TECHNOLOGIE A TRENDY TVORBY VÝUKOVÝCH SIMULÁTORŮ

Jiří Kofránek, Pavol Privitzer, Petr Stodulka

Anotace
Autoři popisují stávající a budoucí technologii vytváření výukových simulátorů. Zdůrazňují oddělené řešení dvou problémů – tvorbu simulačních modelů a tvorbu vlastních simulátorů, pro něž musí být zvoleny různé nástroje. Budoucnost nástrojů pro tvorbu simulačních modelů vidí v programovacím jazyku Modelica, pro vytváření simulátorů se orientují na platformu .NET. Plánují vytvořit implementaci vývojového prostředí Modelicy v platformě .NET. Prozatím vytvořili pro projekt OpenModelica rozšíření pro konverzi modelu z jazyka Modelica do jazyka C#.

Klíčová slova
Výukové simulátory, simulační hry, fyziologické modelování, Modelica, .NET

1. Úvod – škola (simulační) hrou

Výukové programy se simulačními komponentami ***nejsou jen multimediální náhradou klasických učebnic*.** Jsou zcela ***novou výukovou pomůckou***, kde nachází své moderní uplatnění staré krédo Jana Ámose Komenského "Schola Ludus" (škola hrou), které tento evropský pedagog razil již v 17. století. Spojení multimediálního prostředí se simulačními modely studentům umožňuje pomocí experimentů se simulačním modelem názorně prozkoumat vykládaný problém ve virtuální realitě a přináší tak zcela nové možnosti pro vysvětlování složitých problémů. Simulační hry umožňují názorně vysvětlit komplexní vztahy ve fyziologických regulačních systémech a kauzální řetězce v patogenezi nejrůznějších onemocnění.

Vývoj efektivních výukových programů kombinujících multimédia se simulačními hrami je ale náročnou a komplikovanou prací, vyžadující týmovou spolupráci řady profesí – zkušených pedagogů vytvářejících základní scénář, tvůrců simulačních modelů, lékařů, výtvarníků a programátorů. Tuto interdisciplinární kolektivní tvorbu zefektivňuje využívání vhodných vývojových nástrojů, které umožňují komponentovou tvorbu, propojení simulačních programů a interaktivních multimédií podle daného scénáře do kompaktního celku.

Jedním z projektů, kde chceme využít nové možnosti multimédií a simulačních modelů pro výuku, je počítačový ***Atlas fyziologie a patofyziologie***[1].

Atlas [7] je koncipovaný jako multimediální výuková pomůcka, která by názornou cestou prostřednictvím Internetu a s využitím simulačních modelů měla pomoci vysvětlit funkci jednotlivých fyziologických systémů, příčiny a projevy jejich poruch. Viz http://physiome.cz/atlas. Atlas je vytvářen jako společné dílo tvůrčího týmu odborníků různých profesí - lékařů, programátorů, systémových inženýrů a výtvarníků[2]. ***Projekt atlasu je otevřený*** – jeho výsledky v českém jazyce na internetu volně zpřístupňujeme všem zájemcům a při jeho vývoji uvítáme spolupráci se všemi, kdo se budou chtít podílet na jeho postupném budování.

Ukažme si nyní, co vytváření obdobné interaktivní výukové pomůcky vyžaduje, co je v zákulisí tvorby výukových simulátorů a jaké jsou současné i budoucí možnosti technologií, které tvorbu těchto moderních výukových pomůcek usnadňují.

2. Scénáristé výukových her

Obdobně jako sebelepší učebnice zcela nenahradí učitele, tak ho nevytlačí ani seberafinovanější výukový program. A stejně tak jako klíčovým faktorem pro úspěch vysokoškolských učebnic je schopnost jejich autorů názorně a věcně správně vyložit složitou látku, tak pro zdar výukového programu je nejdůležitější jeho ***kvalitní scénář***. Napsat dobrý učební text vysvětlující složité procesy není jednoduché a veškeré pomůcky (multimédia, interaktivní animace, simulační modely apod.) – tedy vše co si lze představit pod donekonečna omílaným slovem e-learning – jsou bez dobrého scénáře vlastního výukového programu či bez zkušeného učitele, který ví, jak složitý simulační model pedagogicky využít, jen pouhou módní ozdobou.

Zkušenosti s využitím simulátorů ve výuce ukazují, že sebesložitější simulátor je bez doprovodu jasného scénáře jeho výukového využití jen těžko ovladatelným počítačovým "Frakensteinem". Příkladem z poslední doby může být simulátor "Quantitative Circulatory Physiology" [1] a jeho nástupce, v současné době snad nejsložitější model fyziologických regulací, "Quantitative Physiological Human"[3], který umožňuje vhodným nastavováním cca 750 parametrů simulovat nejrůznější patologické stavy. Autoři simulátoru ho na své univerzitě s úspěchem využívají ve výuce. Pro ty, kdo složitý simulátor dobře neznají, je však sledování stovek proměnných nadmíru obtížné.

[1] http://www.physiome.cz/atlas
[2] více o technologii výstavby atlasu: http://www.physiome.cz/atlas/info/01/index.htm
[3] dá se stáhnout z adresy: http://physiology.umc.edu/themodelingworkshop/

Z vlastní zkušenosti víme, že pro autory je mnohem kreativnější zamýšlet se nad vylepšením struktury a funkce vlastního simulátoru, než psaní učebních textů a návodů. Bez jasně popsaného scénáře pedagogického využití rafinovaně složitého simulátoru však úsilí, které tvůrci po léta věnují jeho výstavbě, nepřináší kýžený pedagogický efekt. Pro efektivní využití simulátorů by proto scénáře jejich využití ve výuce neměly být opomíjeny.

U složitých simulátorů je z pedagogického hlediska výhodné, když modelovaný objekt můžeme rozdělit na jednotlivé subsystémy a testovat jejich chování odděleně i jako součást vyššího celku. Tím studentům umožníme 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. Tak například v našem simulátoru Golem [6] můžeme postupně odpojovat a zapojovat jednotlivé regulační smyčky ovlivňující homeostázu vnitřního prostředí a tím postupně testovat význam jednotlivých regulačních okruhů a ozřejmit si tím například souvislosti poruch iontové a acidobazické rovnováhy. Podle našich zkušeností právě tento přístup vede k lepšímu pochopení složitých dynamických jevů v patogenezi nejrůznějších onemocnění a porozumění patofyziologických principů příslušných léčebných zásahů.

Ideální je, když jsou simulátory přímo začleněni do multimediálního výukového programu ve formě ***simulační hry***. Příkladem jsou simulační hry ve výše zmiňovaném Atlasu fyziologie a patofyziologie, o nichž jsme referovali na minulých dvou seminářích MEDSOFT.

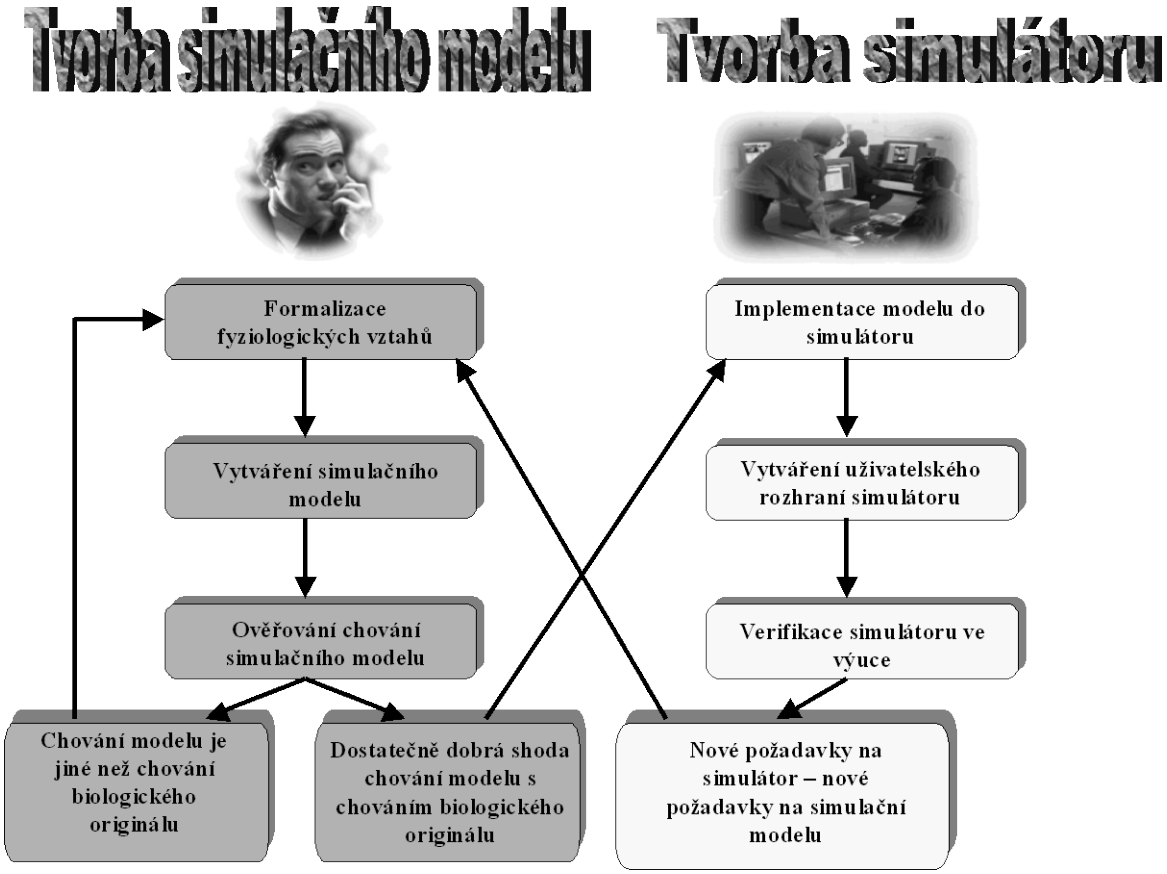
Pro začlenění multimediálních animací do simulačních her a výkladu je podkladem pro tvorbu scénáře nejen (hyper)textová část, ale i jakýsi ***"storyboard"***, který, obdobně jako při produkci kresleného filmu, naznačuje výtvarníkům, jaké mají vytvářet animované obrázky. Důležité je i vytvoření grafického manuálu, který sjednotí grafickou podobu celé aplikace.

2. Dva typy problémů – dva druhy nástrojů

Při vytváření simulátorů a výukových simulačních her je nutno řešit dva typy problémů:

1. ***Tvorba simulačního modelu*** – vlastní teoretická výzkumná práce, jejímž výsledkem je formalizace fyziologických vztahů vyjádřená matematickým modelem.

2. ***Tvorba vlastního multimediálního simulátoru a výukového programu využívajícího simulační hry*** – vývojová práce, která navazuje na vytvořené (a verifikované) matematické modely.



Obr. 1 *Dva typy problému při tvorbě výukových simulátorů*

Tvorba simulačního modelu i tvorba simulátoru spolu navzájem úzce souvisejí (viz obr. 1). Předpokladem pro tvorbu výukového simulátoru je dostatečně dobře verifikovaný model a na druhou stranu využití simulátoru ve výuce přináší nové požadavky na vytvoření nových či modifikaci stávajících simulačních modelů. Každý z těchto problémů má svou specifiku a je proto výhodné ***pro každou etapu použít zcela odlišné vývojové nástroje***.

Vytvoření vlastního simulátoru je spíše vývojářskou prací, která vyžaduje skloubit nápady a zkušenosti pedagogů, vytvářejících scénář výukového programu, kreativitu výtvarníků, vytvářejících interaktivní multimediální komponenty a úsilí programátorů, kteří "sešijí" výsledné dílo do konečné podoby.

Formalizace fyziologických vztahů a vytváření simulačního modelu není vývojářský, ale (poměrně náročný) výzkumný problém, jehož efektivní řešení vyžaduje použít adekvátní nástroje. Těchto nástrojů je celá řada – od profesionálních firemních nástrojů např. Matlab/Simulink

od firmy Mathworks, které používáme v naší laboratoři až po nástroje typu open-source, jako např. JSIM [8, 9], vyvíjený na Washington University a používaný pro knihovnu fyziologických modelů v rámci celosvětového projektu Physiome[4], koordinujícího aktivity v oblasti formalizace popisu fyziologických regulací [2,4].

Někdy se nástroje pro tvorbu simulačních modelů zároveň využívají i jako nástroje pro tvorbu výukových simulátorů. To ovšem v konečném důsledku vede k určitému omezení, danému výrazovými prostředky, které má příslušný vývojový nástroj k dispozici. Častější je opačná situace – programování simulačního modelu přímo v prostředí běžného programovacího jazyka na základě struktury modelu známé z literatury. To je obvykle možné u jednodušších modelů. U složitějších simulátorů je tento postup obtížnější, protože struktura rozsáhlých simulačních modelů se ne vždy kompletně zveřejňuje (rovnice modelu ne často stávají technologickým know-how), a vytvářet a ladit rozsáhlejší simulační model pouze v prostředí standardního programovacího jazyka (Java, C, C#...) je zdlouhavé.

Jestliže pro tvorbu simulačních modelů a pro vytváření vlastního simulátoru používáme odlišné vývojové nástroje, pak musíme ***zajistit dostatečně flexibilní přenos výsledků z jednoho vývojového prostředí do druhého.*** Tak např. modifikujeme-li simulační model v některém nástroji pro tvorbu simulačních modelů, je výhodné zajistit, aby se změny v modelu bez větších potíží mohly rychle promítnout do aktualizace těchto změn ve vlastním simulátoru. U jednodušších modelů je možné změny ručně přeprogramovat. U složitějších modelů je však pracnost a pravděpodobnost zavlečení chyby příliš vysoká.

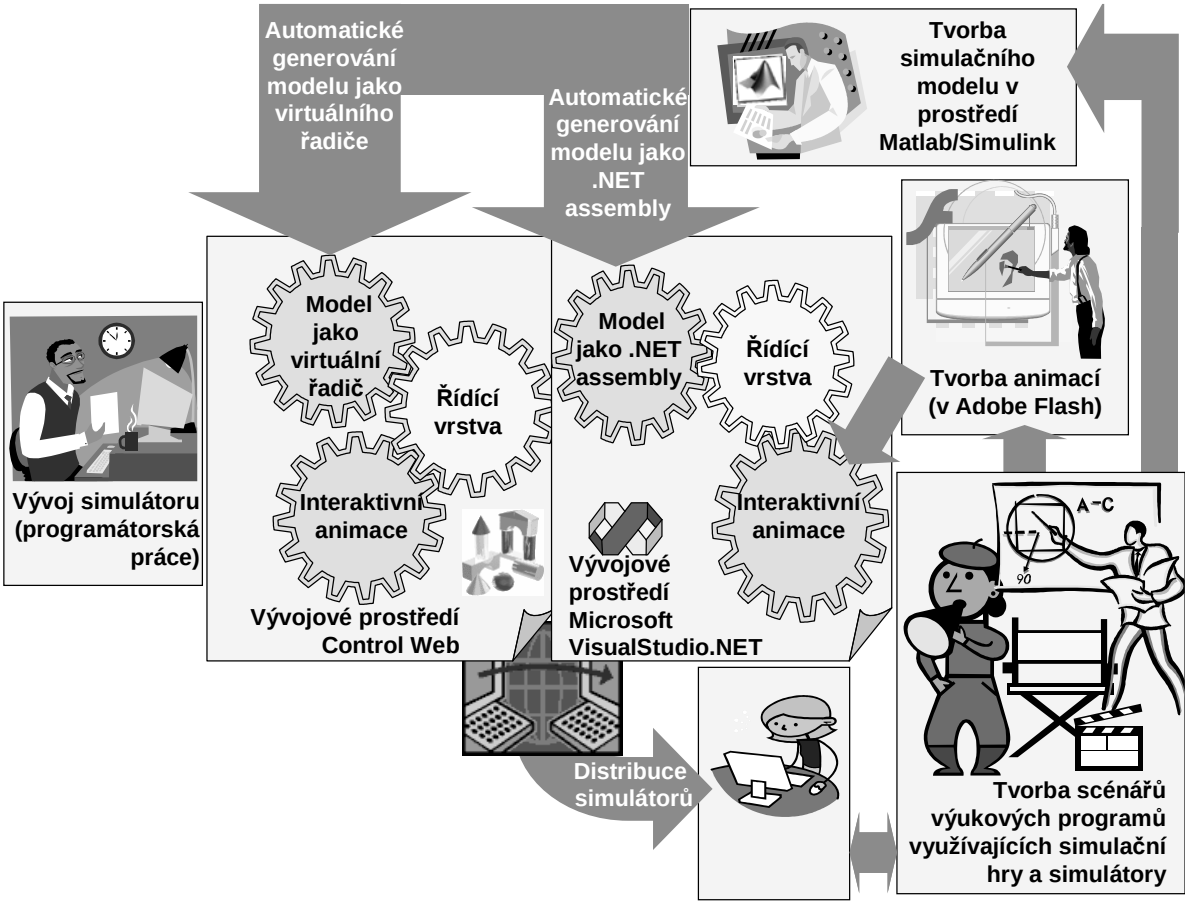
K usnadnění přenosu z jednoho vývojového prostředí do druhého je proto vhodné vytvořit si ***softwarové pomůcky***.

V naší laboratoři (obr. 2) jsme při vytváření a verifikaci ***simulačních modelů*** sáhli po profesionálních nástrojích firmy ***Mathworks (Matlab, Simulink a příslušné aplikační knihovny)*** a v poslední době i po velmi nadějném simulačním jazyku ***Modelica***, pro něž dotváříme i některé vlastní komponenty vývojového prostředí.

Při vytváření ***vlastního simulátoru*** využíváme standardní vývojová prostředí pro ***tvorbu a vývoj interaktivní grafiky*** (např. ***Adobe Flash***), nástroje pro tvorbu softwarových a webových komponent ***Visual Studio .NET*** a prostředí pro návrh a programování interaktivní grafiky v jazyce ***ActionScript***, jako je např. Adobe Flex. V minulosti jsme výukové simulátory vyvíjeli také pomocí nástroje pro vytváření průmyslových aplikací ***Control Web***.

Pro ***propojení*** obou skupin vývojových nástrojů jsme vyvinuli ***vlastní softwarové nástroje pro automatizaci přenosu simulačních***

[4] viz http://physiome.org/jsim/



Obr. 2 *Kreativní propojení nástrojů a aplikací pro tvorbu simulátorů a výukových programů využívajících simulační hry.*

modelů, vytvořených v prostředí Matlab/Simulink do vývojových prostředí Microsoft Visual Studio .NET a do Control Web [10].

3. Nové trendy a nové výzvy

Zdá se, že pomalu končí doba, kdy vytváření výukových programů bylo otázkou entuziasmu a píle skupin nadšenců. Tvorba moderních biomedicínských výukových aplikací je náročný a komplikovaný projekt, vyžadující ***týmovou spolupráci*** řady profesí – zkušených pedagogů vytvářejících základní scénář, tvůrců simulačních modelů, lékařů, výtvarníků a programátorů. 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 a začlenění do výukových multimediálních programů, dostupných prostřednictvím internetu.

Zároveň se objevují technologie, které propojení simulačních nástrojů, programového prostředí, multimédií a internetu podstatně usnadňují. Byl také učiněn velký pokrok ve vývojových softwarových prostředích, které dnes dávají malému týmu programátorů takové možnosti, které dříve měly pouze větší softwarové firmy.

To nás vedlo k určitému přehodnocení našeho dosavadního technologického postupu při tvorbě výukových simulátorů (popsaného na obr. 2).

4. Modelica versus Simulink – dřinu lidem versus dřinu strojům

Dlouho jsme pro vývoj simulačních modelů používali výlučně nástroje Matlab/Simulink od firmy Mathworks. Tyto nástroje dnes patří mezi osvědčené průmyslové standardy.

V poslední době se nicméně pro modelování systémů stále více začíná používat nový objektově orientovaný programovací jazyk **Modelica**. Byl původně vyvinut ve Švédsku a nyní je dostupný jak ve verzi open-source (vyvíjené pod záštitou mezinárodní organizace Modelica Association, http://www.modelica.org/), tak i ve dvou komerčních implementacích – od firmy Dynasim AB (prodává se pod názvem Dymola) a od firmy MathCore (prodává se pod názvem MathModelica).

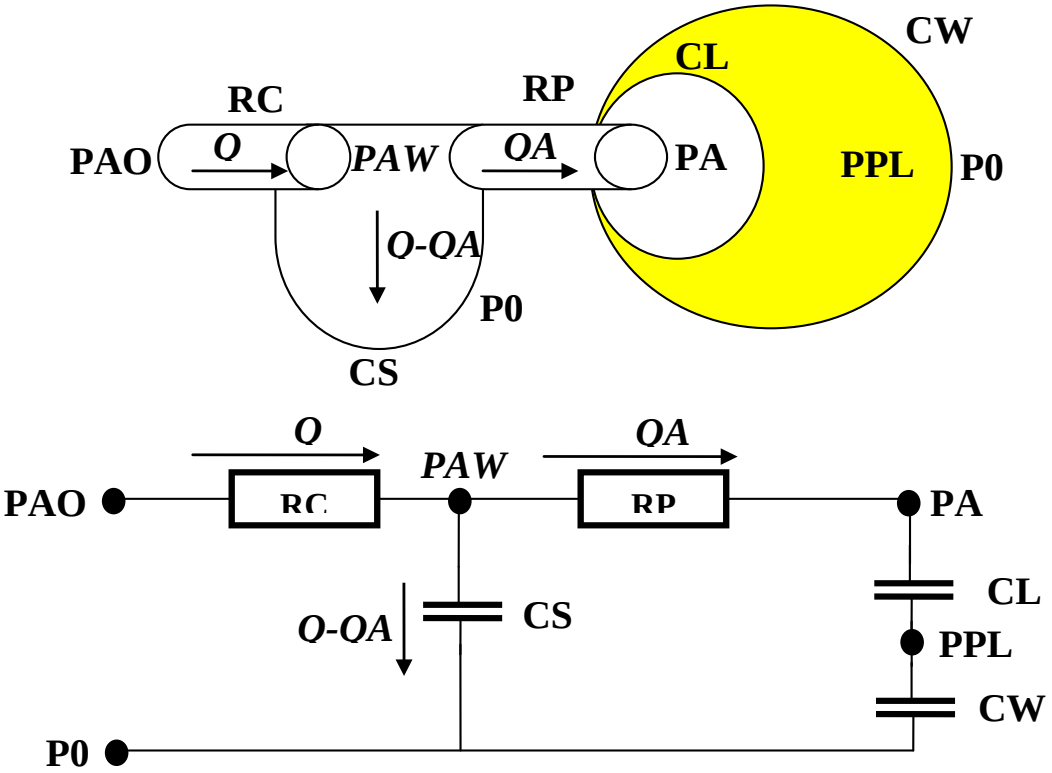
Zásadní inovaci, kterou na rozdíl od námi dosud používaného Simulinku Modelica přináší je ***deklarativní (a tedy akauzální)*** zápis modelů, kdy ***jednotlivé části modelu popisujeme přímo jako rovnice a nikoli jako algoritmus řešení těchto rovnic***.

Rozdíl mezi modelováním v Simulinku a v Modelice si ilustrujme na malém příkladu.

Uvažujme ***jednoduchý model mechaniky plic***, který schematicky zobrazuje obr. 3. Plíce si ve velkém zjednodušení představíme jako tři vaky propojené dvěma trubicemi. Plíce jsou připojeny k ventilátoru umělé plicní ventilace, který periodicky vhání tlakem **PAO** vzduch do plic. **P0** je tlak okolní atmosféry. Proud vzduchu ***Q*** proudí skrze horní cesty dýchací, jejichž odpor je ***RC***. Z horních cest dýchacích se vzduch prodírá dolními dýchacími cestami do alveolů. Odpor dolních dýchacích cest je ***RP***, tlak v centrálních partiích dýchacích cest (na rozhraní horních a dolních dýchacích cest) je ***PAW***, tlak v alveolech je ***PA***.

Vzduch roztahuje plicní alveoly, jejichž poddajnost je ***CL*** (jako celková poddajnost plic). Mezi plícemi a hrudním košem je interpleurální dutina. Tlak v ní je ***PPL***. Při umělé plicní ventilaci, kdy se pod tlakem

vhání vzduch do plic, se ještě musí roztáhnout hrudník – poddajnost hrudníku je **CW**. Malá část vzduchu, která se nedostane až do alveolů, pouze roztahuje dýchací cesty – jejich poddajnost je ***CS*** (prodýchávání tzv. mrtvého prostoru).



Obr. 3 *Jednoduchý model plicní mechaniky (hydraulická a elektrická analogie)*

Nyní si můžeme sestavit rovnice. Podle Ohmova zákona musí platit:

$$\begin{aligned} PAW - PA &= RP\ QA \\ PAO - PAW &= RC\ Q \end{aligned} \qquad (1)$$

Vztah mezi poddajností, tlakovým gradientem a objemem (vyjádřeným jako integrál průtoku) vyjadřují rovnice:

$$\begin{aligned} PA - PPL &= \frac{1}{CL}\int QA\ dt \\ PPL - P0 &= \frac{1}{CW}\int QA\ dt \\ PAW - P0 &= \frac{1}{CS}\int (Q - QA)\ dt \end{aligned} \qquad (2)$$

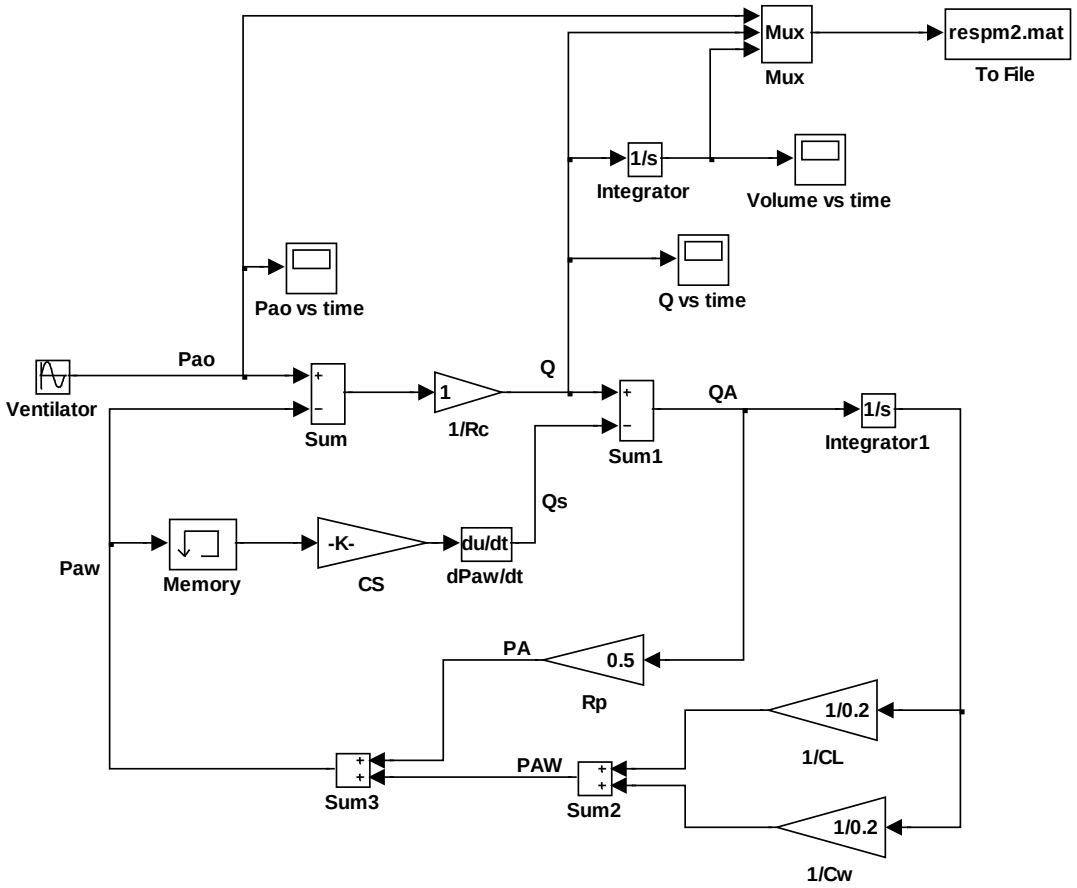
Podle zobecněného Kirchhoffova zákona musí být součet všech tlaků (napětí) podél uzavřené smyčky rovný nule, tj. ve smyčce podél uzlu PAW a podél uzlu PA0 musí platit:

$$(PAW - PA) + (PA - PPL) + (PPL - P0) + (P0 - PAW) = 0$$
$$(PAO - PAW) + (PAW - P0) + (PO - PAO) = 0$$

Po dosazení z rovnic Ohmova zákona a poddajností dostaneme:

$$\begin{cases} RP\,QA + \left(\frac{1}{CL} + \frac{1}{CW}\right)\int QA\,dt - \frac{1}{CS}\int (Q - QA)\,dt = 0 \\ Q\,RC + \frac{1}{CS}\int (Q - QA)\,dt + (P0 - PAO) = 0 \end{cases} \tag{3}$$

4.1 Modelování v Simulinku



Obr. 4 *Implementace modelu v Simulinku podle rovnic (3).*

Stavíme-li model v Simulinku, musíme přesně určit postup výpočtu ze vstupních proměnných na výstupní. Chceme-li počítat reakci toku vzduchu do/z plic (***Q***) na vstup – tj. na změny tlaku na začátku dýchacích cest (***PAO***) způsobované aparátem umělé plicní ventilace – bude simulinkový model vypadat jako na obr. 4.

Simulinkový model můžeme vyjádřit i jednodušeji. Nejprve z rovnice (3) dostaneme diferenciální rovnici (vstupní proměnná ***PAO***, výstupní ***Q***):

$$\frac{d^2PAO}{dt^2}+\frac{1}{RP\ CT}\frac{dPAO}{dt}=RC\frac{d^2Q}{dt^2}+\left(\frac{1}{CS}+\frac{RC}{RP\,CT}\right)\frac{dQ}{dt}+\frac{1}{RP\ CS}\left(\frac{1}{CL}+\frac{1}{CW}\right)Q \quad (4)$$

Při zadání číselných parametrů odporů (v jednotkách cm H_2O/L/sec) a poddajností (v jednotkách L/cmH_2O) [5]:

$$RC=1;\ RP=0{,}5;\ \ CL=0{,}2;\ \ CW=0{,}2;\ \ CS=0{,}005$$

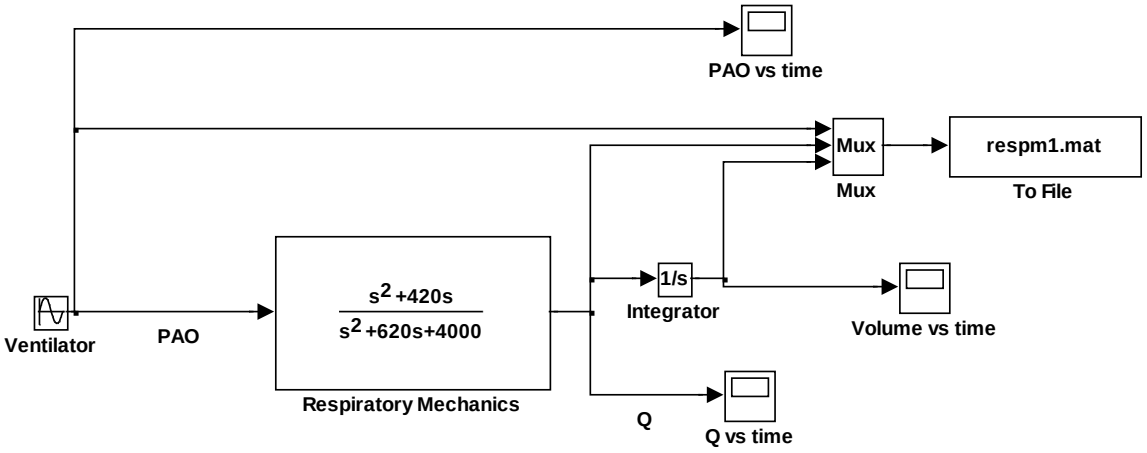
se rovnice (4) zjednoduší:

$$\frac{d^2PAO}{dt^2}+420\frac{dPAO}{dt}=\frac{d^2Q}{dt^2}+620\frac{dQ}{dt}+4000Q \quad (5)$$

V Laplaceově transformaci rovnice (5) dostaneme:

$$\frac{Q(s)}{PAO(s)}=\frac{s^2+420s}{s^2+620s+4000} \quad (6)$$

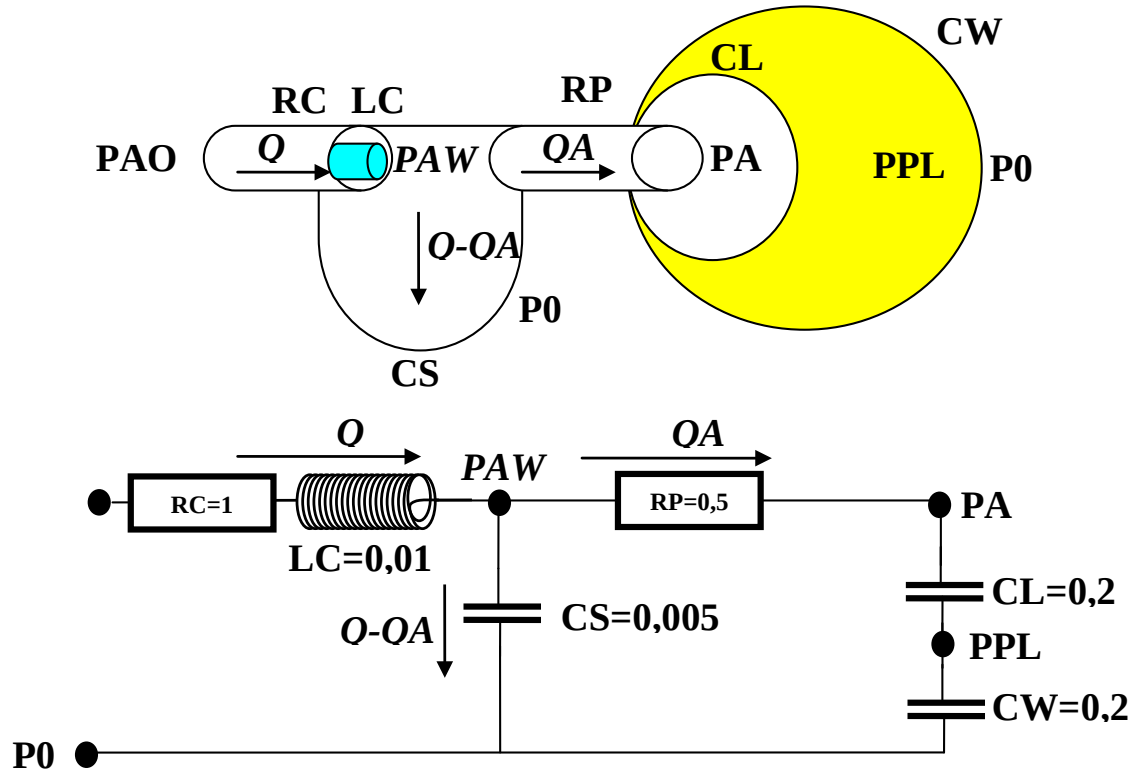
To umožní simulinkový model zjednodušit (obr. 5):



Obr. 5 *Implementace modelu v Simulinku s využitím Laplaceovy transformace podle rovnice (6).*

Při změnách hodnot parametrů ale se musí transformační funkce (6) přepočítat a simulinkový model se změní.

Nyní model mírně zesložitíme tak, že budeme uvažovat ***inerci vzduchu*** v horních dýchacích cestách (obr. 6).



Obr. 6 *Jednoduchý model plicní mechaniky s uvažováním inerce (hydraulická a elektrická analogie)*

Nyní navíc uvažujeme inerční element **LC**=0,01 cm H_2O s^2 L^{-1}:

$$LC = \frac{\Delta P}{\frac{dQ}{dt}}$$

kde ΔP je tlakový gradient a $\frac{dQ}{dt}$ je zrychlení průtoku, neboli:

$$\Delta P = LC\frac{dQ}{dt} \tag{7}$$

Potom místo soustav rovnic (3) dostaneme:

$$\begin{cases} RP\,\dfrac{dQA}{dt} + \left(\dfrac{1}{CL} + \dfrac{1}{CW}\right)QA - \dfrac{1}{CS}(Q - QA) = 0 \\ RC\,\dfrac{dQ}{dt} + LC\,\dfrac{d^2Q}{dt^2} + \dfrac{1}{CS}(Q - QA) + \dfrac{dP0}{dt} - \dfrac{dPAO}{dt} = 0 \end{cases} \tag{8}$$

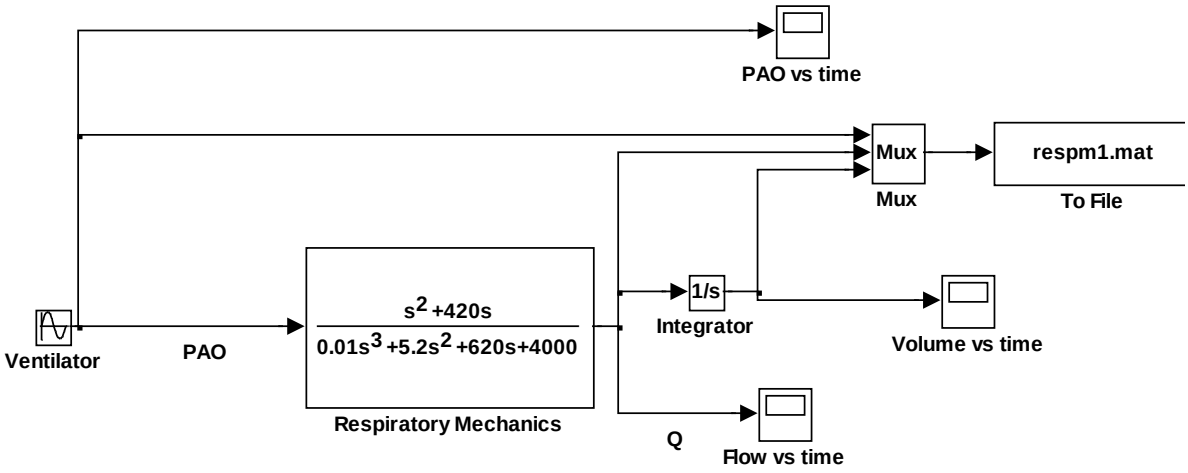
Místo rovnice (5) dostaneme:

$$\frac{d^2PAO}{dt^2} + 420\frac{dPAO}{dt} = 0{,}01\frac{d^3Q}{dt^3} + 5{,}2\frac{d^2Q}{dt^2} + 620\frac{dQ}{dt} + 4000Q \qquad (9)$$

a v Laplaceově transformaci dostaneme:

$$\frac{Q(s)}{PAO(s)} = \frac{s^2 + 420s}{0{,}01s^3 + 5{,}2s^2 + 620s + 4000} \qquad (10)$$

Simulinkový model se změní (obr. 7):



Obr. 7 *Implementace modelu v Simulinku s využitím Laplaceovy transformace podle rovnice (10).*

Díky tomu, že v Simulinku musíme vždy uvažovat směr výpočtu, je vlastní Simulinkové schéma dosti vzdáleno skutečné fyzikální realitě popisovaného systému. I malá změna v modelu, jako je přidání inerčního elementu, nutí k pečlivém propočtu a změně struktury modelu. K zásadní změně modelu dojde i tehdy, pokud bychom místo umělé plicní ventilace uvažovali spontánní dýchání. Vstupem modelu pak nebude tlak ***PAO*** vytvářený respirátorem umělé plicní ventilace, ale například poddajnost hrudní stěny ***CW*** (cyklickou změnou poddajnosti se dá modelovat funkce dechových svalů).

Simulink pracuje s propojenými bloky. V propojkách mezi jednotlivými bloky tečou signály, které přenášejí hodnoty jednotlivých proměnných od výstupu z jednoho bloku ke vstupům do dalších bloků. V blocích dochází ke zpracování vstupních informací na výstupní. ***Propojení bloků v Simulinku*** proto odráží spíše ***postup výpočtu*** a nikoli strukturu modelované reality.

4.1 Modelování v Modelice

V Modelice je to jinak. Místo bloků Modelica pracuje s propojenými komponenty, které představují instance jednotlivých tříd.

Na rozdíl od implementace tříd v jiných objektově orientovaných jazycích (jako jw C#, Java apod.), mají třídy v Modelice navíc zvláštní sekci, v níž se definují ***rovnice***.

Rovnice neznamenají přiřazení (tj. uložení výsledku výpočtu přiřazovaného příkazu do dané proměnné), ale definici vztahů mezi proměnnými (tak, jak je v matematice a fyzice zvykem).

Komponenty (instance tříd) se mohou v Modelice propojovat prostřednictvím přesně definovaných rozhraní – ***konektorů***. Konektory jsou instance konektorových tříd, v nichž se definují proměnné, používané pro propojení.

Propojovat se mohou konektory, patřící ke stejným konektorovým třídám (v nichž se tak mohou propojovat proměnné patřící k ekvivalentním typům). Jinak řečeno – v propojení typ zástrček musí přesně odpovídat typům zásuvek.

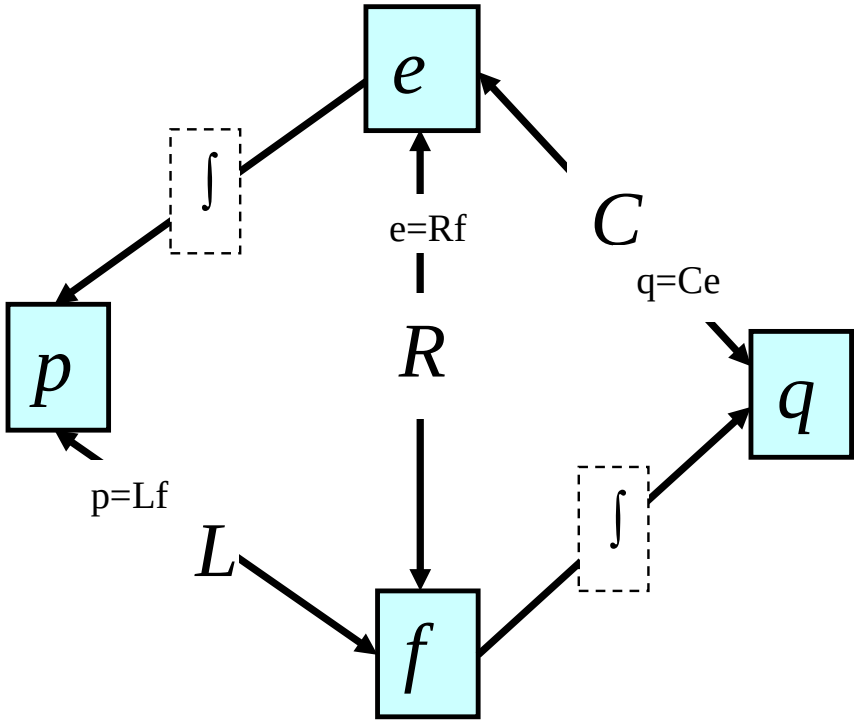
Důležité je to, že propojením komponent vlastně dochází k ***propojení soustav rovnic v jednotlivých komponentách*** mezi sebou. Je-li v konektoru propojených komponent definována stejná proměnná (a nereprezentuje-li tok – viz dále) propojením definujeme, že hodnota této proměnné bude ve všech komponentách stejná. Je-li tato proměnná součástí rovnic v propojených komponentách, propojením zároveň definujeme propojení rovnic, v nichž se tato proměnná vyskytuje.

V bodě, kde je propojeno více komponent, musí být hodnoty proměnných propojených přes společný bod stejné (analogicky jako napětí na společné svorce v elektrických obvodech musí odpovídat prvnímu Kirchoffovu zákonu).

Ale nejenom to.

V konektoru můžeme definovat, že některé propojované proměnné budou reprezentovat tok (flow) – pak to bude znamenat, že hodnoty všech takto označených proměnných budou ve všech komponentách propojených ve stejném bodě nastaveny na takovou hodnotu, aby jejich algebraický součet se rovnal nule (analogicky jako součet proudů na společné svorce v elektrických obvodech musí odpovídat druhému Kirchoffovu zákonu).

Je-li takto označená proměnná součástí rovnic v propojených komponentách, přidáváme tak do propojené soustavy rovnic další rovnici, definující požadavek nulovosti algebraického součtu hodnot této proměnné)



Obr. 8 *Vztahy mezi zobecněnými systémovými vlastnostmi*

- ***e*** *znamená zobecněné úsilí (effort) – v mechanice mu odpovídá síla, v elektrických schématech napětí, v hydraulice tlak atd.*
- ***f*** *je zobecněný tok (flow) – v mechanice mu odpovídá rychlost, v elektrických schématech proud, hydraulice průtok a v termodynamice teplotní tok apod.*
- ***q*** *je zobecněná akumulace,či výchylka, reprezentuje integrál zobecnělého toku. V mechanice jí např. odpovídá natažení pružiny, v hydraulice objem tekutiny, v elektrických schématech náboj, v termodynamice naakumulované teplo atd.*
- ***p*** *je zobecněná hybnost (inertance) - integrál zobecnělého úsilí, reprezentující kinetickou energii, v hydraulice představuje změnu rychlosti proudu úměrnou rozdílu tlaků (průtočnou hybnost), v elektrických obvodech potenciál nutný ke změně elektrického proudu (indukce) apod.*
- ***R***, ***C*** *a* ***L*** *představují konstanty úměrnosti mezi jednotlivými zobecněnými systémovými vlastnostmi. jednotlivými zobecnělými systémovými vlastnostmi. jednotlivými zobecnělými systémovými vlastnostmi. Odpovídá jim např. odpor, kapacitance či hmotnost.*

Propojením modelicových komponent tedy nedefinujeme postup výpočtu, ale modelovanou realitu. Způsob řešení rovnic pak "necháváme strojům".

Zobrazení modelu v Modelice tak více připomíná fyzikální realitu modelovaného světa než propojená bloková schémata v Simulinku.

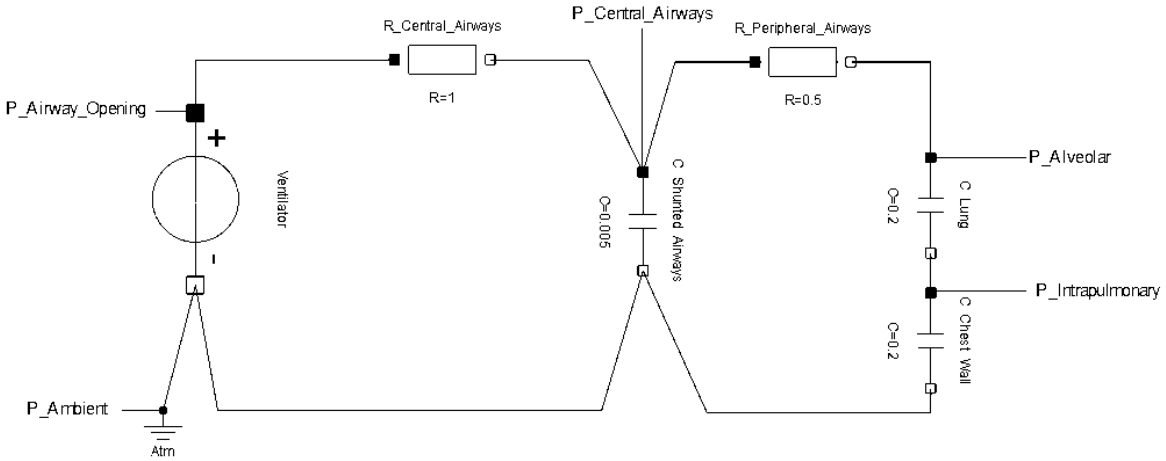
Souvisí to se zobecněnými systémovými vlastnostmi reálného světa (obr. 8) v nichž hrají důležitou roli zobecnělé úsilí (v reálném světě

mu odpovídá síla, tlak, napětí apod.) a zobecněný tok (jemuž v reálném světě mu odpovídá proud, průtok aj.).

Zobrazujeme-li v Modelice realitu propojenými komponenty, pak pro tokové proměnné musí hodnota v bodě propojení odpovídat druhému Kirchhoffovu zákonu (proud se nemůže v propojení akumulovat ani ztrácet) a pro ostatní proměnné musí ve společném propojovacím bodě platit rovnost (podle prvního Kirchhoffova zákona).

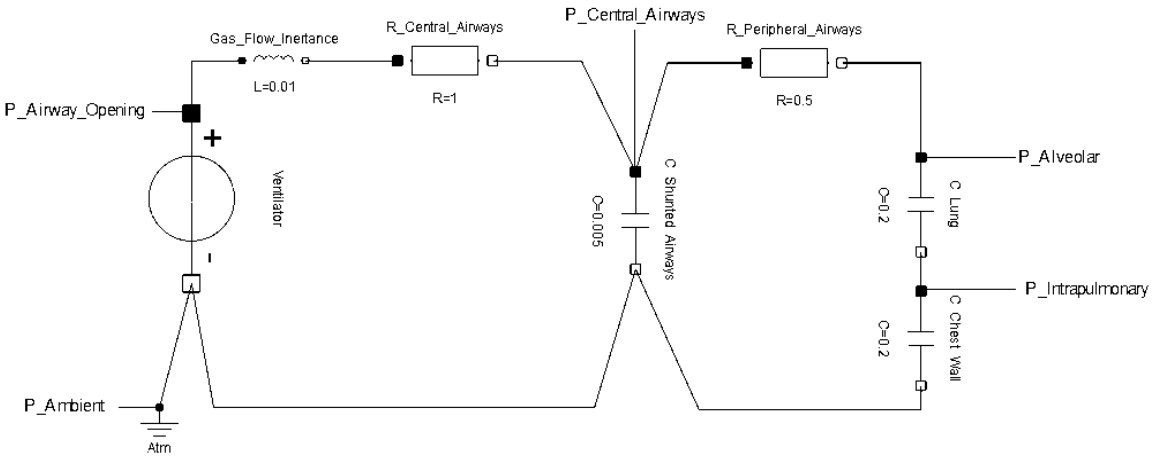
Modelica obsahuje knihovny nejrůznějších tříd pro modelování elektrických, mechanických a hydraulických objektů reálného světa.

Jednoduchý model mechaniky dýchání podle obrázku 3 můžeme v Modelice vyjádřit velmi přímočaře. Zobrazení vztahů odporu, tlakového gradientu a průtoku podle rovnice (1) a vztahů poddajnost, tlaku a průtoku podle rovnice (2) pak v Modelice vypadá takto:



Obr. 9 *Jednoduchý model plicní mechaniky podle obr. 3 v Modelice*

Zesložitění modelu přidáním inerčního elementu (dle rovnice 7 a obr. 6 je jednoduché:



Obr. 10 *Změněný model plicní mechaniky po přidání inerčního prvku*

V daném případě jsme pro rychlé sestavení modelu využily vizuální komponenty elektrických obvodů, nic ale nebrání vytvořit jiný tvar ikon, reprezentujících jednotlivé odporové, kapacitní a inerční elementy v plicích. Zdaleka nejde jen o obrazové ikony. Modelica je objektový jazyk a nic nebrání sestavit speciální třídy, a jejich pomocí např. modelovat toky kyslíku a oxidu uhličitého (s uvažováním vazby kyslíku na hemoglobin, přeměny CO2 na bikarbonát, vlivu acidobazické rovnováhy na přenos krevních plynů apod.).

Vytvoření knihovny fyziologických vztahů je jedním z našich budoucích cílů. Plánujeme převést (a dále v Modelice rozšířit) naši simulinkovou knihovnu fyziologických vztahů Physiology Blockset (www.physiome.cz/simchips).

Vzhledem k tomu, že ve fyziologických modelech se vyskytuje celá řada vztahů (které v Simulinku vedou na řešení implicitních rovnic) je ***akauzální popis modelovaných vztahů***, používaný v Modelice, velkou výhodou. Akauzální popis mnohem lépe vystihuje podstatu modelované reality a simulační modely jsou mnohem čitelnější, a tedy i méně náchylné k chybám. Pro modelování fyziologických systémů je proto Modelica velice vhodným prostředím.

5 WPF animace - loutky na nitích modelů v .NET

Pro vytváření uživatelského rozhraní výukového simulátoru je velmi vhodné simulátor navenek reprezentovat jako pohyblivý obrázek. Animované obrázky mohou být řízeny výstupy implementovaného simulačního modelu a graficky reprezentovat význam číselných hodnot. Např. schematický obrázek cévy se může roztahovat nebo komprimovat, plicní sklípek může hlouběji či mělčeji "dýchat", ručička měřícího přístroje se může pohybovat a průběžně zobrazovat hodnotu výstupní proměnné modelu čtené ze simulačního modelu běžícího na pozadí.

Na druhou stranu můžeme přes vizuální prvky (nejrůznější tlačítka, knoflíky, táhla apod.) do simulačního modelu zadávat vstupy.

Proto simulační model implementovaný v simulátoru propojujeme s multimediální animací prozatím vytvářenou pomocí Adobe Flash. Jednou z klíčových platforem pro tvorbu simulátorů je prostředí .NET. Do tohoto prostředí vkládáme Flashové animace jako ActiveX komponentu.

V nové platformě .NET Framework 3.0 je k dispozici nové grafické prostředí ***WPF (Windows Presentation Foundation)***. Při tvorbě uživatelského rozhraní programů nahrazuje Windows Forms a navíc umožňuje vytváření animací plně integrovaných s vlastním programem. Tím padá technologická bariéra mezi .NET a interaktivními animacemi, kterou jsme překlenovali v případě ActiveX komponent s Flashovými animacemi.

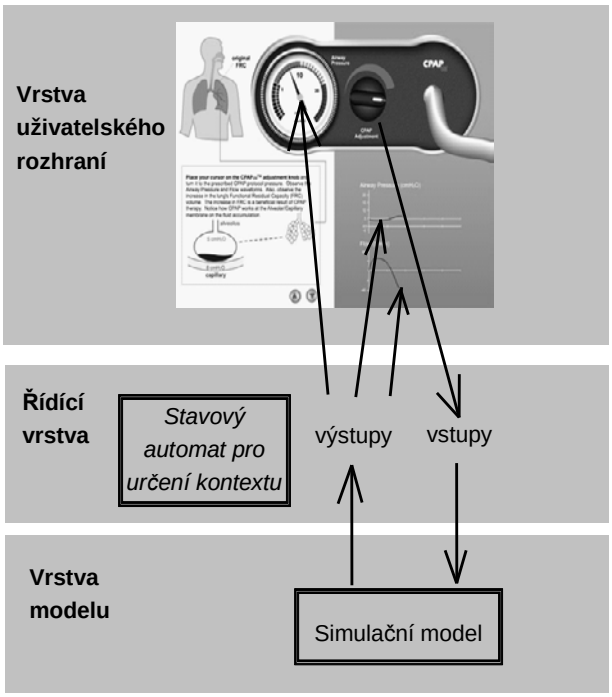
WPF umožňuje těsnější integraci a přesnější řízení animací a vizualizací než ActiveX. Jednou z klíčových vlastností je možnost spouštět aplikace přímo v okně webového prohlížeče.

Proto další technologickou inovací je postupný přechod od Flashových animací k animacím vytvářeným v prostředí WPF.

To samozřejmě neznamená, že se Flashových animací zcela vzdáváme (ještě dlouho bude řada výtvarníků umět kreslit animace v prostředí Flash a nikoli v prostředí WPF). Pouze si myslíme, že budoucnost (alespoň v prostředí .NET) bude mít WPF, na něž se chceme orientovat. Toto prostředí umí dokonale rozčlenit rozhraní určené pro výtvarníka a rozhraní určené pro programátora – tvůrce aplikace.

Logika propojení animace a simulačního modelu v simulátoru bývá poměrně složitá. Proto je vhodné mezi vrstvu vizuálních elementů a vrstvu simulačního modelu vložit řídící vrstvu, která na jednom místě řeší veškerou logiku komunikace uživatelského rozhraní s modelem. V této vrstvě se řeší veškerý kontext zobrazovaných dat a příslušné požadavky na komunikaci s modelem.

U složitějších simulátorů tato vrstva mimo jiné řeší nastavování parametrů simulačního modelu v závislosti na zvoleném scénáři (např. při modelování patogenezy různých patologických stavů nebo terapeutických zásahů, při rozpojování a opětovném spojování regulačních smyček během vysvětlování jednotlivých fyziologických regulací apod.). Důležité je, že veškerá logika zobrazování a komunikace je pak soustředěna do jednoho místa, což podstatně ušetří čas při modifikacích uživatelského rozhraní nebo změnách modelu.



***Obr. 11** MVC architektura při tvorbě simulátorů.*

V literatuře 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, která propojuje vrstvu simulačního modelu s uživatelským rozhraním, se nám velmi osvědčilo využít propojené stavové automaty (jejichž pomocí je možno zapamatovat si příslušný kontext modelu a kontext uživatelského rozhraní).

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

5 Náš cíl: Modelica .NET

Základní platformou, na níž se orientujeme při vytváření simulátorů, je ***platforma .NET*** a ***WPF.*** Prozatímní nevýhodou je to, že nejnovější verze tohoto frameworku je zatím k dispozici pouze pro Windows XP a vyšší. Takto to ale nebude natrvalo. Nastupující platforma MS Silverlight, založená na technologii WPF, přinese plug-in pro internetové prohlížeče a do značné míry zpřístupní tuto technologii i pro jiné operační systémy jako OS Mac či Linux.

Pro publikaci simulačních her využijeme schopnost .NET aplikací běžet přímo v okně webového prohlížeče. Výuková aplikace se transparentně nahraje do paměti počítače a spustí se v bezpečném prostředí, tzv. security sandboxu.

Pro bezešvou technologii vytváření výukových simulátorů potřebujeme ***propojit Modelicu a prostředí .NET***. Proto jsme se pustili do ambiciózního úkolu: implementovat Modelicu a vytvořit příslušné vývojové prostředí pro tvorbu modelů a simulátorů na platformě .NET. Toto prostředí by mělo umožňovat spolupráci více lidí na tvorbě modelu. Vývojové prostředí by mělo běžet v prohlížeči uživatele, který bude komunikovat se serverem na němž poběží kompilátor jazyka Modelica (OCM - Open Modelica Compiler).

Pro projekt ***OpenModelica*** jsme zatím vytvořili ***rozšíření pro konverzi modelicového modelu do jazyka C#,*** který je dostupný na adrese http://patf-biokyb.lf1.cuni.cz/wiki/modelica.net.

Tento nástroj umožňuje převádět model vytvořený v Modelice do podoby třídy napsané v jazyce C#. Tato třída obsahuje soustavu diferenciálních rovnic (reprezentujících model) převedenou do explicitního tvaru vhodného pro řešení pomoci ODE solveru.

Pro demonstraci jednotlivých kroků naší nové technologie jsme vytvořili jednoduchou výukovou aplikaci, jejíž vytváření popisujeme v samostatném příspěvku ve sborníku této konference [11].

6 Závěr

Zdá se, že pomalu končí doba, kdy vytváření výukových programů bylo otázkou entuziasmu a píle skupin nadšenců. Tvorba moderních biomedicínských výukových aplikací je náročný a komplikovaný projekt, vyžadující ***týmovou spolupráci*** řady profesí – zkušených pedagogů vytvářejících základní scénář, tvůrců simulačních modelů, lékařů, výtvarníků a programátorů. Aby tato interdisciplinární kolektivní tvorba byla efektivní, je nutno pro každou etapu tvorby využívat ***specifické vývojové nástroje***, s dostatečnou technickou podporou, které umožňují efektivní tvorbu simulačních modelů, programování výukových simulátorů, kreslení interaktivních multimédií a jejich závěrečné propojení všech komponent podle daného scénáře do kompaktního celku. K ovládnutí těchto nástrojů i vytvoření jejich technologického propojení je zapotřebí věnovat značné úsilí, které se ale nakonec vyplatí.

Nové technologie přinášejí pro tvorbu výukových simulátorů nové možnosti i nové výzvy. Jednou z nich je objektový jazyk ***Modelica***, který podle našeho názoru podstatným způsobem usnadní modelování tak složitých a komplexních systémů jakými jsou fyziologické systémy. Druhou výzvou jsou nové možnosti platformy ***.NET*** a ***WPF***.

Budoucnost (a naše další práce) ukáže, zda a jak se tyto nové možnosti uplatní.

7 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 Physiology for medical education.* Advannced Physiology Education, 31 (2), 202 - 210.

[2] Bassingthwaighte J. B. (2000). *Strategies for the Physiome Project.* Annals of Biomedical Engeneering 28, 1043-1058.

[3] Fritzon, P. (2004): Principles of object oriented modeling and simulation with modelica 2.1. IEE Press, A. John Willey & Sons Inc. Publication, New York, 2004, ISBN 0-471-47163-1

[4] Hunter P. J., Robins, P., & Noble D. (2002) The IUPS *Physiome Project*. Pflugers Archive-European Journal of Physiology, 445, s.1–9

[5] Kho M.C.K. (2000) *Physiological control systems*,.IEE Press, New York 2000, ISBN 0-7808-3408-6

[6] Kofránek, J. Anh Vu, L. D., Snášelová, H., Kerekeš, R., & Velan, T (2001): GOLEM – *Multimedia simulator for medical education.* In Patel, L., Rogers, R., Haux R. (Eds.). MEDINFO 2001, London: IOS Press, 1042-1046.

[7] Kofránek, J., Matoušek, S., Andrlík, M., Stodulka, P., Wünsch, Z., Privitzer, P., Hlaváček, J., Ondřej Vacek, O. (2007): *Atlas of physiology - internet simulation playground.* In. Proceedings of the 6th EUROSIM Congress on Modeling and Simulation, Vol. 2. Full Papers (CD). University of Ljubljana, ISBN 978-3-901608-32-2, MO-2-P7-5, 1-9. 2007.

[8] Miller, J. A., Nair, R. S., Zhang, Z., Zhao, H. (1997). JSIM: A JAVA-Based Simulation and Animation Environment, In *Proceedings of the 30th Annual Simulation Symposium*, *Atlanta, Georgia,* 31-42.

[9] Raymond, G. M., Butterworth E, Bassingthwaighte J. B. (2003). JSIM: Free Software Package for Teaching Physiological Modeling and Research. *Experimental Biology* 280,102-107

[10] Stodulka, P., Privitzer, P., Kofránek, J., Tribula, M., Vacek, O. (2007): *Development of web accessible medical educational simulators*. In. Proceedings of the 6th EUROSIM, Vol. 2. Full Papers (CD), MO-3-P4-2, str. 1-6, ISBN 978-3-901608-32-2, University of Ljubljana, 2007.

[11] Stodulka, P., Privitzer, P., Kofránek, J. (2008): *Jednoduchá simulační hra krok za krokem aneb od představy k hotovému*. Ibid.

Poděkování

Práce na vývoji lékařských simulátorů je podporována projektem Národního programu výzkumu č. 2C06031, rozvojovým projektem MŠMT C34/2008 a společností BAJT servis s.r.o.

Jiří Kofránek
Laboratoř biokybernetiky a počítačové podpory výuky, Ústav patologické fyziologie 1. LF UK, Praha
U nemocnice 5
128 53 Praha 2
tel: 777686868
fax:224912834
e-mail: kofranek@email.cz
http://www.physiome.cz