DOI: 10.35191/medsoft_2021_1_33_25_34

STANDARDIZACE – CESTA K OPEN SOURCE TECHNOLOGIÍM PRO WEBOVÉ SIMULÁTORY

Jiří Kofránek, Tomáš Kulhánek

Abstrakt

Nové webové (HTML 5, WebAssembly, JavaScript, Web Components) a modelovací (FMI) standardy otevírají možnosti vytváření webových simulátorů propojujících simulační modely, grafiku, hypertext a multimédia, které lze spouštět na jakémkoliv zařízení s internetovým prohlížečem. Na těchto standardech je založena i naše technologie BodyLight.js.

1 Úvod

Tvorbě simulačních modelů a výukových simulátorů se věnuji řadu (více než čtyřicet) let. Tak, jak šel technologický pokrok, se nám postupně měnily pod rukama jak technologie tvorby simulačních modelů, tak i technologie tvorby vlastních simulátorů. Původně jsem simulační modely vytvářel v klasických programovacích jazycích - ve Fortranu, C++, Pascalu [1,2].

Dnes je situace jiná.

V současné době jsou pro vývoj, ladění a verifikaci simulačních modelů k dispozici specializovaná softwarová simulační prostředí, v nichž je možné vytvářet model v grafické podobě a poté i testovat jeho chování.

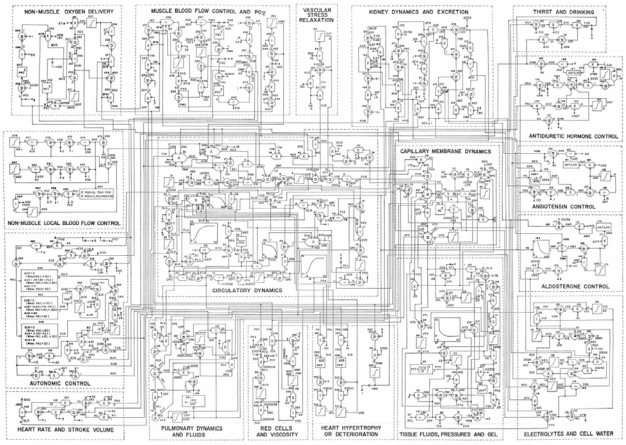
Již dávno pryč je doba, kdy vytváření výukových programů s uplatněním simulačních her bylo otázkou entuziasmu a píle skupin nadšenců. Tvorba moderních výukových aplikací je náročný a komplikovaný projekt, vyžadující týmovou spolupráci řady profesí – od zkušených učitelů, jejichž scénář je základem kvalitní 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 mezioborová spolupráce byla účinná, je zapotřebí pro každou etapu vývoje mít k dispozici řadu specifických vývojových nástrojů a metodologií, které práci jednotlivých členů týmu usnadní a pomohou jim překonat mezioborové bariéry. Propojením různých profesí a technologií se tvorba výukového softwaru stává efektivnější, pozvolna přestává být výsledkem kreativity a pracovitosti jedinců a stále více získává rysy inženýrské práce, kde se do popředí stále více dostává problematika standardů.

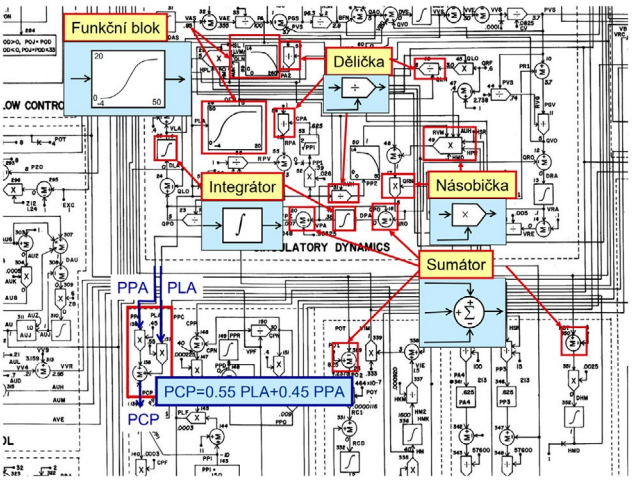
2 Od Fortranu k Modelice při tvorbě simulačních modelů

V roce 1972 v renomovaném odborném lékařském časopise Annual Review of Physiology byl publikován článek [3], který se svou podobou na již první pohled naprosto vymykal navyklé podobě fyziologických článků té doby. Jeho podstatnou část tvořilo rozsáhlé schéma na vlepené příloze. Schéma plné čar a propojených prvků na první pohled vzdáleně připomíná nákres nějakého elektronického zařízení (obr. 1). Avšak místo odporů, kondenzátorů, cívek, tranzistorů či jiných elektrotechnických součástek zde byly zobrazeny propojené výpočetní bloky (násobičky, děličky, sumátory, integrátory, funkční bloky), které symbolizovaly matematické operace prováděné s fyziologickými veličinami (obr. 2).

Svazky propojovacích vodičů mezi bloky na první pohled vyjadřovaly složité zpětnovazebné propojení fyziologických veličin. Bloky byly seskupeny do osmnácti skupin, které představovaly jednotlivé propojené fyziologické subsystémy. Centrálním byl subsystém reprezentující cirkulační dynamiku – s ním byly



Obrázek 1 - graficky vyjádřená struktura modelu Guytona a spol. z roku 1972



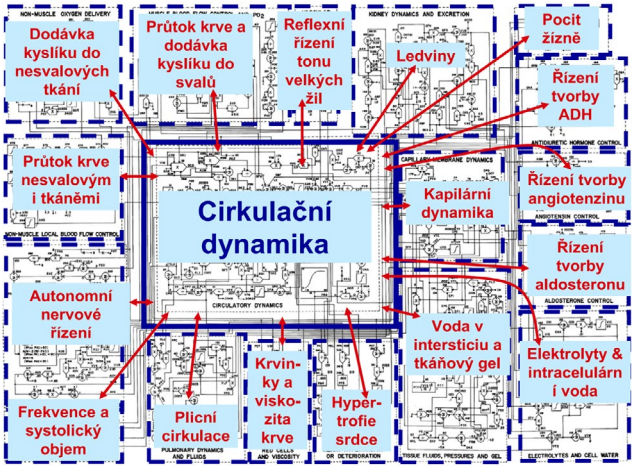
Obrázek 2 - Guytonův model byl tvořen kalkulačními prvky (děličkami, násobičkami, integrátory, funkčními bloky) propojenými spojnicemi, které reprezentovaly přenos hodnot jednotlivých proměnných mezi jednotlivými elementy. Dohromady pak celé grafické schéma reprezentovalo soustavu algebrodiferenciálních rovnic názorně popisujících matematický model regulace oběhu s návaznými subsystémy. V sedmdesátých a osmdesátých letech se Guytonova notace stala nepsanou normou, kterou užívali i další autoři.

do jednoho celku zpětnovazebně provázány ostatní bloky: od ledvin, přes tkáňové tekutiny, elektrolyty, až po autonomní nervovou regulaci a hormonální řízení zahrnující ADH, angiotenzin a aldosteron (obr. 3).

Autoři tímto tehdy naprosto novým způsobem pomocí graficky vyjádřených matematických symbolů popisovali fyziologické regulace cirkulačního systému a jeho širší fyziologické souvislosti a návaznost na ostatní subsystémy organismu – ledviny, regulaci objemové a elektrolytové rovnováhy aj. Místo vypisování soustavy matematických rovnic se v článku využívalo grafické znázornění matematických vztahů. Tato syntaxe umožnila graficky zobrazit souvislosti mezi jednotlivými fyziologickými veličinami ve formě propojených bloků reprezentujících matematické operace. Celé schéma tak představovalo formalizovaný popis fyziologických vztahů v oběhovém systému pomocí graficky vyjádřeného matematického modelu.

Guytonův model byl jedním z prvních rozsáhlých matematických popisů fyziologických funkcí propojených subsystémů organismu a odstartoval oblast fyziologického výzkumu, která je dnes někdy popisována jako „integrativní fyziologie" [4]. Obdobně jako se teoretická fyzika formálními prostředky snaží popsat fyzikální realitu a vysvětlit výsledky experimen-

DOI: 10.35191/medsoft_2021_1_33_25_34



Obrázek 3 – Gyutonův model byl jedním z prvních modelů tzv. integrativní fyziologie, která se zabývá integrativním pohledem na vzájemné regulační vztahy jednotlivých fyziologických systémů. Model se tedy zdaleka nezabývá jen cirkulačním systémem, ale všemi jeho vazbami na okolní subsystémy lidského organismu. Obdobně jako model letadla slouží podkladem pro letecké simulátory, tak i integrativní modely lidské fyziologie dnes slouží teoretickou bazí pro lékařské simulátory.

tálního výzkumu, tak se i „integrativní fyziologie“ na základě experimentálních výsledků snaží vytvořit formalizovaný popis vzájemného propojení fyziologických regulací a vysvětlit jejich funkci v rozvoji nejrůznějších onemocnění.

Z tohoto hlediska byl Guytonův model určitým mezníkem, který se snažil systémovým pohledem na fyziologické regulace zachytit dynamiku vztahů mezi regulací oběhu, ledvin, dýchání, objemu a iontového složení tělních tekutin pomocí graficky znázorněné sítě.

Guytonova grafická notace formalizovaného popisu fyziologických vztahů, inspirovaná tehdy hojně používanými analogovými počítači, představuje velmi přehledné vyjádření matematických souvztažností – bloky v uzlech sítě představují grafické symboly pro jednotlivé matematické operace a vodiče reprezentují jednotlivé proměnné.

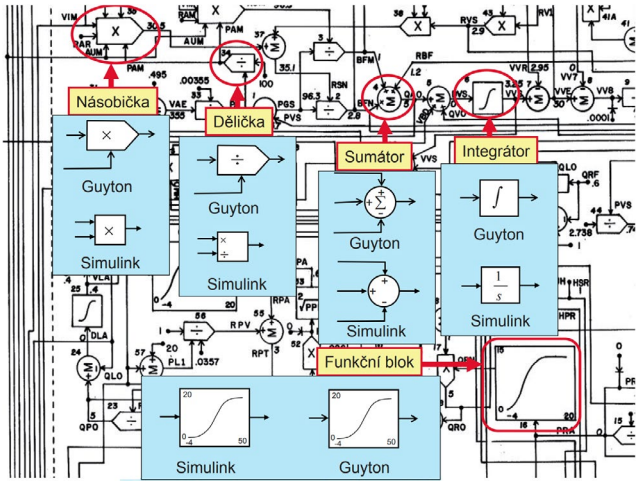
Guytonovu grafickou notaci záhy převzali i jiní autoři – např. Ikeda a spol. v Japonsku [5] nebo výzkumná skupina Amosova v Kijevě [6]. Spoluautor článku Thomas Coleman ji v roce 1983 použil ve svém rozsáhlém modelu lidské fyziologie nazvaném Human [7].

Grafický zápis matematického modelu prostřednictvím sítě propojených bloků byl ale v době svého vzniku pouhým obrazovým znázorněním – Guytonův model i jeho další modifikace (stejně jako i modely dalších autorů, kteří Guytonovu vyjadřovací notaci přejali) byly původně implementovány ve Fortranu a později v jazyce C++.

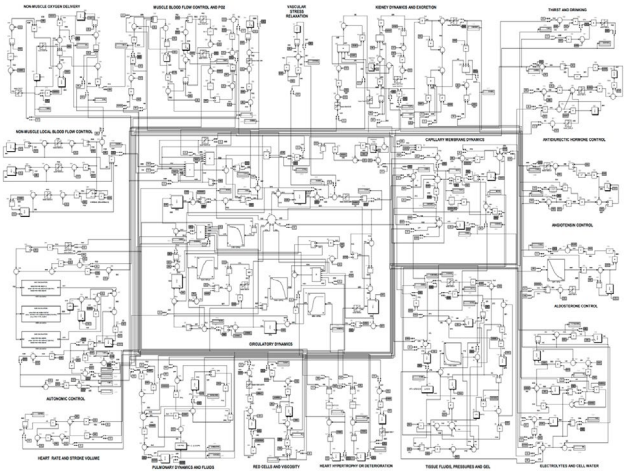
Počátkem devadesátých let se objevily blokově orientované jazyky, např. Simulink od firmy Mathworks – kde se model tvořil propojováním jednotlivých komponent pospojovaných pomocí počítačové myši do simulačních sítí. Výsledný model byl pak přehledně reprezentován sítí vzájemně propojených bloků. Simulační bloky v Simulinku velice připomínaly kalkulační prvky, které použil Guyton ve svém modelu integrativní fyziologie oběhu (obr. 4), a to nás inspirovalo i k tomu, že jsme guytonův model přepsali do podoby simulinkového modelu - se stejnou strukturou, jakou mělo i původní grafické schéma modelu (obr. 5), a se stejnými výsledky jaký dával původní model implementovaný ve Fortranu [8]. Nebylo to jednoduché, protože v původním schématu modelu byly čtyři drobné grafické překlepy (záměna znamének + a -, obrácené zapojení integrátoru aj.), které ale při znalosti fyziologie bylo možné snadno odhalit. Jiní autoři, [9] kteří se snažili převést Guytonův model přímo z Fortranu do Simulinku, vytvořili simulinkovou strukturu mnohem větší a s mnohem složitější strukturou, zcela odlišnou od původního schématu v Guytonově publikaci. Nebylo to ale tím, že by fortranský program se týkal jiného, rozsáhlejšího modelu, ale tím, že do Simulinku otrocky překlopili i numerické algorimty, kterými ve Fortranu pomáhaly řešit numeriku algebro diferenciálních “stiff” rovnic. Protože numerika výpočtu se musela potýkat s výpočtem rovnic vyjadřujících rychlé děje (např. hemodynamiku) a zároveň o několik řádů pomalejší děje (přesuny tekutin, hemopoeza, některé neurohumorální adaptace apod.), model ve Fortranu zahrnoval krom rovnic modelu také algoritmy, které pomáhaly složité numerické problémy řešit. V Simulinku se ale o numeriku stará nastavená volba numerických metod na pozadí a simulinková síť vyjadřuje skutečnou strukturu modelu.

Blokově orientované simulační jazyky, jejichž typickým představitelem je právě Simulink, umožňují sestavovat počítačové modely z jednotlivých bloků, s definovanými vstupy a výstupy. Bloky jsou seskupeny v knihovnách a pomocí počítačové myši se při tvorbě modelu vytvářejí jejich jednotlivé instance, jejichž vstupy a výstupy se propojují pomocí vodičů, kterými „proudí“ informace.

Simulinkovou síť je možné hierarchicky uspořádat. Bloky je možno seskupovat do jednotlivých subsystémů, které s jejich vnějším okolím komunikují prostřednictvím definovaných



Obrázek 4 – Počátkem devadesátých let firma Mathworks uvedla na trh platformu Simulink, která používala obdobnou notaci, jakou zavedl Guyton. Zásadní rozdíl byl ale v tom, že model v Simulinku již nebyl jen grafický obrázek, ale spustitelný matematický model.



Obrázek 5 - Guytonův model realizovaný v jazyce Simulink. Na rozdíl od obrázku 1, je tento model nejen grafické znázornění v matematických vztahů, ale spustitelný simulační model.

DOI: 10.35191/medsoft_2021_1_33_25_34

vstupních a výstupních „pinů" a představují tak jakési „simulační čipy". Simulační čip skrývá před uživatelem strukturu simulační sítě, obdobně jako elektronický čip ukrývá před uživatelem propojení jednotlivých tranzistorů a dalších elektronických prvků. Uživatel se pak může zajímat pouze o chování čipu a nemusí se starat o vnitřní strukturu a algoritmus výpočtu. Chování simulačního čipu pak může testovat pomocí sledování výstupů na připojených virtuálních displejích či na virtuálních osciloskopech.

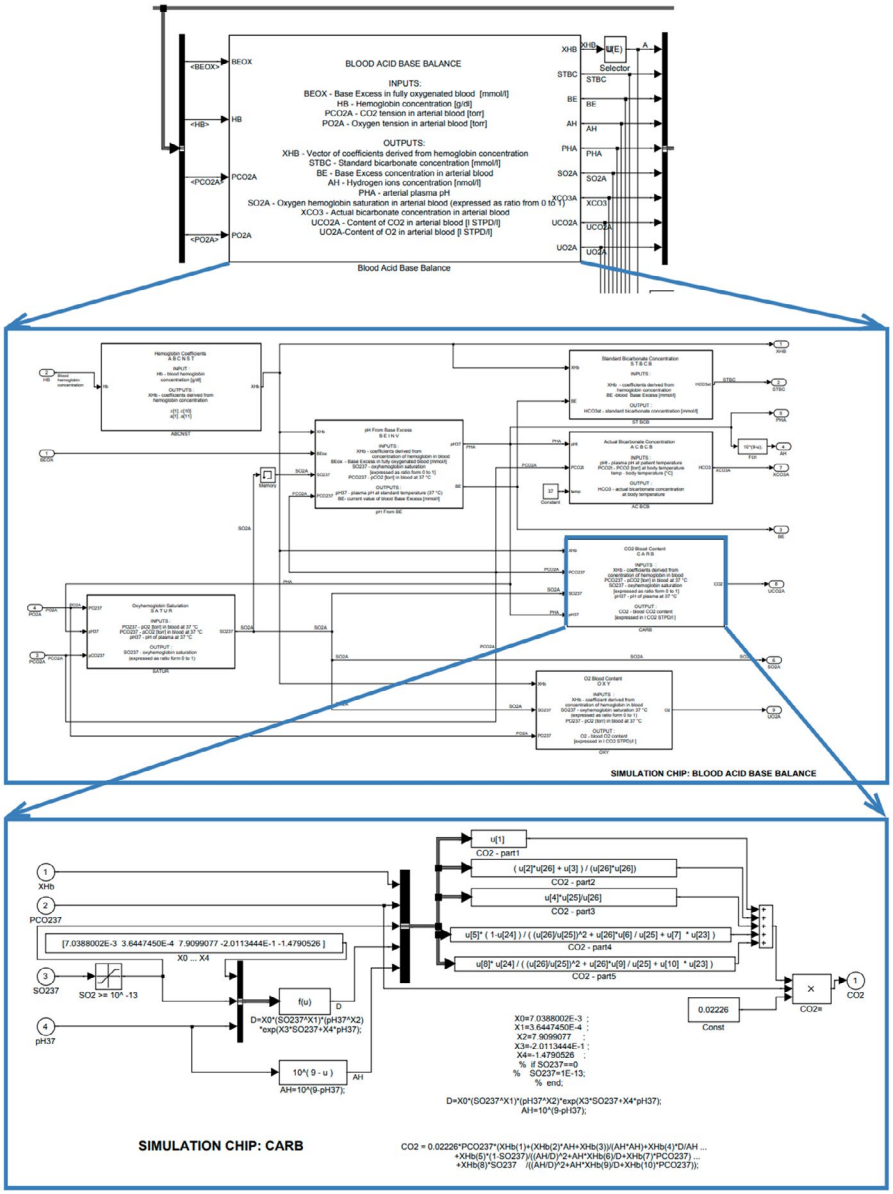
Pomocí simulačních čipů lze snadněji testovat chování modelu a zejména přehledněji vyjádřit vzájemné závislostí mezi proměnnými modelovaného systému. Celý složitý model pak můžeme zobrazit jako propojené simulační čipy a ze struktury jejich propojení je jasné, jaké vlivy a jakým způsobem se v modelu uvažují (obr. 6).

To je velmi výhodné pro mezioborovou spolupráci – zejména v hraničních oblastech jako je např. modelování biomedicínských systémů [10]. Experimentální fyziolog nemusí dopodrobna zkoumat, jaké matematické vztahy jsou ukryty „uvnitř" simulačního čipu, z propojení jednotlivých simulačních čipů mezi sebou však pochopí strukturu modelu a jeho chování si může ověřit v příslušném simulačním vizualizačním prostředí. V Simulinku jsme např. vytvořili rozsáhlý model, který byl podkladem pro simulátor Golem, určený k výuce klinické fyziologie poruch homeostázy vnitřního prostředí (obr. 7) a také i knihovnu pro modelování fyziologických regulací [11,12].

Simulink (a jiné obdobné blokově orientované simulační jazyky, kde se model vytvářel propojováním jednotlivých bloků) podstatně zjednodušily a urychlily naprogramování modelu – tj. jeho implementaci do počítače.

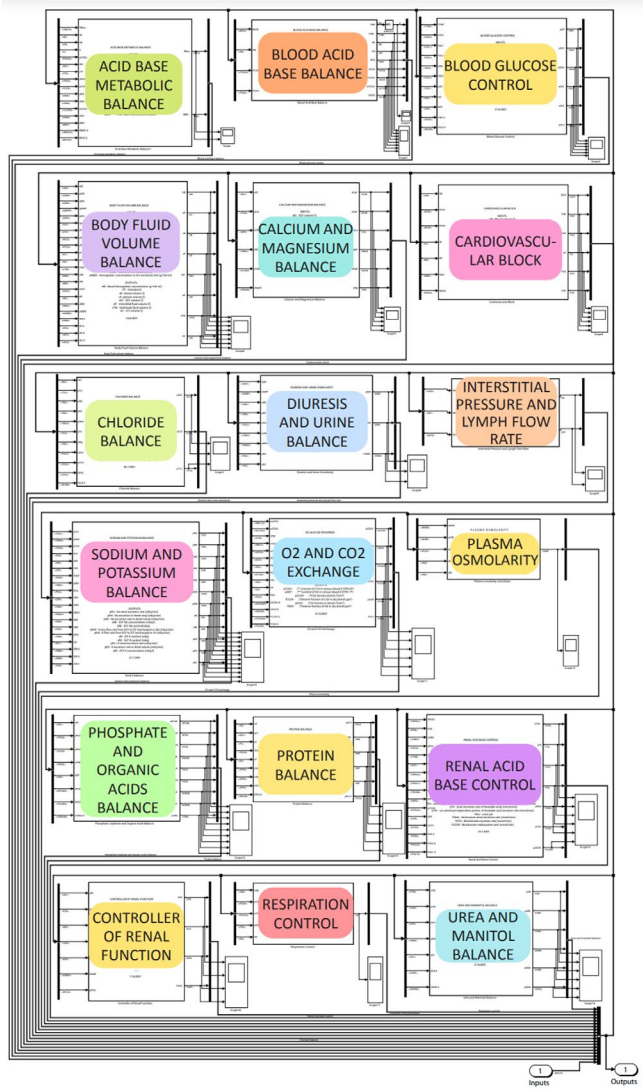
Propojení simulačních bloků muselo být takové, aby umožňovalo vypočítávat výstupní hodnoty modelu ze vstupních hodnot, což zvláště u složitých modelů není jednoduché. Simulinkové agregované bloky se do sebe mohou zanořovat a na nejnižší hierarchické úrovni je blok tvořen sítí numerických kalkulačních bloků, jejichž propojení vyjadřuje vlastní algoritmus výpočtu výstupních proměnných ze vstupních proměnných.

Propojování bloků do sítě vztahů ale bohužel nemůže být zcela libovolné. V propojených prvcích se nesmějí vytvářet algebraické smyčky – tj. cyklické struktury, kdy nějaká vstupní hod-



Obrázek 6 - Simulinkové kalkulační prvky se dají seskupovat do subsystémů s definovanými vstupy a výstupy. Tvoří tak jakési "simulační čipy". Subsystémy mohou být součástí subsystémů vyšší hierarchické úrovně. Tímto způsobem se dají tvořit složité hieararchicky uspořádané subsystémy realizované pomocí propojených "simulačních čipů". Na nejnižší hierarchické úrovni jsou ale vždy základní kalkulační prvky simulinku.

DOI: 10.35191/medsoft_2021_1_33_25_34



Obrázek 7 - Příklad modelu sestaveného ze simulinkových subsystémů ("simulačních čipů"). Tento model sloužil jako podklad pro sestavení našeho simulátoru "Golem".

nota přiváděná jako vstup do výpočetního bloku ve stejném časovém kroku závisí (přes několik prostředníků) na výstupní hodnotě z tohoto bloku.

Propojení bloků v Simulinku (a obdobných blokově orientovaných modelovacích jazycích) proto odráží spíše postup výpočtu než vlastní strukturu modelované reality. Hovoříme o tzv. **kauzálním modelování.**

U složitých systémů se díky tomuto přístupu pod strukturou výpočtu pomalu ztrácí fyzikální realita modelovaného systému.

Na přelomu tisíciletí došlo k vývoji nových tzv. **„akauzálních" nástrojů** pro tvorbu simulačních modelů. Zásadní inovací, 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, které představují instance tříd, v nichž jsou přímo definovány rovnice. Tyto komponenty (tj. instance tříd s rovnicemi) se mohou propojovat prostřednictvím přesně definovaných rozhraní – konektorů a definovat tak soustavy rovnic.

Moderním simulačním jazykem, který je přímo postaven na akauzálním zápisu modelů je ***Modelica*** (*https://modelica.org/*). Jazyk Modelica byl původně vyvinut ve Švédsku a vývojové prostředí jazyka Modelica je dostupné jak ve verzi open-source (*https://www.openmodelica.org/*) vyvíjeného pod záštitou sdružení Open Source Modelica Consortium, tak i v řadě komerčních implementací - např. Dymola od koncernu Dassault Systemes, nebo System Modeler od společnosti Wolfram.

V Modelice je možné vytvářet modely velmi rozsáhlých systémů s desítkami tisíc rovnic a proto Modelica velmi rychle našla své uplatnění zejména v mnoha průmyslových aplikacích, v počítačovém konstruování, v automobilovém a leteckém průmyslu, v energetice a dalších oblastech.

V blokově orientovaných modelovacích jazycích struktura modelu připomíná spíše strukturu výpočtu výstupních proměnných ze vstupů, zatímco struktura modelů vytvářených v Modelice se více blíží struktuře modelovaného systému. Názorným příkladem může být již výše zmíněný Guytonův model oběhového systému. Na obr. 8 je pro srovnání zobrazena centrální část modelu implementovaného v Simulinku a v Modelice. Model v Simulinku přesně odpovídá části grafického schématu v původním článku, zatímco model v Modelice ukazuje strukturu modelované části oběhu - vidíme, že původní Guytonův model, který na schématu vypadá velice složitě, ve skutečnosti pracuje s výrazným zjednodušením reality. Zde je nutno připomenout, že Guyton a jeho žáci model od roku 1972 stále rozvíjeli, současná verze modelu, nazvaná Hummod (*http://hummod.org/*) je podstatně složitější - obsahuje více než 10 000 proměnných [13–17]. My se skupinou následovníků Guytona na Mississippi University již více než patnáct let spolupracujeme a naši verzi modelu Hummod jsme implementovali v Modelice [18–22].

Modelicu dnes využíváme jako základní jazyk pro tvorbu simulačních modelů. V Modelice jsme v např. vytvořili aplikační knihovnu Physiolibrary a Chemical (*https://www.physiolibrary.org/*) pro modelování fyziologických a fyzikálně chemických systémů [23–25]. Tyto knihovny jsme mimo jiné využili při implementaci rozsáhlého integrativního modelu lidské fyziologie (*https://www.physiomodel.org/*)

Tím, že Modelica umožnila zautomatizovat odvození algoritmu výpočtu výstupních proměnných modelu, podstatně zjednodušila a urychlila tvorbu simulačních modelů (obr. 9).

3 Chyceni do pasti...

Pro vytváření výukových simulátorů však samotný simulační model nestačí. Vytvořený simulační model je zapotřebí implementovat do vlastního simulátoru a především naprogramovat jeho uživatelské rozhraní. Podkladem simulátoru jsou vytvořené (a verifikované) matematické modely. Tvorba výukových simulátorů je náročná vývojová práce, 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.

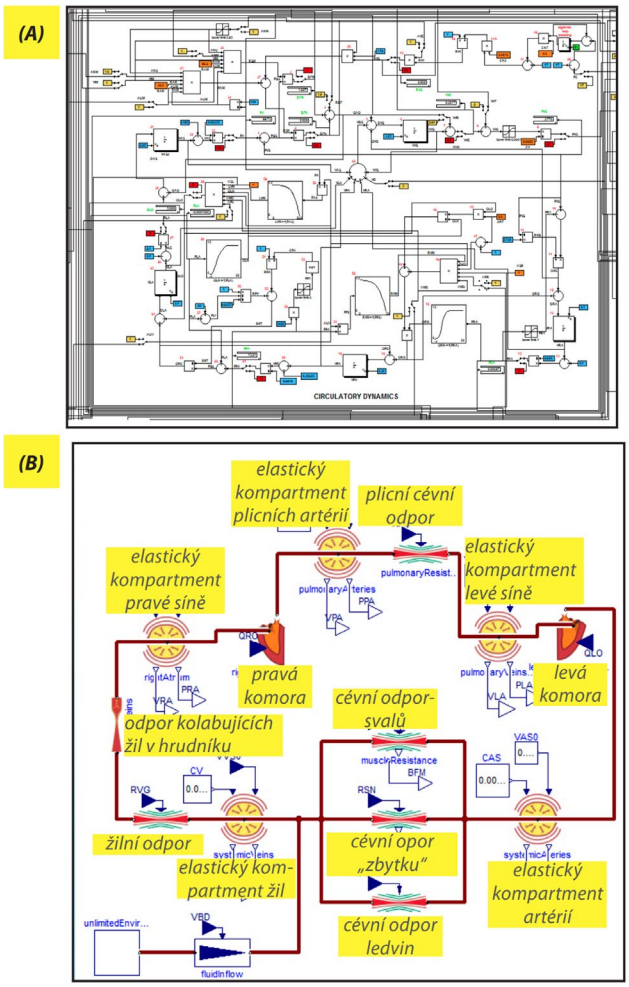
Každý z těchto problémů má své zvláštnosti, a vyžaduje proto použít zcela odlišné vývojové nástroje (viz obr. 10)

Pro tvorbu simulátoru jsme v druhé polovině devadesátých let využívali prostředí pro vizualizaci a řízení technologických procesů Control Web od firmy Moravské Přístroje. Uživatelské rozhraní průmyslové aplikace vytvářené v prostředí ControlWeb komunikovalo s průmyslovým technologickým zařízením.

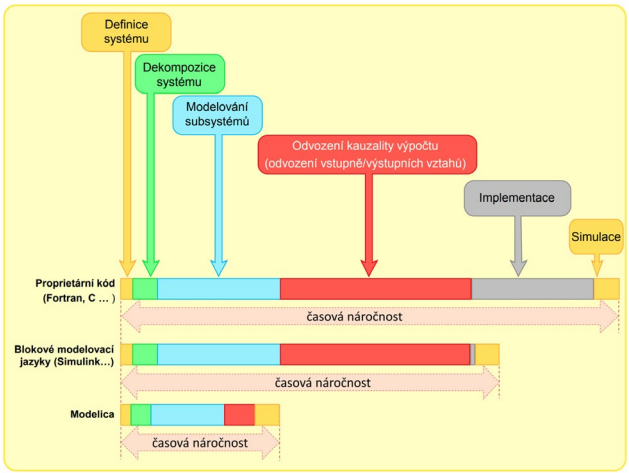
Uživatelské rozhraní našeho simulátoru, naprogramované v prostředí Control Web místo s průmyslovým technologickým zařízením komunikovalo se simulačním modelem. Tímto způsobem jsme např. vytvořili výukový simulátor Golem [11].

Koncem devadesátých let jsme pro vytváření simulátorů začali využívat vývojové nástroje prostředí Windows. Podařilo se nám propojit simulační modely vytvářené v Simulinku a později i v Modelice s prostředím Microsoft .NET. To nám umožnilo programovat simulátory jako aplikace pro prostředí Windows, které

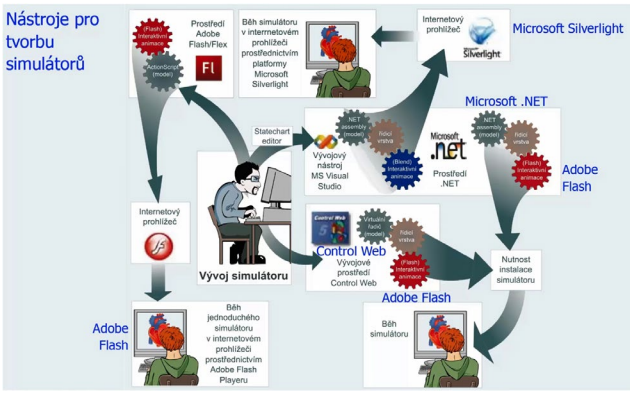
DOI: 10.35191/medsoft_2021_1_33_25_34



Obrázek 8 – Struktura modelu v blokově orientovaných jazycích, např. v SImulinku, připomíná spíše strukturu výpočtu, zatímco struktura modelu v Modelice se více blíží struktutře modelovaného systému. Příkladem je centrální část modelu cirkulační hemodynamiky klasického Guytonova modelu v simulinku a stejná část vytvořená v Modelice. Uvnitř jednotlivých komponent v Modelice jsou příslušné rovnice, jejichž řešení je úlohou překladače jazyka Modelica.



Obrázek 9 – Srovnání náročnosti vytváření modelů v různých prostředích. Blokové modelovací jazyky (např. Simulink) zjednodušily implementaci - vlastní naprogramování modelu na počítači. Nicméně i v blokově orientovaných jazycích je zapotřebí odvodit algoritmus výpočtu výstupních proměnných z hodnot vstupních proměnných a parametrů. Odvození kauzality pomáhají vyřešit na rovnicích založené (equation based) programovací jazyky, kam patří Modelica, které odvození algoritmu výpočtu ze soustavy rovnic nechávají na překladači.



Obrázek 10 – Naše původní technologie tvorby výukových webových simulátorů. Jednoduché výukové modely jsme vytvářeli v prostředí Adobe Flash a jazyce ActionScript. Model pak mohl běžet v prohlížeči s rozšířením Flash player. Složitější modely jsme vytvářeli nejprve v prostředí Simulink/Matlab a později v jazyce Modelica. Pro vytvoření simulátoru bylo nutné propojit matematický model s uživatelským rozhraním, obsahujícím mimo jiné interaktivní grafiku řízenou modelem na pozadí. Pro vytvoření simulátorů jsme využívali různé vývojové nástroje. Využívali jsme např. systém Control-web původně určený pro vytváření průmyslových řídících aplikací a velínů. Později jsme simulátory vytvářeli v prostředí Microsoft .Net. Nakonec jsme využívali také technologii Microsoft Silverlight, která umožňovala vytvářet numericky náročné simulátory s přitažlivým grafickým rozhraním spustitelné přímo v internetovém prohlížeči.

bylo možné stáhnout z našeho atlasu fyziologie a patofyziologie a nainstalovat v počítači s operačním systémem Windows.

V uživatelském rozhraní simulátorů jsme také v hojně využívali interaktivní animované obrázky vytvářené v prostředí Adobe Flash. Animované obrázky mohly být propojeny s výstupy modelu a představovat tak jakési loutky řízené modelem. Jednoduché modely jsme přímo programovali v jazyce ActionScript, kterým se ovládaly flashové animace.

Výhodou modelů vytvářených v jazyce ActionScript bylo to, že se nemusely instalovat a běžely v každém internetovém prohlížeči s nainstalovaným rozšířením FlashPlayer. To šlo ale jen u jednoduchých modelů. Simulátory se složitějšími modely bylo nutné vytvářet jako aplikace, které bylo zapotřebí nainstalovat do počítače.

V roce 2007 se objevila nová technologie Silverlight, kterou Microsoft reagoval na tehdy velmi rozšířený animační program Adobe Flash. Nová technologie od Microsoftu svými možnostmi Flash v mnohém překonala. V Silverlightu bylo možné vytvářet numericky náročné simulátory s přitažlivým grafickým rozhraním spustitelné přímo v internetovém prohlížeči.

Modely jsme přitom mohli pohodlně vyvíjet v prostředí jazyka Modelica. Pomocí námi vyvinutého programu Animtester mohli výtvarníci vytvářet interaktivní animace snadno napojitelné na vstupy a výstupy modelu na pozadí.

Výsledkem pak byly aplikace s animovanými obrázky řízenými modelem na pozadí a celá aplikace potřebovala pouze internetový prohlížeč se zásuvným modulem Silverlight [10,26–28].

Ale právě v tom byl zakopaný pes. Microsoftu se nepodařilo prosadit rozšíření svého Silverlightu na jiné platformy. Společnosti Microsoft, která nakonec v roce 2015 dotáhla Silverlight do páté verze, oznámila ukončení podpory tohoto produktu.

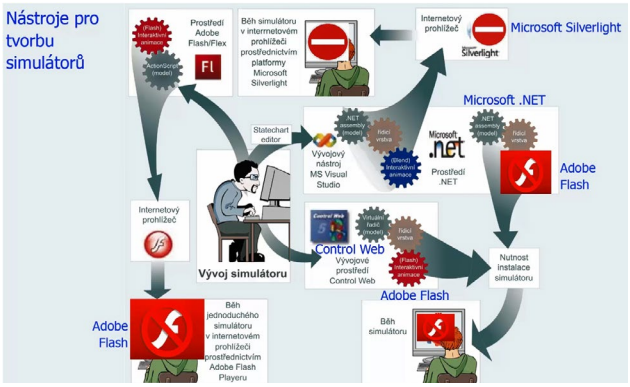
Takže dnes (viz obr. 11) do internetových prohlížečů, včetně prohlížeče Microsoft Edge, zásuvný modul Silverlight nelze instalovat. Do roku 2020 šlo pro běh aplikace Silverlight využít i starý Internetr Explorer, s novou verzí Windows to však již není možné a námi dříve v této technologii vytvořené simulátory nelze spustit.

Kromě toho, společnost Adobe ohlásila konec podpory zásuvného modulu Flash Player v roce 2020, takže od počátku roku 2021 flashové aplikace nelze spouštět v žádném interne-

DOI: 10.35191/medsoft_2021_1_33_25_34

tovém prohlížeci. Aby toho nebylo dost, nový update Windows znemožnil spouštět i jakékoliv aplikace které využívají animační prvky Adobe Flash, což se bohužel týká všech našich simulátorů naprogramovaných pro prostředí Windows.

Znamenalo to, že při tvorbě webových simulátorů jsme se ocitli opět na začátku, a stáli jsme před úkolem vytvořit zcela novou technologii, která nám umožní v tvorbě webových simulátorů pokračovat.



Obrázek 11 – V roce 2015 Microsoft ukončil podporu své technologie Silverlight a zásuvné moduly Silverlight do internetových prohlížečů dnes nelze nainstalovat. Společnost Adobe ohlásila konec podpory zásuvného modulu Flash a od počátku roku 2021 nelze flashové aplikace spouštět v internetových prohlížečích. A k dovršení všeho, nový update Windows dnes znemožňuje spouštět i jakékoli aplikace, které uvnitř využívají animační prvky Adobe Flash, což se bohužel týká všech našich simulátorů naprogramovaných pro prostředí Windows. Znamenalo to, že jsme se ocitli v situaci, kdy veškeré námi vytvořené výukové simulační aplikace nebylo možné spouštět a stáli jsme před úkolem urychleně vytvořit zcela novou technologii, která nám umožní v tvorbě webových simulátorů pokračovat.

Obrázek 12 – Pokud nechceme být pro příště závislí na proprietární technologii nějakého výrobce (jako byla naše dřívější závislost na Adobe Flash, Microsoft Silverlight) musíme se důsledně orientovat na standardy. Interaktivní animovanou grafiku proto vytváříme v JavaScriptu a HTML 5. Standard HTML 5 přinesl animační plátno, což umožňuje vytvářet animace přímo v prohlížeči bez nutnosti zásuvného modulu, jako to bylo dříve u animací vytvořených pomocí Adobe Flash. Nová norma jazyka JavaScript rozšířila možnosti jazyka. Nový webový standard Web Assembly umožňuje v prohlížeči provádět kód téměř tak rychle jako nativní strojový kód. Byl přijat standard Web Components jako standardizovaná forma sdílení uživatelsky vytvářených interaktivních prvků ve webových aplikacích, což umožňuje vytvářet knihovny opakovatelně použitelných webových komponent.

4 Nový začátek – webové simulátory založené na standardech

Na druhé straně nové internetové technologie přinesly zcela nové možnosti i nové výzvy.

Současnou výzvou je nová generace elektronických učebnic s výukovými simulátory, které přinášejí zcela nové možnosti pro vysvětlování složitých dynamických vztahů pomocí simulačních her.

Bylo by velmi užitečné, kdyby simulátor mohl být také součástí wiki-rozhraní, např. wikiskript, kde pohyblivé obrázky řízené modelem na pozadí a interaktivní grafy by vhodně doplňovali výklad.

Chceme, aby naše výukové simulátory nebyly příliš závislé na platformě a operačním systému, na kterém běží.

Společným prvkem uživatelských výpočetních zařízení - od chytrých telefonů, přes tablety až k počítačům, bez ohledu na operační systém, na kterém pracují, je webová platforma. Chceme-li vytvořit na platformě nezávislou technologii simulátorů, jsou internetové prohlížeče jedinou možností, jak tento cíl realizovat.

Internetové prohlížeče se v posledních několika letech zásadně změnily.

Nové internetové technologie a standardy změnily dnešní prohlížeče k nepoznání. Nepotřebujeme v prohlížečích zásuvné proprietární moduly od třetích stran jako byl Flash nebo Silverlight, jenom proto abychom mohli v prohlížeči vytvářet aplikace podobné těm desktopovým.

Díky vytrvalému mezinárodnímu úsilí dobrovolníků, standardizačních skupin a velkých internetových společností, dostupnosti sdílených knihoven a softwarových vývojových nástrojů, je dnes možné vytvářet webové aplikace jako nikdy předtím.

Při programování numericky náročných aplikací (jako jsou často i simulační modely) nejsme omezeni pomalou rychlostí interpretovaného javascriptu. Nový standard WebAssembly umožňuje spouštět v prohlížečích numericky náročné aplikace, takže v prohlížeči je dnes možné provádět i složité simulační výpočty.

Webový prohlížeč se v současné době stal tím, čím byl operační systém personálních počítačů v devadesátých letech minulého století.

Po dlouhých letech „válek webových prohlížečů“, kdy mezi sebou soupeřili různí výrobci internetových prohlížečů, konečně došlo k dohodě o nových webových standardech, které přispívají k rozvoji webových aplikací (obr. 12).

Nové webové standardy se snaží omezit nutnost implementace různých zásuvných modulů, a snaží se formou standardů nabídnout takové možnosti, které by dosud používané zásuvné moduly nahradily. Obětí byl již zmiňovaný flash player, který byl z nových prohlížečů vyhozen.

Možnosti animací přináší nová norma **HTML 5**, která zavádí animační plátno, na němž lze vytvářet animace pomocí jazyka JavaScript.

Nová norma jazyka **ECMAScipt** 6 zásadně rozšířila možnosti skriptovacího jazyka JavaScript.

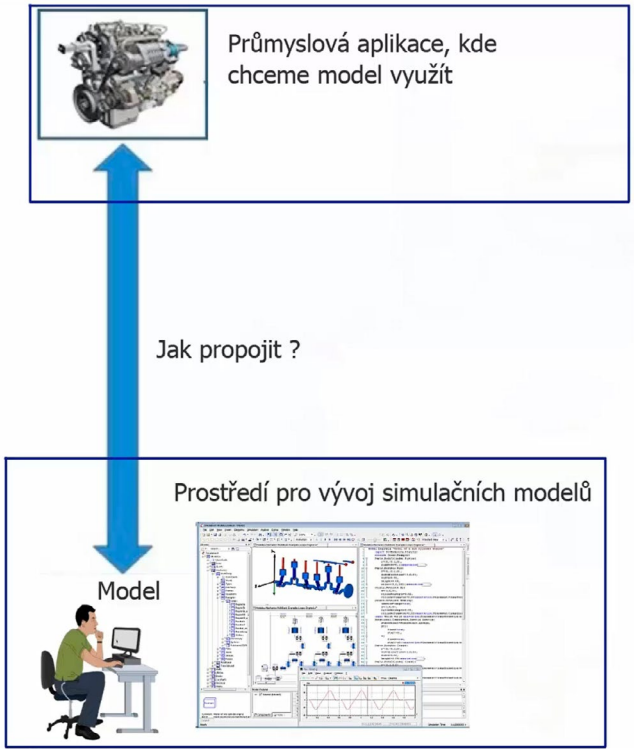
Moderní prohlížeče implementovaly nový webový standard, nazvaný **WebAssembly**, který definuje formát speciálního jazyka pro přenositelný strojový kód spustitelný na webových stránkách. Umožňuje provádět kód téměř tak rychle, jako nativní strojový kód a tím podstatně urychluje výpočty prováděné přímo ve webovém prohlížeči.

Byl definován standard Web Components, který umožňuje vytvářet opakovaně použitelné webové komponenty jako nové prvky jazyka HTML.

Simulace jsou dnes často využívanou metodou, nejen ve vědě a výzkumu, ale zejména v průmyslu.

DOI: 10.35191/medsoft_2021_1_33_25_34

Simulační modely se obvykle vytvářejí ve specializovaných programových prostředích, které poskytují nástroje pro rychlý vývoj modelů. Problém ale nastává, pokud chceme model, vytvořený a odladěný v pohodlí nástroje pro vývoj simulačních modelů, využít v nějaké aplikaci. Často nezbylo nic jiného, než model ručně přepsat do příslušného programovacího jazyka nebo vytvořit vlastní software, který umožní propojit model vytvořený ve vývojového prostředí s příslušnou aplikací (obr. 13).



Obrázek 13 – S rozvojem informačních technologií se simulační modely stále více využívají v průmyslu. Modely jsou vytvářeny ve specializovaných programových nástrojích. Pokud vytvořený model chceme využít v nějaké průmyslové aplikaci, pak vzniká problém, jak propojit program, řídící nějakou průmyslovou aplikaci se simulačním modelem vytvářeným v pohodlí programovacího prostředí.

Ještě větší problém nastává, pokud v nějaké složitější technologické aplikaci chceme propojit více modelů často navíc vytvářených v různých vývojových nástrojích. To je situace, s níž se setkáváme při konstruování složitých technologických celků, např. v automobilovém nebo leteckém průmyslu (obr. 14).

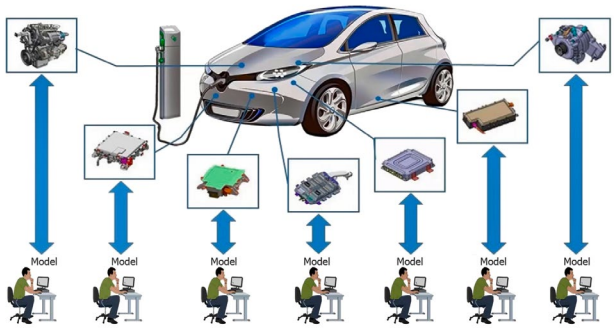
Proto, zejména ze strany průmyslu byly podporovány snahy o vytvoření určitých standardů, které by propojování modelů usnadnily.

Výsledkem je nový mezinárodní standard **FMI (Functional Mock up interface** – funkční maketové rozhraní), který definuje způsob propojení simulačních modelů mezi sebou a s okolním prostředím (obr. 15).

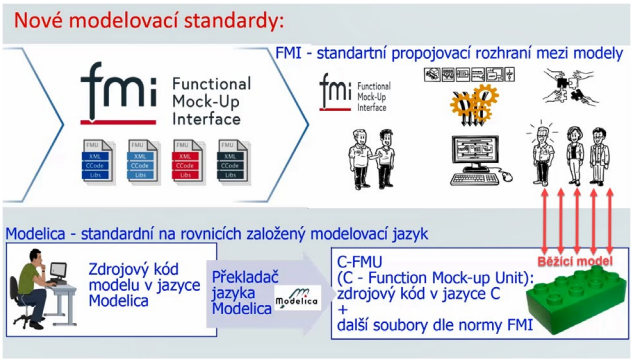
Podle tohoto standardu např. můžeme přeložit zdrojový kód modelu v jazyce Modelica do standardizovaného zdrojového kódu v jazyce C, provázeného dalšími popisnými soubory.

To ale zdaleka není jen pouhý kus kódu v jazyce C, vytvořený jako jedna z komponent pro nějakou softwarovou aplikaci. Tato komponenta obsahuje model, s nímž lze definovaným způsobem komunikovat – tj. zadávat mu parametry a počáteční hodnoty, model spouštět, zastavovat, měnit jeho parametry a znovu ho spouštět a přitom kontinuálně do něj posílat hodnoty vstupů a sledovat jak na ně reagují jeho výstupy.

Je to vlastně jakási "kouzelná skříňka", nazývaná funkční maketovou jednotkou (**Function Mock-up Unit**) s ukrytým fungujícím simulačním modelem. Může být třeba součástí programu, který řídí nějakou technologii, nebo může komunikovat s jiným modelem ukrytým v jiné komponentě také vytvořené podle standardu FMI. Tento standard funkčního maketového rozhraní doširoka otevírá dveře praktickému využití simulačních modelů v mnoha oborech.



Obrázek 14 – Při konstruování komplexnějších technologických celků, např. v automobilovém nebo leteckém průmyslu, se často využívají modely jednotlivých komponent. Pak nastává problém, jak propojit software ovládající tyto komponenty s modely vytvářenými často i v různých nástrojích pro tvorbu modelů. Proto, zejména ze strany průmyslu, byly podporovány snahy o vytvoření určitých standardů, které by propojování modelů usnadnily.



Obrázek 15 – Na základě požadavků průmyslu byl vytvořen nový mezinárodní standard nazvaný FMI (Functional Mock up interface – funkční maketové rozhraní), který definuje způsob propojení simulačních modelů mezi sebou a s okolním prostředím. Podle tohoto standardu např. můžeme přeložit zdrojový kód modelu v jazyce Modelica do standardizovaného zdrojového kódu v jazyce C, provázeného dalšími popisnými soubory. To ale není jen kus kódu v jazyce C, který je možné využít jako jednu z komponent při tvorbě nějaké softwarové aplikace. Tato komponenta obsahuje model, s nímž lze definovaným způsobem komunikovat – zadávat mu parametry a počáteční hodnoty, model spouštět, zastavovat, měnit jeho parametry a znovu ho spouštět a přitom kontinuálně do něj posílat hodnoty vstupů a sledovat jak na ně reagují jeho výstupy. Je to taková kouzelná skříňka, nazývaná funkční maketovou jednotkou (Function Mock-up Unit - C-FMU) s ukrytým simulačním modelem, naprogramovaným v jazyce C. Může být třeba součástí programu, který řídí nějakou technologii, nebo může komunikovat s jiným modelem ukrytým v jiné komponentě také vytvořené podle standardu FMI. Tento standard funkčního maketového rozhraní doširoka otevírá dveře praktickému využití simulačních modelů v mnoha oborech.

A obdobně, jako webové standardy podnítily současný rozkvět webových technologií tak i nové modelovací standardy přispívají k rozvoji simulací.

Pro tvorbu webových simulátorů jsou jak modelovací tak i webové standardy klíčové.

DOI: 10.35191/medsoft_2021_1_33_25_34

5 Technologie Bodylight.js

Na otevřených webových a modelovacích standardech je založena naše nová technologie tvorby webových simulátorů, kterou jsme nazvali Bodylight.js [29–31] (*https://bodylight.physiome.cz/*).

Interaktivní animovanou grafiku vytváříme v JavaScriptu a HTML 5. Vytvářet takové grafické aplikace např. umožňuje program Animate od firmy Adobe (obr. 16). Model vytváříme v jazyce Modelica, přeložíme ho do jazyka C podle standardu FMI - výsledkem je funkční maketová jednotka (C-FMU), kterou transpilujeme do JavaScriptu a WebAssembly. Pro interaktivní vývoj webové aplikace jsme vytvořili nástroj BodyLight Composer v němž propojíme model s grafikou a dalšími ovládacími a vizuálními prvky a vytvoříme tak vizuální tvář celé aplikace.

Bodylight Composer nám umožní vygenerovat webovou simulační aplikaci sestávající z HTML5, JavaScriptu a webAssembly, kterou mohou spouštět všechny moderní internetové prohlížeče (v počítači, v tabletu či v chytrém telefonu) v různých operačních systémech. Výsledná aplikace běží (včetně grafiky a simulačních numerických výpočtů) přímo ve webovém prohlížeči - proto hovoříme o in-browser simulátoru. Vygenerovaná aplikace nevyžaduje trvalé připojení k internetu.

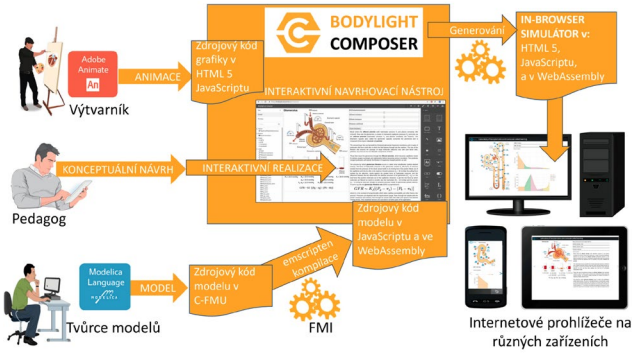
Určitým problémem je to, že při tvorbě aplikace jsme do určité míry omezeni nabídkou komponent (grafů, tlačítek, posuvníků..), které nabízí Bodylight Composer, který byl vytvořen v prostředí frameworku React. Pokud chceme vytvořit novou komponentu, která není v nabídce, musíme ji buď vytvořit jako interaktivní grafický obrázek (v HTML 5 a Javascriptu) nebo ji v Reactu doprogramovat.

Nejsme ale omezeni jen na Bodylight Composer.

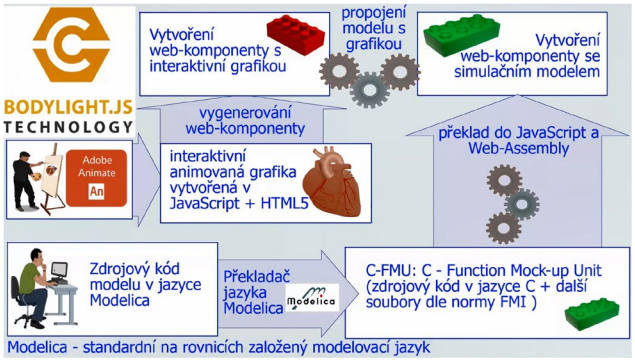
Abychom umožnili větší flexibilitu při vytváření webových simulačních aplikací, rozšířili jsme naši technologii o tvorbu standardních webových komponent.

Jak model, tak i webová interaktivní grafika jsou pak realizovány dle standardu webových komponent (obr. 17).

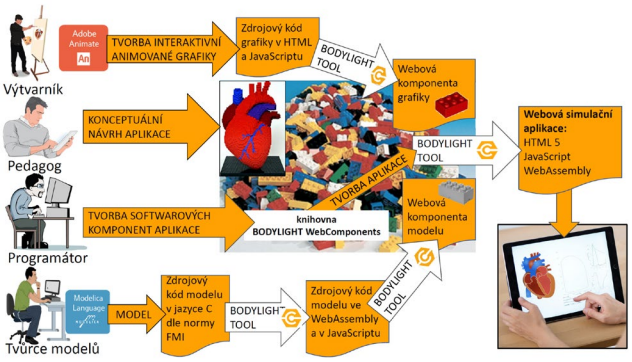
Vytvořili jsme knihovnu "Bodylight.js components", která obsahuje širokou, stále se rozšiřující nabídku standardizovaných webových komponent (obr. 18).



Obrázek 16 – Naše technologie tvorby webových simulátorů, kterou jsme nazvali "Bodylight.js" je založena na využití webových a modelovacích standardů. Pro vývoj aplikace v naší nové technologii tvorby webových simulátorů jsme vytvořili interaktivní vývojový nástroj Bodylight Composer. Výtvarník do tohoto nástroje umístí grafické komponenty ve formě HTML5 a zdrojového kódu v jazyce JavaScript. Model původně vytvořený v jazyce Modelica a přeložený do jazyka C ve formě tzv. jednotek C-FMU převedeme (emscripten kompilací) do JavaScriptu a WebAssembly. V nástroji Bodylight Composer navrhneme vizuální tvář aplikace a propojení grafiky s proměnnými modelu. Nástroj nám umožní vygenerovat simulátor jako webovou aplikaci sestávající z HTML5 a zdrojových kódů v jazyce Javascript a WebAssembly. Tuto aplikaci pak mohou přehrávat všechny moderní webové prohlížeče. Grafika i simulační výpočty probíhají uvnitř internetového prohlížeče (proto hovoříme o in-browser simulačních výpočtech). Výsledná aplikace je proto spustitelná v internetovém prohlížeči na jakýchkoli zařízeních a operačních systémech a nevyžaduje trvalé připojení k internetu.



Obrázek 17 – Interaktivní nástroj Bodylight Composer byl naprogramován ve frameworku React. Umožňuje interaktivně sestavovat webovou stránku propojováním simulačních modelů, grafických objektů a předpřipravených komponent (grafů, tlačítek, posuvníků aj.). Pokud chceme rozšířit nabídku komponent v nástroji Bodylight Composer, musíme příslušnou komponentu do nástroje Bodylight Composer v Reactu doprogramovat. Abychom toto omezení obešli, a umožnili větší flexibilitu vytváření webových simulačních aplikací, rozšířili jsme naši technologii o tvorbu standardizovaných webových komponent obsahujících grafiku a simulační modely. Webovou komponentu grafiky můžeme propojovat s webovou komponentou modelu a vytvářet tak animovanou grafiku jako grafickou loutku, řízenou modelem. Standardizované webové komponenty můžeme využívat ve webových aplikacích bez vazby na konkrétní vývojový framework.



Obrázek 18 – Součástí naší technologie Bodylight.js je neustále se rozšiřující knihovna standardizovaných webových komponent "Bodylight.js web components". Na základě konceptuálního návrhu pedagoga výtvarník vytváří prvky interaktivní animované grafiky a tvůrce modelů vytváří příslušný model v jazyce Modelica. S využitím naší knihovny webových komponent pak vytváříme webovou simulační aplikaci ve formě HTML, jazyce JavaScript a webAssembly, která je pak spustitelná v internetových prohlížečích na různých zařízeních a platformách v různých operačních systémech.

Těmito tzv. vlastními elementy (custom elements) lze obohatit HTML stránku, nebo WIKI zdroj (třeba Wiki skripta) o prvky, které webový simulátor vybaví numerickým řešičem modelu, propojí proměnné modelu s animací, propojí grafy s proměnnými modelu, napojí tlačítka a posuvníky a další prvky s měnitelnými parametry modelu aj.

Využití knihovny vlastních webových komponent pak snižuje závislost aplikace na cizích knihovnách a frameworcích. Tvůrce webového simulátoru pak může použít plnou vizualizační sílu HTML a CSS bez nutností navázání na další framework, nebo, naopak může využít jakýkoliv svůj oblíbený framework nebo knihovnu.

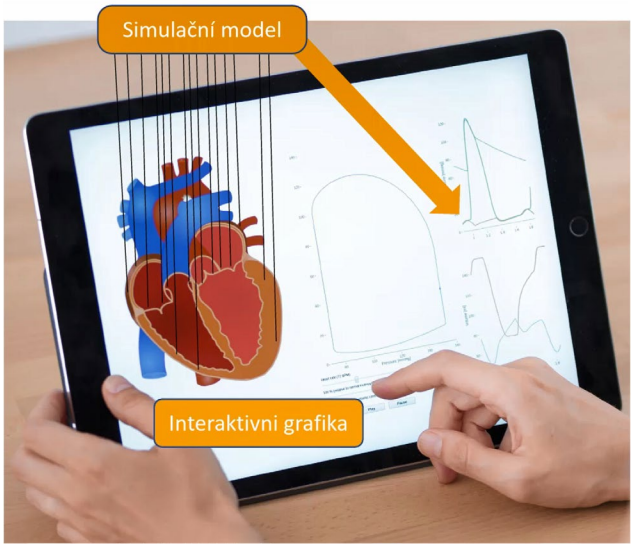
Pro tvorbu standardizovaných webových komponent dnes existují frameworky - např. Aurelia (*https://aurelia.io*) nebo Vue (*https://vuejs.org*) - což dává velkou flexibilitu pro doplnění naši knihovny vlastních webových komponent podle požadavků na aplikaci.

6 ZÁVĚR

Naše technologie umožňuje vytvářet nový typ elektronických učebnic, spustitelných na jakémkoliv zařízení s internetovým prohlížečem, propojujících hypertext, multimedia, simulační modely a interaktivní grafiku řízenou modelem na pozadí (obr. 19).

Taková učebnice by pak byla skutečná škola hrou pro 21. století, která pomocí simulačních her může pomoci studentům pochopit složité dynamické souvislosti současného světa.

Technologie je ale jen předpoklad - podstatný je didaktický obsah, který nový druh elektronických učebnic může učinit skutečností.



Obrázek 19 – Naše nová technologie umožňuje vytvářet výukové simulační aplikace, spustitelné v internetovém prohlížeči, které obsahují interaktivní grafiku, řízenou modelem na pozadí (jako jakési modelem řízené loutky). Vše může být provázeno hypertextem, multimediálními doplňky (zvuk, video) a vytvářet tak elektronické učebnice zcela nového typu.

Literatura

[1.] Kofránek J. Modelování acidobazické rovnováhy krve, (Modeling of blood acid-base). Dissertation Thesis CSc. (Ph.D.), Faculty on general Medicine, Charles University. 1980.

[2.] Kofránek J, Munclinger M, Šerf B, Fusek M, Kautzner J, Duchác V, et al. Evaluation of Cardiorespiratory Functions during Heart Catheterisation through Simulation Model Identification. Advances in Biomedical Measurement. Springer, Boston, MA; 1988. pp. 311–319.

[3.] Guyton AC, Coleman TG, Granger HJ. Circulation: overall regulation. Annu Rev Physiol. 1972;34: 13–46.

[4.] Coleman TG, Summers RL. Using mathematical models to better understand integrative physiology. J Physiol Biochem. 1997;53: 45–46.

[5.] Ikeda N, Marumo F, Shirataka M, Sato T. A model of overall regulation of body fluids. Ann Biomed Eng. 1979;7: 135–166.

[6.] Amosov NM, Palec BL, Agapov BT, Jermakova II, Ljabach EG, Packina SA, et al. Theoretical research of physiological systems: matematical modeling (in Russian). Naukova Dumka; 1977.

[7.] Coleman TG, Randall JE. HUMAN. A comprehensive physiological model. Physiologist. 1983;26: 15–21.

[8.] Kofránek J, Rusz J, Matoušek S. Guytons diagram brought to life-from graphic chart to simulation model for teaching physiology. Technical Computing Prague 2007. dsp.vscht.cz; 2007. pp. 978–980.

[9.] Fontecave-Jallon J, Thomas SR. Implementation of a model of bodily fluids regulation. Acta Biotheor. 2015;63: 269–282.

[10.] Kofránek J, Kripner T, Andrlík M, Mašek J. Creative connection between multimedia, simulation and software development tools in the design and development of biomedical educational simulators. Proceedings of Simulation Interoperability Workshop, Orlando 2003, Position papers, Volume II. SISo Inc.; 2003. pp. 677–687.

[11.] Kofránek J, Vu LDA, Snaselova H, Kerekes R, Velan T. GOLEM-multimedia simulator for medical education. Stud Health Technol Inform. 2001; 1042–1046.

[12.] Kofránek J, Andrlík M, Kripner T, Mašek J, Velan T. Simulation chips for GOLEM - multimedia simulator of physiological functions. Simulation in Health and Medical Sciences. Society for Computer Simulation International, Simulation Councils, San Diego; 2002. pp. 159–163.

[13.] Hester R, Brown A, Husband L, Iliescu R. HumMod: a modeling environment for the simulation of integrative human physiology. Frontiers in Physiology. 2011. Available: http://journal.frontiersin.org/article/10.3389/fphys.2011.00012

[14.] Lerant AA, Hester RL, Coleman TG, Phillips WJ, Orledge JD, Murray WB. Preventing and Treating Hypoxia: Using a Physiology Simulator to Demonstrate the Value of Pre-Oxygenation and the Futility of Hyperventilation. Int J Med Sci. 2015;12: 625–632.

[15.] Pruett WA, Clemmer JS, Hester RL. Validation of an integrative mathematical model of dehydration and rehydration in virtual humans. Physiol Rep. 2016;4: 1–20.

[16.] Sims CR 3rd, Delima LR, Calimaran A, Hester R, Pruett WA. Validating the Physiologic Model HumMod as a Substitute for Clinical Trials Involving Acute Normovolemic Hemodilution. Anesth Analg. 2018;126: 93–101.

[17.] Hester RL, Pruett W, Clemmer J, Ruckdeschel A. Simulation of integrative physiology for medical education. Morphologie. 2019. doi:10.1016/j.morpho.2019.09.004

[18.] Kofranek J, Matejak M, Privitzer P. Hummod-large scale physiological models in modelica. Proceedings of the 8th International Modelica Conference; March 20th-22nd; Technical Univeristy; Dresden; Germany. Linköping University Electronic Press; 2011. pp. 713–724.

[19.] Kofránek J, Mateják M, Privitzer P, Tribula M, Kulhánek T, Šilar J, et al. HumMod-Golem Edition: large scale model of integrative physiology for virtual patient simulators. Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp); 2013. p. 1.

[20.] Mateják M, Kofránek J. Physiomodel - an integrative physiology in Modelica. 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). 2015. pp. 1464–1467.

[21.] Mateják M. Formalization of Integrative Physiology. Charles University in Prague. Kofránek J, editor. Ph.D., Charles University. 2015. Available: https://github.com/MarekMatejak/dissertation/blob/master/thesis.pdf

[22.] Kofránek J, Kulhánek T, Mateják M, Ježek F, Šilar J. Integrative physiology in Modelica. Proceedings of the 12th International Modelica Conference, Prague, Czech Republic, May 15-17, 2017. Linköping University Electronic Press; 2017. pp. 589–603.

[23.] Mateják M, Kulhánek T, Šilar J, Privitzer P, Ježek F, Kofránek J. Physiolibrary-Modelica library for physiology. Proceedings of the 10 th International Modelica Conference; March 10-12; 2014; Lund; Sweden. Linköping University Electronic Press; 2014. pp. 499–505.

[24.] Mateják M, Ježek F, Tribula M, Kofránek J. Physiolibrary 2.3-An Intuitive Tool for Integrative Physiology. IFAC-PapersOnLine. 2015;48: 699–700.

[25.] Matejak M, Tribula M, Ježek F, Kofranek J. Free Modelica Library for Chemical and Electrochemical Processes. Proceedings of the 11th International Modelica Conference, Versailles, France, September 21-23, 2015. Linköping University Electronic Press; 2015. pp. 359–366.

[26.] Kofranek J, Matousek S, Rusz J, Stodulka P, Privitzer P, Matejak M, et al. The Atlas of Physiology and Pathophysiology: Web-based multimedia enabled interactive simulations. Comput Methods Programs Biomed. 2011;104: 143–153.

DOI: 10.35191/medsoft_2021_1_33_25_34

[27.] Kofránek J, Andrlík M, Kripner T, Mašek J, Stodůlka P. "Od umění k průmyslu" - propojení technologií při tvorbě lékařských výukových programů. Medsoft. 2003;15: 43–56.

[28.] Stodulka P, Privitzer P, Kofránek J, Mašek J. Nové postupy v tvorbě simulátorů-inteligentní propojení Matlabu a Simulinku s platformou .NET a tvorba stavových automatů řídících výslednou aplikaci. Medsoft. 2006;18: 177–184.

[29.] Polák D, Ježek F, Šilar J, Kofránek J. Technologie tvorby webových simulátorů. MEDSOFT. 2019;31: 122–139.

[30.] Šilar J, Polák D, Mládek A, Ježek F, Kurtz TW, DiCarlo SE, et al. Development of In-Browser Simulators for Medical Education: Introduction of a Novel Software Toolchain. J Med Internet Res. 2019;21: e14160.

[31.] Kofránek J, Kulhánek T, Mateják M, Ježek F, Šilar J, Mládek A, et al. Schola Ludus for the 21st century: simulators in the Internet browser. FASEB J. 2020;34: 1–1.

Poděkování

Tato práce vznikla za přispění grantů MPO Trio FV20628 Lékařský trenažér a MPO Trio FV30195 Robotické mechanotronické trenažéry s rozšířenou realitou pro lékařskou výuku.

Kontakt

doc. MUDr. Jiří Kofránek, CSc.
Oddělení biokybernetiky
Ústav patologické fyziologie 1. LF UK
e-mail: *kofranek@gmail.com*
tel: +420 777 68 68 68

Mgr. Tomáš Kulhánek, Ph.D.
Oddělení biokybernetiky
Ústav patologické fyziologie 1. LF UK
e-mail: *tmkulhanek@gmail.com*
tel: +420 775 178 931