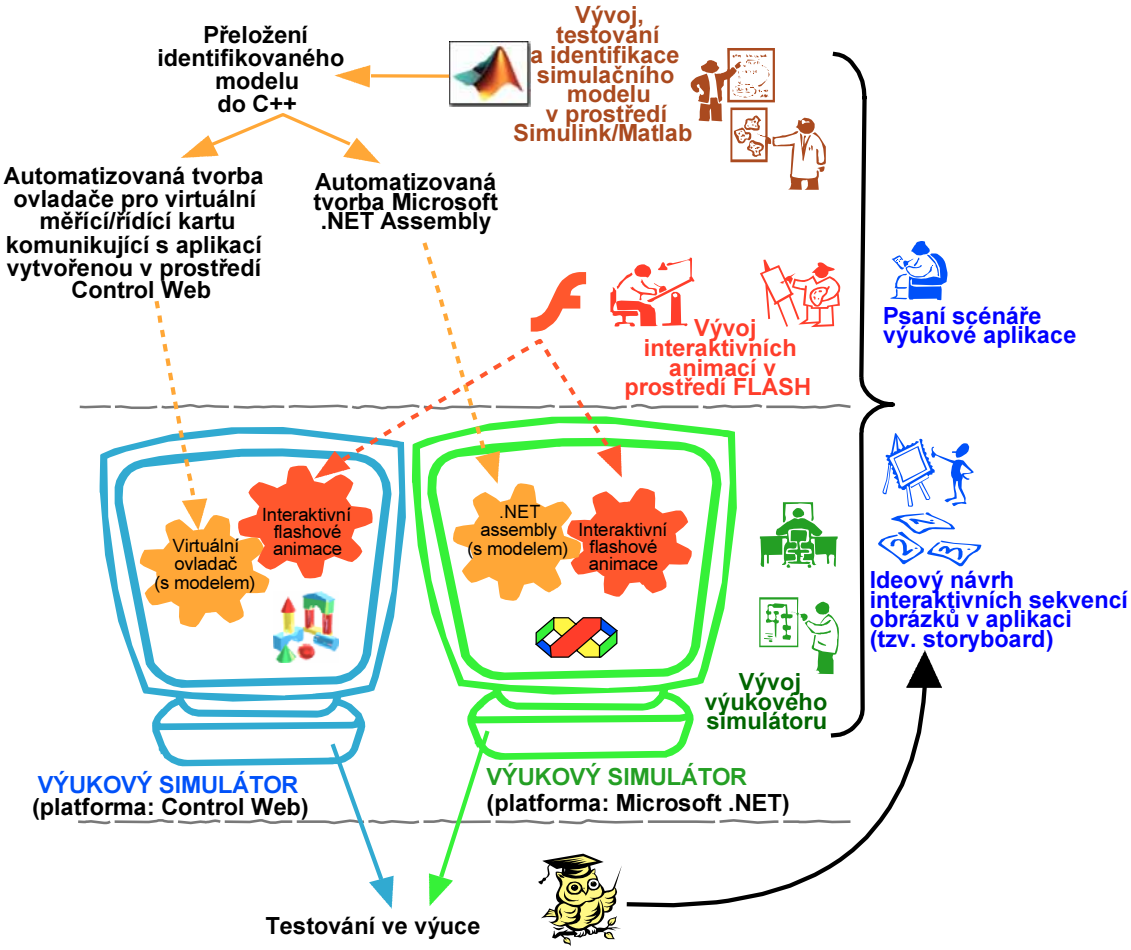
Obdobně jako jsme vytvořili speciální nástroj, který z modelu v prostředí Simulinku automaticky vytváří "virtuální ovladač" pro Control Web, naprogramovali jsme i nástroj pro ***automatický převod simulačního modelu vytvořeného v prostředí Simulink/Matlab do prostředí Microsoft .NET.*** Nezanedbatelnou výhodou tohoto nového systémového prostředí od Microsoftu jsou i nové možnosti, které pro tvorbu webových e-learninogvých aplikací přináší technologie ASP .NET.

6. Propojení multimediálních prvků a simulačních modelů

Control Web i Visual Studio .NET nabízí pro vizualizaci simulátorů celou řadu vlastních virtuálních přístrojů a komponent. Pokud však chceme vytvářet vlastní virtuální přístroje – interaktivní animované obrázky, které jsou řízeny podle proměnných modelu, musíme sáhnout po dalším nástroji. Tímto nástrojem je ***Macromedia Flash MX*** – vývojové prostředí, umožňující vytvářet animované interaktivní komponenty, jejichž chování se dá programovat [1]. Vytvořené komponenty se dají přehrávat (pomocí vestavěného interpretu, volně stažitelného z Internetu) přímo ve webových stránkách, nebo se dají využít jako ActiveX komponenty v jiných programech.

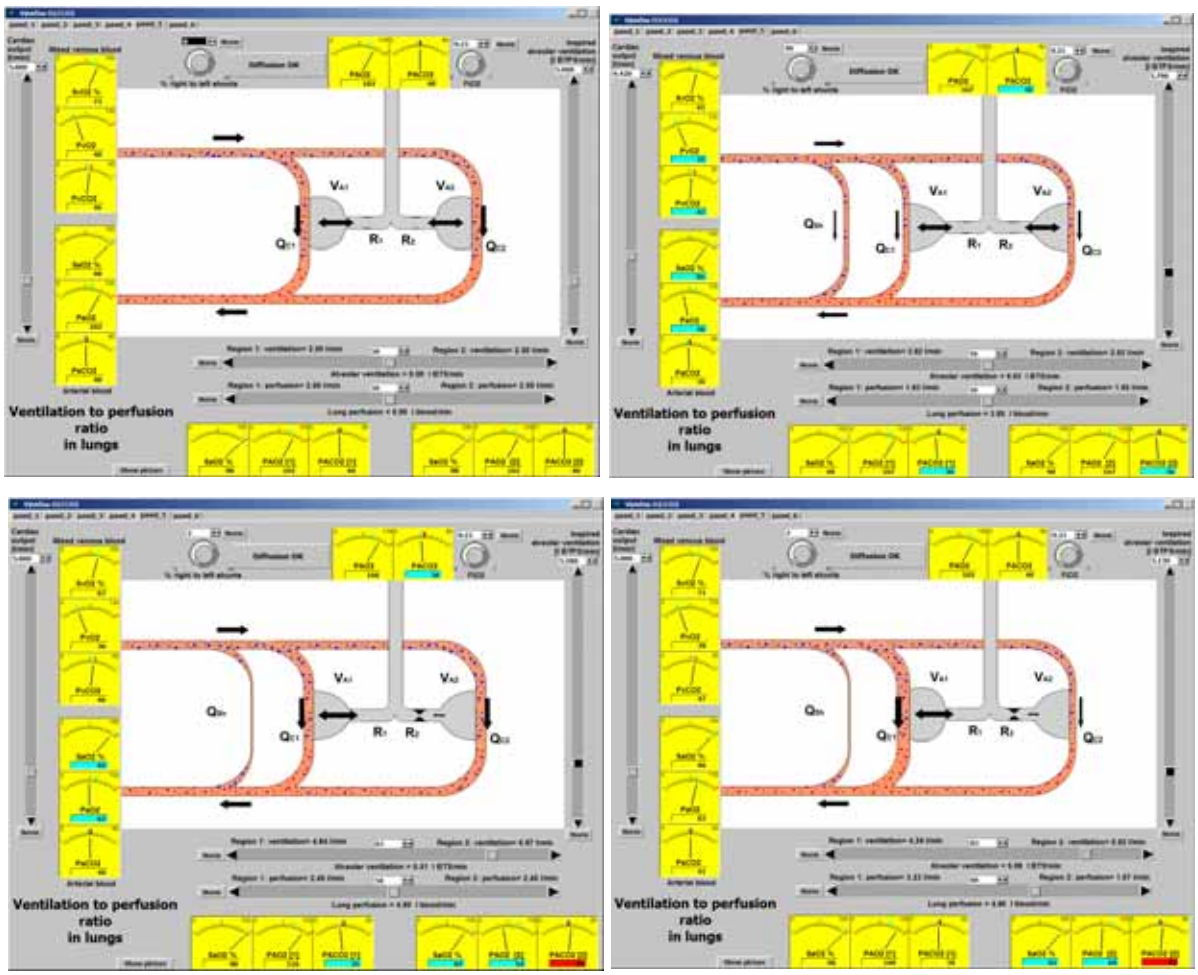
Velký úspěch vývojového prostředí Macromedia Flash (a Macromedia Director) je mimo jiné založen na tom, že se jejich tvůrcům poměrně úspěšně podařilo definovat rozhraní pro výtvarníky (vytvářející základní animační prvky) a programátory, kteří těmto komponentám mohou pomocí speciálního objektového jazyka (Macromedia Action Script) vdechnout interaktivnost. Náš vývojový tým vytvořil pro vývoj interaktivní grafiky speciální laboratoř na

Grafické škole Václava Hollara a věnoval velké úsilí tomu, aby profesionální výtvarníky naučil pracovat se systémem Macromedia Flash a Macromedia Director.



Obr 9. *Vývojový cyklus tvorby výukových simulátorů. Na počátku je sestavení scénáře výukového programu včetně návrhu sekvencí obrázků (tzv. storyboard). Pak následuje tvorba simulačních modelů, které budeme využívat v simulátoru, resp. výukovém programu využívajícím simulační hry. Pro tvorbu simulačních modelů využíváme vývojové prostředí Matlab/Simulink od firmy Mathworks. Zároveň v naší laboratoři interaktivní grafiky vytváříme pohyblivé animační obrázky v prostředí Macromedia Flash. Odladěný simulační model je pak implementován ve formě řadiče virtuální měřící/řídící karty do prostředí Control Web od firmy Moravské přístroje, v němž je vytvořeno uživatelské rozhraní. Řadič virtuální karty, obsahující simulační model, toto prostředí "ošálí" – výstupy modelu jsou v prostředí Control Web interpretovány jako měřené signály z technologie a vstupy modelu jsou interpretovány jako řídící signály směřující do technologického zařízení. Flashové animace jsou do vytvářené výukové aplikace umístěny jako Active X komponenty a propojeny se vstupy/výstupy simulačního modelu. Animace pak mohou být řízeny simulačním modelem a do simulačního modelu mohou zároveň přicházet hodnoty vstupů generované interakcí uživatele s flashovou grafikou. Další platformou, kterou jsme v poslední době začali využívat při tvorbě výukových simulátorů je prostředí Microsoft .NET. Do něj umisťujeme jak simulační model ve formě automaticky generovaného .NET assembly ze Simulinku, tak i flashovou interaktivní animaci (případně do něj můžeme umístit i celou aplikaci vytvořenou v nové verzi prostředí Control Web). Důležité je otestování výukových simulátorů ve výuce, které přináší nové požadavky pro revizi, rozšíření či vytvoření dalších výukových simulátorů.*

Naším cílem je využít možností interaktivní grafiky vytvářené pomocí vývojových nástrojů Macromedia pro vizualizaci chování simulačních modelů. Propojení interaktivní animace vytvořené pomocí Macromedia Flash (či Macromedia Director) v podobě ActiveX komponenty s jejím vnějším okolím je přes komponentové rozhraní velmi snadné.



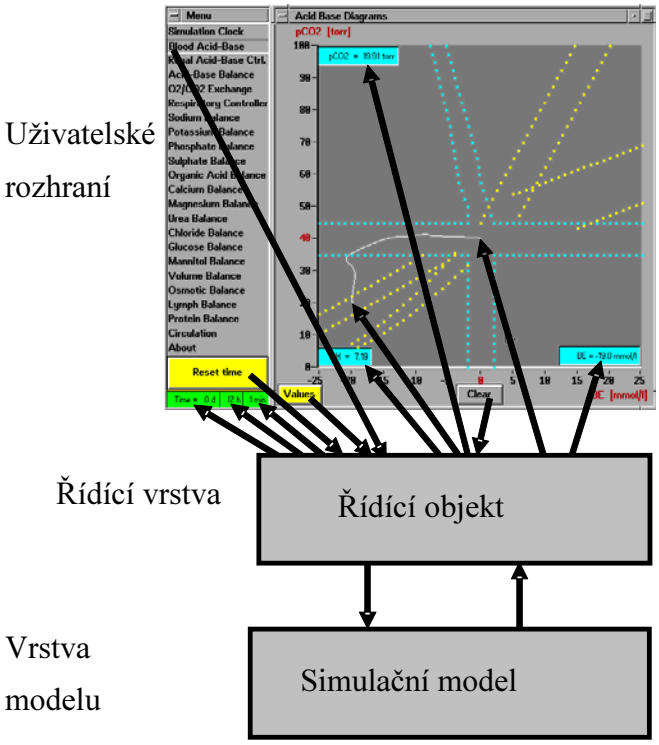
Obr. 10. *Ukázka různých stavů vizuálního rozhraní v simulační hře ve výukovém programu poruch respirace. Uživatelské rozhraní, vytvořené v prostředí Control Web s využitím flashové animace, schematicky zobrazuje různé stupně ventilace a perfúze dvou částí plic. Student si "hraje" s pohyblivým obrázkem a model na pozadí propočítává příslušné fyziologické parametry, zobrazované jako hodnoty virtuálních měřících přístrojů (v Control Webu). Zároveň se mění "hloubka dýchání" a "velikost prokrvení" plic na schematickém obrázku tvořeném flashovou animací.*

Tyto komponenty snadno dostaneme do prostředí Visual Studia .NET, kde můžeme vytvářet "pohyblivé interaktivní obrázky" řízené modelem. Obdobně můžeme vkládat interaktivní animace do simulátorů vytvářených v prostředí Control Web. Jedním z virtuálních přístrojů v prostředí Control Web je totiž ***kontejner pro komponenty ActiveX***, který tak představuje most mezi systémem Control Web a vlastnostmi a metodami (OLE Automation) ActiveX komponent. To znamená, že do aplikace lze zabudovat ActiveX komponenty a programově je ovládat - nastavovat jim vlastnosti a volat metody z procedur jakýchkoliv přístrojů.

Můžeme tedy např. pomocí systému Macromedia Flash MX vytvořit interaktivní výukové animace a ty pak uložit jako ActiveX komponenty do kontejneru. ***Animace pak mohou být řízeny na základě výstupů implementovaného simulačního modelu*** – např. schematický obrázek cévy se může roztahovat nebo komprimovat, plicní sklípek může hlouběji či mělčeji "dýchat" atd. Na druhé straně můžeme ***přes vizuální prvky*** vytvořené ve Flashi (nejrůznější tlačítka, knoflíky, táhla apod.) ***do simulačního modelu zadávat vstupy***.

Příklad využití flashové animace ve výukovém programu využívajícím simulační hru je zobrazen na obr. 10. Interaktivní animace, vytvořená v prostředí Macromedia Flash byla umístěna do kontejneru pro ActiveX komponentu v programu vytvořeném v prostředí Control Web. Ručičky virtuálních přístrojů zobrazují příslušné fyziologické hodnoty načítané z běžícího simulačního modelu na pozadí a zároveň se příslušně mění chování flashové animace.

Model běžící na pozadí (realizovaný ve formě řadiče virtuální měřící/řídící karty v případě aplikace v Control Webu, nebo jako .NET assembly v případě aplikace implementované v Microsoft .NET) komunikuje s vizuálními elementy na popředí. V případě flashových animací je změna zobrazovaných hodnot či požadavek na změnu vstupních dat modelu realizován pomocí událostí, které jsou příslušně obslouženy a změní nastavení vizuálních elementů či vstupní hodnoty modelu. Prostředí Control Webu nebo .NET je tak kontejnerem pro simulační modely, multimediální prvky i interaktivní (se simulačním modelem komunikující) animace.



Obr. 11. *Tzv. UCM architektura při tvorbě simulátorů. Mezi vrstvu modelu a vrstvu uživatelského rozhraní je vhodné vložit řídící vrstvu kam jsou směrovány veškeré zprávy a události vznikající ve virtuálních přístrojích uživatelského rozhraní a kam je zároveň směrována veškerá komunikace s modelem. V této vrstvě se řeší veškerý kontext zobrazovaných dat a příslušné požadavky na komunikaci s modelem. Veškerá logika zobrazování a komunikace je pak soustředěna do jednoho místa což podstatně ušetří čas při modifikacích uživatelského rozhraní nebo změnách modelu.*

7. UCM architektura

V případě složitější architektury simulátoru může být logika změn poměrně složitá, proto je vhodnější mezi vrstvu vizuálních elementů a vrstvu simulačního modelu vložit řídící vrstvu, kde se na jednom místě řeší veškerá logika komunikace uživatelského rozhraní s modelem a kde je ukládán i příslušný kontext (obr. 11). Toto uspořádání je nezbytné při složitějších modelech a simulátorech, jejichž uživatelské zobrazení je reprezentováno mnoha virtuálními přístroji na více propojených obrazovkách, které využívají kontextově závislé ovládací prvky. Výhody tohoto uspořádání zvláště vyniknou při modifikacích jak modelu, tak i uživatelského rozhraní. V literatuře [2] se hovoří o tzv. UCM architektuře výstavby simulátorů (User interface – Control layer – Model layer).

Velmi se nám osvědčilo využít v tomto jádře hierarchický stavový automat (který nám nejlépe zapamatuje příslušný kontext) v pojetí D. Harela [2]. Hierarchický stavový automat nepoužívá k uchování kontextu aplikace (simulačního modelu a uživatelského rozhraní) nepřehlednou sadu stavových proměnných, ale přehlednou strukturu vnořených stavů s jasně viditelnými kritérii přechodu mezi stavy a reakcemi na změnu stavů. Všechny obslužné akce vázané na jeden stav jsou soustředěny na jednom místě v kódu a změna struktury stavového automatu je triviální. Harelovu koncepci hierarchických stavových automatů využívá

prostředí Matlab/Simulink v toolboxu Stateflow, kde automaty jsou bezprostředně propojitelné se simulačním modelem v Simulinku (a umožňují zapamatovat kontext modelu). Stavový automat je však možno realizovat i na straně uživatelského rozhraní – přímo ve Flashi. Tento automat pak bezprostředně komunikuje s vizuálními objekty ve Flashi. Obdobným způsobem konstruují simulátory Kaye a Castillo [3]. V poslední době jsme vytvořili nástroj [7], který umožňuje vizuálně navrhnout a funkčně otestovat komunikující hierarchické stavové automaty (včetně příslušných událostí a akcí) v prostředí C# .NET. Tyto automaty jsme s úspěchem využili jako základ střední vrstvy UCM[1].

8. Závěr

Zdá se, že pomalu končí doba, kdy vytváření výukových programů bylo otázkou entuziasmu a píle skupin nadšenců. Tvorba moderních výukových aplikací je náročný a komplikovaný projekt, vyžadující ***týmovou spolupráci*** řady profesí – od zkušených učitelů, jejichž scénář je základem kvalitní výukové aplikace, přes systémové analytiky, kteří jsou ve spolupráci s profesionály daného oboru odpovědni za vytvoření simulačních modelů pro výukové simulační hry, výtvarníky, kteří vytvářejí vnější vizuální podobu, až po programátory, kteří celou aplikaci "sešijí" do výsledné podoby. Aby tato interdisciplinární kolektivní tvorba byla efektivní, je nutno pro každou etapu tvorby využívat ***specifické vývojové nástroje***, s dostatečnou technickou podporou, které umožňují komponentovou tvorbu simulačních modelů, vytváření interaktivních multimédií a jejich závěrečné propojení podle daného scénáře do kompaktního celku [4-6].

9. Literatura

1. Hall, B., Wann, S. (2003): *Object-oriented programming with ActionScript.* New Riders, Boston, Indianopolis, London, Munich, New York, san Francisco, 2003, ISBN 0-7357-1183-6.
2. Harel, D.: Statecharts: A visual formalism for complex systems. *Science of Computer Programing,* vol. 8 (1987), pp. 231-274.
3. Kaye J., Castillo D. (2003): *Flash MX for interactive simulation.* Thomson Delmar Learning Inc. New York, 2003, ISBN 1-4018-1291-0.
4. Kofránek, J., L. D. Anh Vu, H. Snášelová, R. Kerekeš, T. Velan, (2001): GOLEM – Multimedia simulator for medical education. In: Studies in Health Technology and Informatics., vol. 84. *MEDINFO 2001, Proceedings of the 10th World Congress on Medical Informatics.* (editors: V.L. Patel, R. Rogers, R. Haux), IOS Press, Amsterdam, Berlin, Oxford, Washington DC, 2001, pp. 1042-1046.
5. Kofránek, J.,T. Kripner, M. Andrlík, J. Mašek (2003): Creative connection between multimedia, simulation and software development tools in the design and development of biomedical educational simulators, In*: Simulation Interoperability Workshop*, Position papers, Volume II, Orlando, FALL 2003, pp. 677 - 687, ISBN 1-930638-32-9
6. Kofránek, J, P. Maruna, M. Andrlík, P. Stodulka, T. Kripner, Z. Wünsch, P. Maršálek, D. Smutek, Š. Svačina (2004): The design and development of interactive multimedia in educational software with simulation games. In: *Proceedings of the Seventh IASTED International Conference on Computer Graphics And Imaging*, Kauai, Hawai, USA 2004, pp. 164-170. ISBN: 0-88986-418-7
7. Stodulka, P.: Vývojový nástroj pro návrh a testování komunikujících stavových automatů. Diplomová práce. MFF UK Praha, 2004.

Spojení na autory: Dr Jiří Kofránek, CSc, Biokybernetické odd. ÚPF, U Nemocnice 5, 12853 Praha 2, tel. 22496 2793, GSM 777 68 68 68m e-mail: kofranek@kofranek@cesnet.cz

---

[1] Práce na vývoji lékařských simulátorů a tvorba výukových programů se simulačními hrami je podporována **Výzkumným záměrem MSM-111100008 (physiome.cz)** a **Rozvojovým programem MŠMT č. 394**.