AKAUZÁLNÍ MODELOVÁNÍ – NOVÝ PŘÍSTUP PRO TVORBU SIMULAČNÍCH HER

Jiří Kofránek, Pavol Privitzer, Marek Mateják, Martin Tribula

Anotace

Modely vytvářené pomocí klasických simulinkových sítí přehledně graficky vyjadřují jednotlivé matematické vztahy. 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 pak odráží spíše postup výpočtu, než vlastní strukturu modelované reality. Hovoříme o tzv. kauzálním modelování. Při vytváření a hlavně při prezentování a popisu modelu je ale důležité, aby vlastní struktura modelu, spíše než vlastní algoritmus simulačního výpočtu, vystihovala především fyzikální podstatu modelované reality. Proto se v moderních simulačních prostředích začíná stále více uplatňovat deklarativní (akauzální) zápis modelů, kdy v jednotlivých komponentách modelu popisujeme přímo rovnice a nikoli algoritmus jejich řešení. Propojením jednotlivých komponent dochází k propojení soustav rovnic mezi sebou. Propojením komponent pak nedefinujeme postup výpočtu, ale modelovanou realitu. Způsob řešení rovnic pak „necháváme strojům". Moderním simulačním jazykem, který je přímo postaven na akauzálním zápisu modelů je Modelica. Pro modelování rozsáhlých a komplexních systémů je velmi vhodným prostředím.

Klíčová slova

Akauzální modelování, Kauzální modelování, Modelica, .NET

1. Úvod

Internetem zpřístupněné výukové simulační hry doplněné výkladem s multimediálním uživatelským rozhraním jsou novou perspektivní výukovou pomůckou, umožňující názorně ozřejmit vykládaný problém ve virtuální realitě, Jsou moderní realizací starého Komenského kréda „Schola Ludus" (škola hrou). Na našem pracovišti se léta zabýváme využitím interaktivních multimédií a simulačních her pro lékařskou výuku [7,8,10,13]. Některé výsledky naší práce jsou na CD ROM multimediální přílohou tohoto článku.

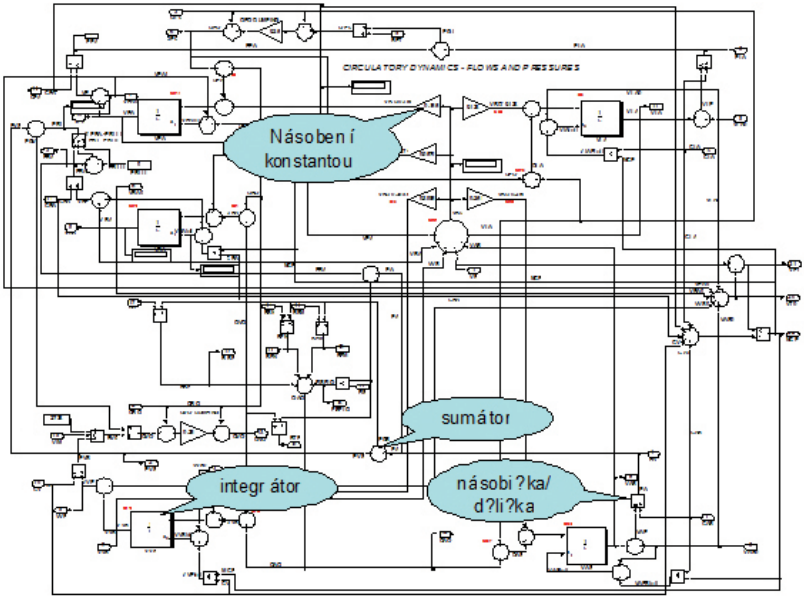
Tvorba výukových programů využívajících simulační hry není jednoduchá a vyžaduje vyřešit dva druhy problémů:

1. **Vytvoření simulačního modelu**
2. **Tvorba vlastního multimediálního simulátoru pro výukové simulační hry.**

Každý z těchto problémů má svou specifiku a její efektivní zvládnutí vyžaduje proto použít zcela odlišné vývojové nástroje.

Zatímco vytvoření vlastního simulátoru je spíše vývojářskou a programátorskou prací, tvorba simulačního modelu není vývojářský, ale spíše (poměrně náročný) výzkumný problém, jehož efektivní řešení vyžaduje použít adekvátní nástroje pro podporu tvorby simulačních modelů.

2. Softwarové nástroje pro tvorbu modelů: od Matlabu a Simulinku k Modelice.



Obr. 1 – Ukázka části modelu subsystému krevního oběhu v Simulinku. Simulinková síť propojuje jednotlivé výpočetní bloky a představuje spíše postup výpočtu než strukturu modelu.

Dlouhá léta jsme vyvíjeli a ladili matematické modely v standardních nástrojích od firmy Mathworks – v prostředí Matlab/Simulink, které dnes patří mezi osvědčené průmyslové standardy. V tomto prostředí jsme vytvořili speciální knihovnu formalizovaných fyziologických vztahů Physiology Blockset, volně dostupnou na našich webových stránkách

(www.physiome.cz/simchips). V Simulinku jsme implementovali matematický model, který byl podkladem pro simulátor Golem [7,8] v Simulinku jsme také implementovali rozsáhlé Guytonovy modely [4,5,9].

Simulink zpravidla 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. Simulink nabízí velkou sadu elementárních bloků (násobičky,děličky,integrátory atd.), realizujících přímo nějakou matematickou operaci, nebo i nějaký test, na jehož výsledku závisí řízení dalšího postupu výpočetního toku *(obr. 1)*. Propojováním těchto elementů se dají vytvářet počítací sítě realizující i poměrně komplikované algoritmy. Simulink umožňuje počítací sítě sdružovat do subsystémů, které navenek komunikují se svým okolím přes vstupní a výstupní porty a chovají se jako „simulační čipy". To umožňuje v Simulinku vytvářet přehledně hierarchicky strukturované modely, tvořené propojenými ***„simulačními čipy"***.

Propojování simulinkové sítě ale bohužel nemůže být zcela libovolné. V propojených prvcích se ***nesmějí vytvářet algebraické smyčky*** – tj. cyklické struktury, kdy (přes mnoho prostředníků) nějaká vstupní hodnota přiváděná do simulinkového elementu ve stejném časovém kroku závisí na výstupní hodnotě tohoto elementu [3]. Simulinková síť totiž ***netvoří grafické zobrazení matematických vztahů***, ale spíše grafické vyjádření řetězce transformací vstupních hodnot na výstupní přes jednotlivé simulinkové elementy, kde cyklení není dovoleno.

Pokud při stavbě modelu v Simulinku budeme myslet spíše na zobrazení struktury matematických vztahů, než na algoritmus výpočtů, snadno do modelu algebraické smyčky zaneseme (na což nás ovšem kompilátor upozorní). Existují metody, jak se algebraických smyček zbavit – vedou však k takovým transformacím, které strukturu modelu dále zesložití a model je méně přehledný. Požadavek pevně zadaného směru spojení od vstupů k výstupům s vyloučením algebraických smyček vede i k ***náročnější stavbě modelu***.

Propojení bloků v Simulinku proto odráží spíše postup výpočtu než vlastní strukturu modelované reality. Hovoříme o tzv. ***kauzálním modelování***.

V poslední době došlo k vývoji nových tzv. „akauzálních" nástrojů pro tvorbu simulačních modelů. Zásadní inovaci, kterou akauzální modelovací nástroje přinášejí je ***možnost popisovat jednotlivé části modelu přímo jako soustavu rovnic*** a nikoli jako algoritmus řešení těchto rovnic. Zápis modelů je deklarativní (popisujeme strukturu a matematické vztahy, nikoli algoritmus výpočtu) – zápis je tedy akauzální.

Akauzální modelovací nástroje pracují s propojenými komponentami, v nichž jsou definovány 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).

Tyto komponenty (které představují instance tříd s rovnicemi) se mohou ***propojovat*** prostřednictvím přesně ***definovaných rozhraní – konektorů***. Důležité je to, že propojením komponent vlastně ***dochází k propojení soustav rovnic v jednotlivých komponentách mezi sebou***.

Typickým představitelem akauzálních modelovacích nástrojů je právě nový objektově orientovaný programovací jazyk ***Modelica*** [6]. 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 - Dassault Systems pod názvem ***Dymola***, a od firmy MathCore pod názvem ***MathModelica***).

Výrobce simulačních nástrojů Matlab/Simulink – firma Mathworks reagovala na nové trendy vytvořením speciální ***akauzální simulinkové knihovny Simscape*** a návazných doménových knihoven SimElectronics, SimHydraulics, SimMechanics aj.

V souladu s moderními trendy jsme na našem pracovišti dosud užívané vývojové nástroje pro tvorbu matematických modelů (tj. Matlab/Simulink) rozšířili o nástroje využívající jazyk akauzálního modelování – jazyk Modelica. Okamžitě jsme ovšem zjistili značné výhody, oproti tvorbě modelů v prostředí Matlab/Simulink, které tento nový nástroj přináší, zejména při tvorbě rozsáhlejších modelů [11,12,13,14]. To nás vedlo k zásadnímu přehodnocení dosavadní strategie a jako základní nástroj pro tvorbu simulačních modelů jsme pro další vývoj simulátorů zvolili prostředí programovacího jazyka Modelica.

Ze tří výše zmíněných vývojových nástrojů pro modelování v Modelice je v současné době nejvyspělejším Dymola (od (od firmy Dynasim, viz: http://www.dynasim.se/index.htm). Proto jsme Dymolu zvolili jako referenční nástroj pro vytváření a ladění modelů.

3. Základní kameny akauzálního modelování

Při vytváření matematického popisu simulované reality je vhodné dodržovat určité konvence, které umožní dodržet přehlednost daného matematického zápisu. Každý zápis zkoumané reality pomocí soustav rovnic by měl mít přesně specifikované proměnné. Pokud je to možné, měl by mít i kromě svého významu také měřitelnou fyzikální jednotku. Modelica nabízí kontrolu fyzikálních jednotek. To je velmi výhodné, protože kontrola kompatibility jednotek nám umožní se vyhnout velmi špatně hledatelné chybě, kdy omylem v propojeních prohodíme konektory (pokud se zjistí,

že jednotky jsou inkompatibilní, kontrola nám vytvoření špatného propojení vůbec nedovolí).

Přehlednost modelu se zvýší, když jej rozdělíme na vhodné části, které mají ucelený význam, například proto, aby je bylo možné zkoumat samostatně v určitých podmínkách, nebo je i znovu použít (ať již na jiném místě stejného modelu nebo v jiném modelu). Proto často vytváříme znovupoužitelné knihovny „simulačních čipů".

V akauzálních jazycích mohou mít elementární prvky simulované reality nakonec velmi triviální zápis vztahů mezi danými veličinami. Příkladem je odpor, kondenzátor či cívka z elektrické fyzikální domény.

Složitý systém pro výpočet vznikne, když tyto elementární prvky začneme propojovat do sítí - při jejich vzájemném propojování vznikají soustavy rovnic. Jejich numerické řešením v kauzálních simulačních nástrojích nemusí být triviální – vzpomeňme na RCL modely cirkulace či respirace implementované v Simulinku.

V Modelice je to jednoduché. Stačí vytvořit instance těchto elementárních prvků a přes konektory je spojit. O algoritmus řešení vzniklé soustavy rovnic se pak postará samotný akauzální nástroj, a po spuštění simulace můžeme na různých místech simulovaného obvodu počítat proud a napětí [6].

Analogicky je možné vytvořit elementární prvky libovolné fyzikální domény s určitými fyzikálními vlastnostmi.

4. Akauzální konektory

Akauzální konektorové spojení těchto elementárních prvků se realizuje pomocí dvou typů veličin: jedné, jejíž hodnota zůstává na všech připojených uzlech stejná, a druhé, která představuje tok – pro něj platí, že součet hodnot toků je na všech připojených uzlech nulový (protože v oblasti rozvětvení do připojených uzlů se žádná látka neakumuluje).

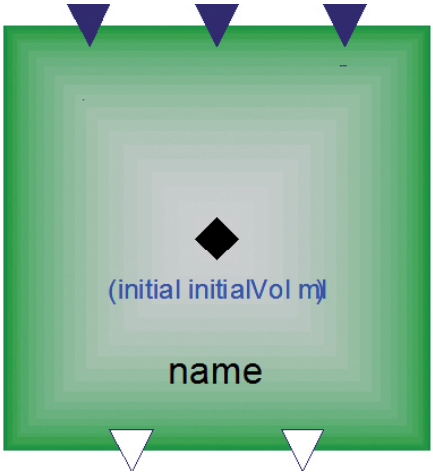
Prvku s daným typem spojení už stačí nadefinovat rovnice vztahů mezi veličinami jeho spojení. A ty se pak automaticky přidávají do soustavy rovnic při jeho začlenění do modelu.

Dnešní nástroje pro akauzální modelování jsou schopné generovat a numericky vyřešit velké soustavy rovnic, což umožňuje přímo při implementaci modelu zapisovat schémata fyzikálních, chemických nebo biologických dějů. A z těchto schémat je pak možné kliknutím myši přímo dostávat výsledky simulací.

5. Příklad prvku – elastický kompartment

Ukažme si jednoduchý příklad. Při modelování dynamiky cév často potřebujeme elastický (nafukovací) kompartment.

Nadefinujeme si proto třídu ***VascularElacticBloodCompartment*** jejíž instance budou elastické akauzálně propojitelné kompartmenty, které bude možno přes akauzální konektor připojit na „rozvod" tekutiny – tekutina může do/z kompartmentu proudit určitou rychlostí a pod určitým tlakem. V programovacím prostředí můžeme každé třídě, reprezentující model nebo konektor přiřadit grafickou ikonu. I pro náš vytvářený elastický kompartment můžeme vytvořit ikonu *(obr. 2)*.

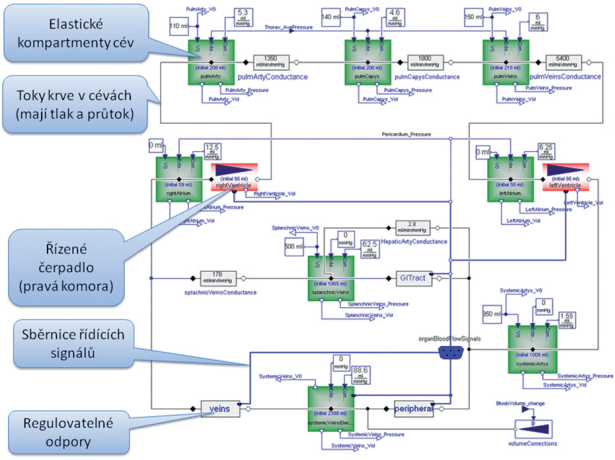


Obr. 2 – Modelica umožňuje vytvořit pro každou vytvářenou třídu, reprezentující model nebo konektor, vytvořit ikonu, která poslouží k tomu, aby instance třídy bylo možno pomocí grafických nástrojů propojovat s jinými instancemi. Výsledkem je struktura modelu z propojených instancí, velmi blízká modelované realitě. V daném případě jsme pro vytvoření ikony elastického kompartmentu vytvořili ikonu s jedním akauzálním konektorem (černý kosočtverec), třemi konektory pro signálové vstupy a dvěma konektory pro signálové výstupy. Každá instance elastického kompartmentu bude mít tuto ikonu s tím, že se v ikoně bude místo řetězce „initialVol" se bude zobrazovat skutečná hodnota počátečního objemu (zadaná jako parametr) a místo stringu „name" se bude zobrazovat název instance.

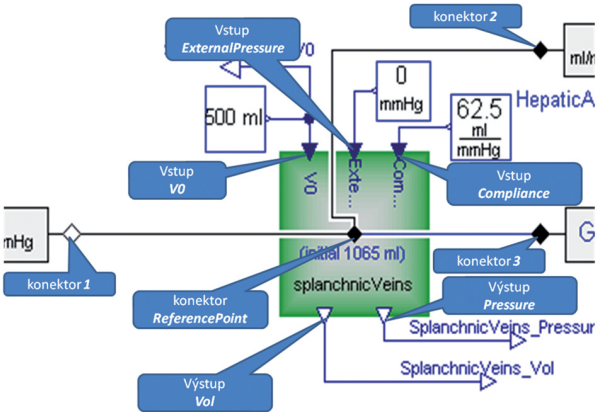
Není to jen čistě školní případ – tento kompartment uvažujeme v naší Modelicové implementaci rozsáhlého modelu fyziologických funkcí „Quantitative Human Physiology" [1,2]. Na *obr. 3* je zobrazen příklad využití instancí elastického kompartmentu v naší implementaci tohoto rozsáhlého modelu.

Elastický kompartment si můžeme představit jako nafukovací vak s jedním ***akauzálním propojovacím konektorem*** (nazveme jej třeba ***„ReferencePoint"***) přes který se budeme propojovat na okolí – tento konektor nám zprostředkuje dvě veličiny:

- tok ***„ReferencePoint.q"***
- tlak ***„ReferencePoint.pressure"***



Obr. 3 – Ukázka části modelu subsystému krevního oběhu („VascularCompartments") v Modelice (část Modelicové implementace rozsáhlého modelu Quantitative Human Physiology). Model je hierarchicky organizován jednotlivé bloky se dají „rozkliknout", představují instance tříd, v nichž jsou uvedeny rovnice. Modelicová síť se vyjadřuje spíše strukturu modelovaného systému, než způsob výpočtu.



Obr. 4 – Instance „splanchnicVeins" elastického kompartmentu „VascularElacticBloodCompartment". Akauzální spojení s příslušnými konektory na regulovatelných odporech (zde označených jako „konektor1", „konektor2" a „konektor3") propojí rovnice v instanci elastického kompartmentu „splanchnicVeins" do soustavy rovnic všech propojených elementů. Hodnota tlaku bude stejná na všech propojených konektorech: splanchnicVeins.ReferencePoint.pressure=konektor1.pressure=konektor2.pressure=konektor3.pressure. Algebraický součet všech toků na propojených konektorech musí být nulový: splanchnicVeins.ferencePoint.q+konektor1.q+konektor2.q+konektor3.q=0.

Pokud bude konektor zapojen přes konektor do okolí, pak hodnota tlaku bude na všech ke kompartmentu připojených uzlech skutečně stejná, a tok se bude rozdělovat do všech připojených uzlů tak, že jeho algebraický součet bude nulový (v oblasti rozvětvení se nebude vůbec nic akumulovat) – *viz obr. 4.*

Do kompartmentu budou vstupovat zvnějšku ***tři signálové (kauzální) vstupy:***

– základní náplň ***„V0"*** – hodnota objemu, po jehož dosažení bude v elastickém kompartmentu stoupat tlak. Pokud bude objem menší než nula, bude tlak v kompartmentu nulový.

– vnější, externí tlak ***„ExternalPressure"*** – tlak vnějšího okolí na elastický kompartment

– poddajnost elastického kompartmentu ***„Compliance"*** – té bude nepřímo úměrný tlak v kompartmentu pokud objem kompartmentu překročí základní náplň.

Z kompartmentu budou navenek vystupovat dva (kauzální) signálové výstupy:

– informace o momentálním objemu kompartmentu ***„Vol"***

– informace o hodnotě tlaku uvnitř kompartmentu ***„Pressure"***

Pro kompartment je výhodné ještě navrhnout parametr (jehož hodnota se načte před začátkem simulace), kterým by se určila jeho počáteční náplň:

– počáteční objem kompartmentu ***„initialVol"***

V programovém prostředí můžeme navrhnout i ikonku pro zobrazení elastického kompartmentu.

Vlastní fragment programu popisující chování elastického kompartmentu vypadá v Modelice následovně:

```
model VascularElacticBloodCompartment
 extends QHP.Library.Interfaces.BaseModel;

Real StressedVolume(inal quantity="Volume", final unit="ml");

parameter Real initialVol( final quantity="Volume", final
unit="ml")
 „initial compartment blood volume";

initial equation
 Vol = initialVol;

equation
 der(Vol) = referencePoint.q;
 StressedVolume = max(Vol-V0,0);
```

Pressure = (StressedVolume/Compliance) + ExternalPressure;

referencePoint.pressure = Pressure;

end VascularElacticBloodCompartment;

První dva řádky deklarují třídu modelu, dále se deklaruje reálná proměnná ***„StressedVolume"*** u níž budou kontrolovány i fyzikální jednotky. Dále se deklaruje parametr ***„InitialVol"*** u něhož také budou kontrolovány fyzikální jednotky. A pak následuje sekce rovnic. Nejprve se deklaruje inicializace počátečního objemu kompartmentu, tj. proměnné ***„Vol"***. Na dalších řádcích se v sekci **equation** deklarují čtyři rovnice. První je diferenciální rovnice – derivace objemu ***„der(Vol)"*** se rovná přítoku ***„q"*** z konektoru ***„referencePoint"***.

Další rovnice deklaruje, že hodnota elasticky napínaného objemu ***„StressedVolume"*** bude počítána jako rozdíl mezi objemem kompartmentu ***„Vol"*** a hodnotou jeho základní náplně ***„V0"*** (která je vstupem), a dákle rovnice říká, že hodnota objemu kompartmentu nikdy nemůže poklesnout k záporným hodnotám.

Třetí rovnice deklaruje vztah mezi tlakem v kompartmentu ***„Pressure"***, hodnotou napínaného objemu ***„StressedVolume"***, poddajností ***„Compliance"*** a externím tlakem ***„ExternalPressure"***. Ještě jednou připomínám, že se zde jedná o rovnice a nikoli o přiřazení. Rovnici by bylo možno v Modelice napsat i takto:

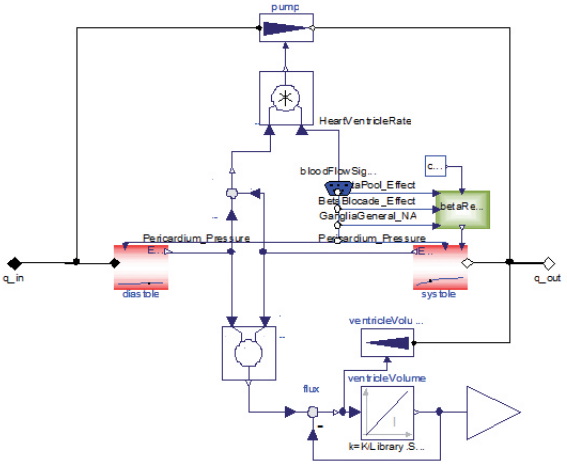
Pressure - ExternalPressure= (StressedVolume/Compliance);

Poslední rovnice propojuje hodnotu tlaku v kompartmentu ***„Pressure"*** s hodnotou tlaku propojovanou ***„referencePoint.pressure"*** akauzálním konektorem s okolím.

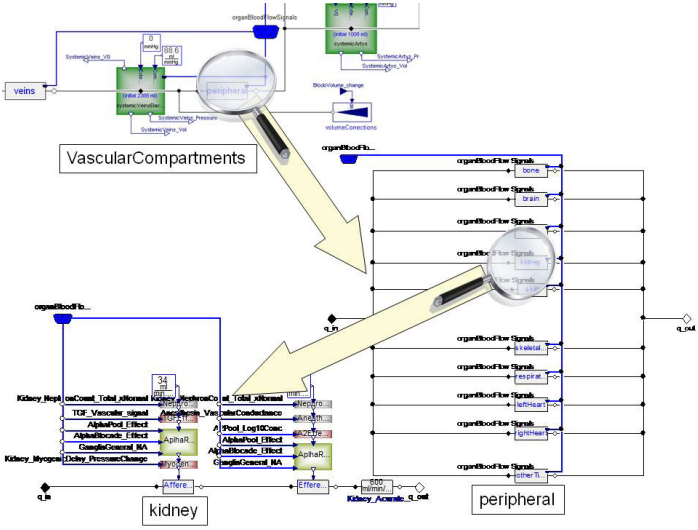
Hodnota „Pressure", je zároveň signálovým výstupem z kompartmentu – jako signál ji můžeme přivádět k dalším blokům – je to ale kauzální výstupní (signálová) proměnná a její hodnota nemůže být ovlivněna tím k čemu ji připojíme. U propojení z akauzálního konektoru je to ale jiné. Když instanci elastického kompartmentu propojíme akauzálním konektorem s dalšími prvky, pak čtveřice rovnic v kompartmentu se stane součástí soustavy rovnic daných příslušným propojením a hodnoty proměnných v instanci elastického kompartmentu budou záviset na řešení této soustavy rovnic.

6. Kombinace akauzálních a kauzálních signálových vazeb

V Modelice se hojně využívá kombinace akauzálních vazeb a kauzálních signálových vazeb. U akauzálních se nestaráme o směr výpočtu – postup je výsledkem řešení soustavy rovnic, která vznikne propojením akauzálních vazeb přes příslušné konektory. U kauzálních signálových vazeb je vždy nutno specifikovat co je vstup (propojovaný přes vstupní konektor) a co je výstup (propojovaný přes výstupní konektor vycházející z instance komponenty).



Obr. 5 – „Vnitřek" instance pumpy pravého srdce z obr. 3. Kombinace akauzálních a (kauzálních) signálových vazeb. Bohatost grafických možností pro zobrazení modelovaných vztahů umožňuje vytvářet hierarchicky členěné a „samodokumentující se" modely.

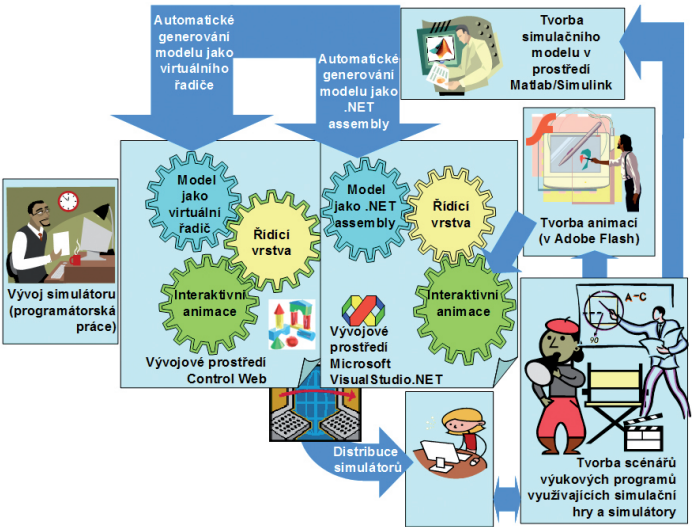


Obr. 6 – Hierarchické uspořádání modelů v Modelice. V komponentě „VascularCompartments" (z obr. 3) je jeden periferní řízený odpor s názvem „peripheral". Rozkliknutím se zobrazí řada paralelně zapojených řízených odporů. Rozkliknutím jednoho z nich – s názvem „kidney" se zobrazí složitě řízené odpory v ledvinách.

Příklad kombinace kauzálních a akauzálních vazeb jsou na *obr. 3* např. vstupní signálové vazby řídící hodnotu periferních odporů a čerpací funkce pravé a levé komory. *Obr. 5* zobrazuje „vnitřek" komponenty řídící čerpadlo pravé komory.

V Modelice je velmi důležité využívání hierarchičnosti a komponentové výstavby modelu (viz obr. 6). Pro architekturu stavby modelů je vhodné dodržovat pravidlo, aby se struktura komponenty vždy vešla na jednu obrazovku. Složitá spleť propojek nesvědčí o dobrém návrhu a vede ke zmatkům.

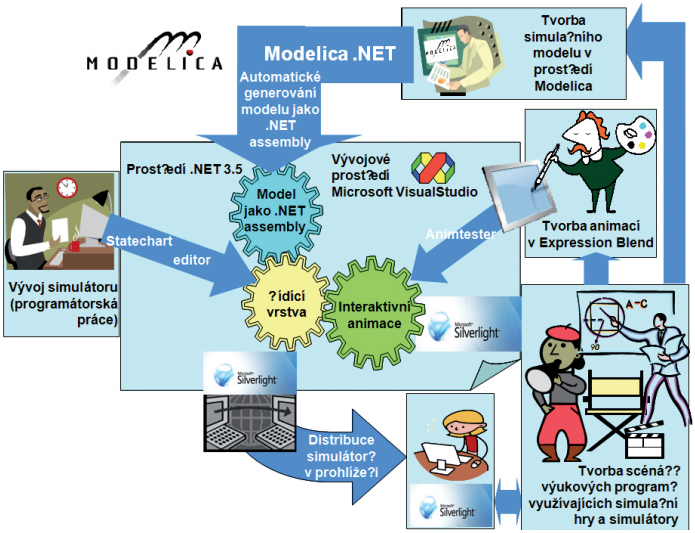
7. Nové softwarové nástroje pro tvorbu simulátorů – cíl: webově spustitelné simulátory



Obr. 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 animacev prostředí Adobe Flash. Jádrem simulátorů je simulační model, vytvářený v prostředí speciálních vývojových nástrojů, určených pro tvorbu simulačních modelů. My jsme zde dlouho 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.

Zásadní inovaci, kterou jsme provedli v prostředcích pro návrh simulačních modelů – odchod od platformy Matlab/Simulink směrem k Modelice (a také i naše účast v projektu Open Modelica, kde vytváříme nástroj pro

generování kódu v jazyce C#), spolu s novými možnostmi které přinášejí nové nástroje Microsoftu nás vedly i k přehodnocení dosud používané technologie pro tvorbu multimediálních simulátorů.



Obr. 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 animace v prostředí Expression Blend. Pro vytváření a testování animací, které budou posléze řízeny simulačním modelem, výtvarník využívá námi vyvinutý softwarový nástroj Animtester, který mu umožní bez programování vytvořit a ladit plynulé animace řízené změnami některých vlastností scény v čase. Jádrem simulátorů je simulační model, vytvářený v prostředí speciálních vývojových nástrojů, určených pro tvorbu simulačních modelů. My zde nyní využíváme velmi efektivní prostředí využívající simulační jazyk Modelica. V mezinárodní spolupráci pracujeme na generátoru kódu překladačem jazyka Modelica do podoby .NET komponenty, která spolu s řešičem diferenciálních rovnic implementovaným také na platformě .NET bude sloužit jako „datová vrstva" simulátoru s implementovaným modelem. Uživatelské prostředí je se simulačním modelem propojeno pomocí konceptu Data Binding, který zajišťuje inteligentní automatickou propagaci hodnot mezi těmito vrstvami, tedy přenos dat. Pro návrh vnitřní logiky aplikace používáme hierarchické stavové automaty (jejichž pomocí je možno zapamatovat příslušný kontext modelu a kontext uživatelského rozhraní). Vyvinuli jsme také vizuální prostředí (Statecharts editor) umožňující graficky automaty navrhnout, vygenerovat jejich kód a také je ladit. Výsledný simulátor je webová aplikace pro platformu Silverlight umožňující distribuovat simulátor jako webovou aplikaci, běžící přímo internetovém prohlížeči (i na počítačích s různými operačními systémy – stačí aby prohlížeč měl instalován příslušný plugin).

Naši původní technologii, kdy jsme byli nuceni pracovat se třemi typy rozdílných softwarových nástrojů *(obr. 7)* postupně přehodnocujeme a přecházíme na technologii výstavby simulátorů, jejímž základem je simulační jazyk Modelica, prostředí Sliverlight a platforma .NET *(obr. 8).*

V mezinárodní spolupráci vyytváříme ***back-end překladač jazyka Modelica***, zajišťujícího převod simulačního modelu ***do podoby*** managed .NET kódu a ***numerický řešič soustavy diferenciálních rovnic***. Spojili jsme úsilí v mezinárodní kooperaci s švédskou ***firmou Mathcore*** a ***Open Source Modelica Consortium***.

To umožní vygenerovat z modelu implementovaném v Modelice kód pro platformu .NET. Ve spojení s cílovou platformou ***Microsoft Silverlight*** nám to umožní vytvářet složité, ***numericky náročné, multimediální simulátory jako webové aplikace***.

8. Závěr

Nové technologie přinášejí pro tvorbu simulačních modelů nové možnosti i nové výzvy.

Jednou z nich je nový objektově orientovaný jazyk Modelica, který podle našeho názoru podstatně zjednoduší modelování tak složitých a komplexních systémů s jakými se setkáváme ve fyziologii. Vzhledem k tomu, že se ve fyziologických systémech vyskytuje řada vztahů, které v blokově orientovaných jazycích (např. v Simulinku) vedou na řešení implicitních rovnic, je akauzální popis používaný v Modelice velkou výhodou. Akauzální popis mnohem lépe vystihuje podstatu modelované reality a simulační modely jsou mnohem přehlednější a tudíž i méně náchylné k chybám.

Pro modelování fyziologických systémů je Modelica velmi vhodným prostředím.

Při tvorbě výukových programů v medicíně je úzkým místem nedostatek simulačních programů, na nichž by byly založeny výukové simulační hry. Modelica zde může být velmi užitečným nástrojem.

Poděkování

Práce na vývoji lékařských simulátorů je podporována projektem Národního programu výzkumu č. 2C06031, rozvojovým projektem MŠMT C20/2009 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 Physiology for medical education. *Advanced Physiology Education*, 31 (2), 202 – 210.

[2] Coleman T.G, Hester, R., Summers, R. (2008): *Quantitative Human Physiology* [Online] http://physiology.umc.edu/themodelingworkshop/
[3] Dabney J.B., Harman T.L. (2004) *Mastering Simulink,* Prentice Hall, Houston, 2004, ISBN: 0-13-142477-7
[4] Guyton AC, Coleman TA, and Grander HJ. (1972): *Circulation: Overall Regulation. Ann. Rev. Physiol., 41, s. 13-41.*
[5] *Guyton AC, Jones CE and Coleman TA. (1973): Circulatory Physiology: Cardiac Output and Its Regulation. Philadelphia: WB Saunders Company,1973.*
[6] Fritzson P. (2003): *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1,* ISBN 0-471-47163-1,Wiley-IEEE Press, 2003
[7] Kofránek J., Velan T., and Kerekeš R. (1997): Golem: a Computer Simulator of Physiological Functions as an Efficient Teaching Tool. Theo, Y. M., Wong, W. C., Okeu, T. J., and Rimane, R. 407-411. 1997. Singapore, IEE Singapore Section. Legacy for 21 Century. *Proceedings of the World Congress on System Simulation.*
[8] 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, Proceedings of the 10th World Congress on Medical Informatics*. London: IOS Press, 1042-1046.
[9] Kofránek J., Rusz J., Matoušek S. (2007): Guyton's Diagram Brought to Life - from Graphic Chart to Simulation Model for Teaching Physiology. In:Technical Computing Prague 2007. 15th Annual Conference Proceedings. Full paper CD-ROM proceedings. (P. Byron Ed.), Humusoft s.r.o. & Institute of Chemical Technology, Prague, ISBN 978-80-7080-658-6, str. 1-13, 2007. (V elektronické podobě dostupné na: *http://www.humusoft.cz/akce/matlab07/sbor07.htm#k*)
[10] Kofránek J., Mateják M., Matoušek S., Privitzer P., Tribula P., Vacek O. (2008e): School as a (multimedia simulation) play: use of multimedia applications in teaching of pathological physiology. In *MEFANET 2008.* (Daniel Schwarz, Ladislav Dušek, Stanislav Štípek, Vladimír Mihál Eds.), Masarykova Univerzita, Brno, 2008, ISBN 978-80-7392-065-4, CD ROM, str. 1-26, [Online] *http://www.mefanet.cz/res/file/articles/prispevek-mefanet-anglicky-kofranek.pdf*
[11] Kofránek J., Mateják M., Privitzer P. (2008d): Causal or acausal modeling: labour for humans or labour for machines. In Technical Conmputing Prague 2008, 16th Annual Conference Proceedings. (Cleve Moler, Aleš Procházka, Robert bartko, Martin Folin, Jan Houška, Petr Byron Eds). Humusoft s.r.o., Prague, 2008, ISBN 978-80-7080-692-0. CD ROM, str. 1-16, [Online] *http://www2.humusoft.cz/kofranek/058_Kofranek.pdf*

[12] Mateják M., Kofránek J. (2008): Modelica vs. blokovo-orientované jazyky matematického modelovania. In OBJEKTY 2008 (Žilina SR): Žilinská Univerzita, 20.-21.11.2008, (Jan Janech Ed.), Edis, Žilina, s. 79-94. ISBN 978-80-8070-923-3 (v elektronické podobě dostupné na: *http://patf-biokyb.lf1.cuni.cz/wiki/_media/_modelica_vs.pdf?id=-nase_publikace&cache=cache*).

[13] Rusz, J., Kofránek, J. (2008): Tools development for physiological educational simulators. In *Digital Technologies 2008* (Daša Ticha Ed.) [CD-ROM]. Žilina: University of Žilina, Fakulty of electrical engineering, 2008, vol. 1, ISBN 978-80-8070-953-2,(CD ROM), pp. 1-4 (v elektronické podobě dostupné na: *http://patf-biokyb.lf1.cuni.cz/ wiki/_media/rusz_kofranek_dt2008.pdf?id=nase_publikace&cache=cache*).

[14] Stodulka P., Privitzer P., Kofránek J. (2008): Jednoduchá simulační hra krok za krokem aneb Od představy k hotovému. In *MEDSOFT 2008.* (Milena Ziethamlová, Ed.) Praha: Agentura Action M, Praha 2008, s. 149-156. ISBN 978-80-86742-22-9 (V elektronické podobě dostupné na *http://patf-biokyb.lf1.cuni.cz/wiki/_media/jednoducha_simulacni_hra_krok_za_krokem.pdf?id=nase_publikace&cache=cache*).

Kontakt
Jiří Kofránek
Pavol Privitzer
Marek Mateják
Martin Tribula
Oddělení biokybernetiky a počítačové podpory výuky, ÚPF 1. LF UK, Praha
U nemocnice 5, 128 53 Praha 2
tel.: 777 686 868
e-mail: *kofranek@email.cz*
e-mail: *pavol.privitzer@lf1.cuni.cz*
e-mail: *matejak.marek@gmail.com*
e-mail: *mtrib@lf1.cuni.cz*
http://www.physiome.cz