SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Evidenčné číslo: FCHPT-5414-61855

Navrhovanie simulinkovej schémy cez Internet

Diplomová práca

2016 **Bc. Nikola Míková**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta chemickej a potravinárskej technológie

Evidenčné číslo: FCHPT-5414-61855

Navrhovanie simulinkovej schémy cez Internet

Diplomová práca

Študijný program: automatizácia a informatizácia v chémii a potravinárstve
Študijný odbor: 5.2.14. automatizácia
Školiace pracovisko: Ústav informatizácie, automatizácie a matematiky
Vedúci záverečnej práce: Ing. Ľuboš Čirka, PhD.

Bratislava 2016 **Bc. Nikola Míková**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2015/2016
Evidenčné číslo: FCHPT-5414-61855

STU FCHPT

ZADANIE DIPLOMOVEJ PRÁCE

Študentka: **Bc. Nikola Míková**

ID študenta: 61855

Študijný program: automatizácia a informatizácia v chémii a potravinárstve

Študijný odbor: 5.2.14. automatizácia

Vedúci práce: Ing. Ľuboš Čirka, PhD.

Názov práce: **Navrhovanie simulinkovej schémy cez Internet**

Špecifikácia zadania:

Cieľom práce je vytvorenie webovej aplikácie, ktorá umožní užívateľovi vytvárať jednoduchú schému v okne webového prehliadača. Po vykreslení a následnom uložení vygeneruje schému pre program Simulink vo formáte slx.
Aplikácia bude vytvorená v štandardných jazykoch: HTML, Javascript, PHP a SQL (MySQL).

Úlohy:
1. Literárna rešerš.
2. Štúdium použitých technológii.
3. Štúdium štruktúry schém v Simulinku.
4. Vytvorenie aplikácie.
5. Testovanie aplikácie.
6. Vypracovanie dokumentácie.

Rozsah práce: 50

Zoznam odbornej literatúry:

1. Škultéty, R. *JavaScript: Programujeme internetové aplikace.* Praha : Computer Press, 2005. 224 s. ISBN 80-251-0144-4.
2. Resig, J. *JavaScript a Ajax : Moderní programování webových aplikací.* Praha: Computer Press, 2007. 360 s. ISBN 978-80-251-1824-5.
3. David, F. *JavaScript: Kompletní průvodce.* Praha: Computer Press, 2002. 825 s. ISBN 80-7226-626-8.
4. Odell , D. *JavaScript. Průvodce programováním ajaxových aplikací.* Praha: Computer Press, 2010. 368 s. ISBN 978-80-251-2733-9.

Riešenie zadania práce od: 15. 02. 2016

Dátum odovzdania práce: 22. 05. 2016

L. S.

Bc. Nikola Míková
študentka

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Poďakovanie

Touto cestou by som sa chcela poďakovať svojmu vedúcemu diplomovej práce, Ing. Ľubošovi Čirkovi, PhD. za všetky jeho rady, pripomienky a pomoc, ktorú mi počas vypracovávania práce poskytol. Ďakujem.

Abstrakt

Diplomová práca sa zaoberá vytváraním webovej aplikácie, ktorá umožní užívateľovi vytvoriť jednoduchú blokovú schému vo webovom prehliadači. Túto schému je možné následne vygenerovať vo formáte slx. Vygenerovaná schéma musí byť spustiteľná v programe MATLAB/Simulink. Aplikácia má uplatnenie pri modelovaní a riadení systémov v rámci vzdialených laboratórií. Online Simulink aplikácia je vytvorená v technológiách a jazykoch ako HTML, CSS, JavaScript, PHP, AJAX, JSON, XML.

Kľúčové slová: Simulink; bloková schéma; JSON

Abstract

Diploma thesis deals with creating web application that allows users to create a simple block diagram of a web browser. The scheme may then be generated in the format SLX. The generated schema must be executable in Matlab / Simulink. The application has an application in modeling and control systems in the remote laboratories. Simulink Online application is created in technologies and languages such as HTML, CSS, JavaScript, PHP, AJAX, JSON, XML.

Key words: Simulink; block diagram; JSON

Obsah

Zoznam obrázkov

Zoznam kódov

Zoznam skratiek

MDL Metadata Description Language

XML eXtensible Markup Langugage

MATLAB MAtrix LABoratory

HTML HyperText Markup Language

CSS Cascading Style Sheets

SEO Search Engine Optimization

AJAX Asynchronous JavaScript and XML

W3C World Wide Web Consortium

DTD Document Type Definition

XHTML eXtensible HyperText Markup Language

DOM Document Object Model

PHP Personal Home Page

SQL Structured Query Language

JSON JavaScript Object Notation

1 Úvod

Internet má v dnešnom svete dôležitú úlohu. Jeho vplyv zasahuje do viacerých oblastí každodenného života. Jednou z oblastí je aj vzdelávanie. Práve preto sa školy snažia zaujať svojich študentov modernými aplikáciami, ako sú virtuálne laboratóriá, ktoré sú umiestnené na stránkach fakúlt.

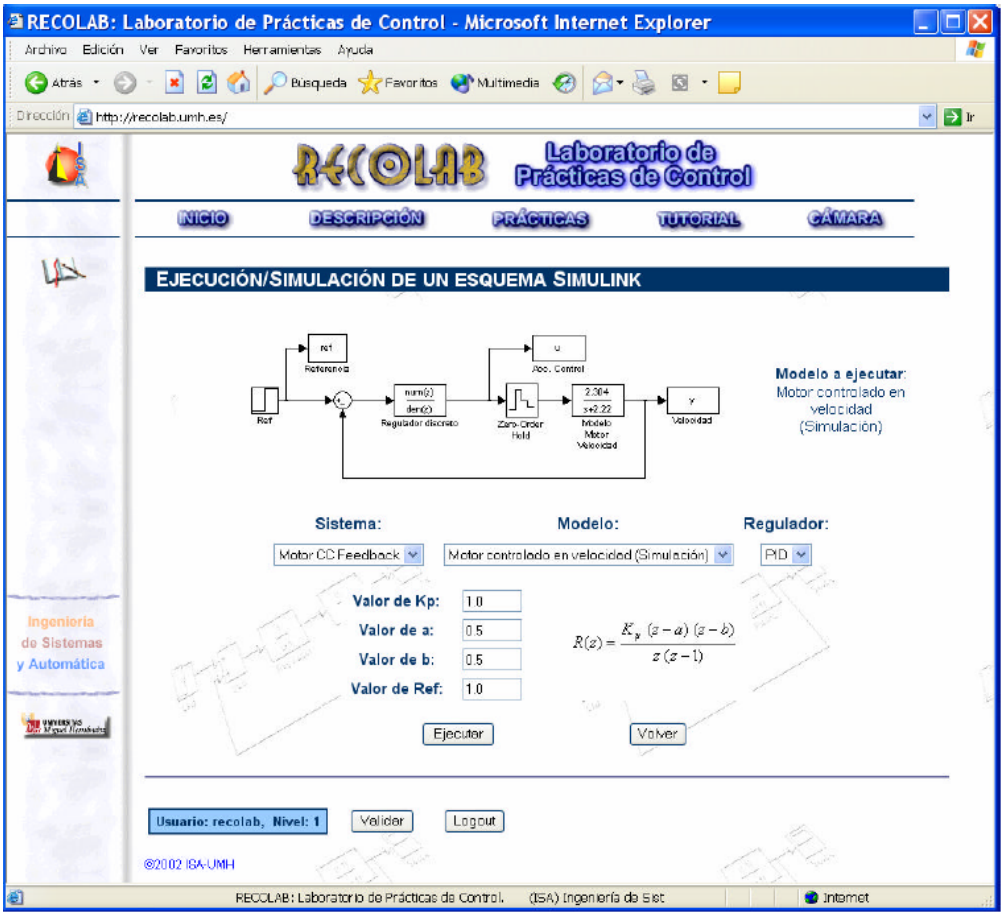
Aj na stránkach Fakulty chemickej a potravinárskej technológie sa nachádzajú virtuálne laboratória, do ktorých majú študenti dostupnosť 24 hodín denne. Môžu si tu vyskúšať svoje poznatky získané na vyučovaní, ale čo je najhlavnejšie, otestovať riešenie v „laboratóriu“. Študenti k nim vedia pristupovať prostredníctvom webových prehliadačov aj mimo univerzity. V takýchto virtuálnych laboratóriách je možné nájsť modely procesov, ktoré je možné ovládať prostredníctvom blokových schém vytvorených v Simulink-u. Veľkou nevýhodou je, že študent má k dispozícii už pripravenú schému, v ktorej vie upravovať len niektoré parametre.

Na Univerzite Miguela Hernándeza (Alicante, Španielsko) vyvíjali aplikáciu pre vzdialené laboratórium, ktorá má pomôcť pri výučbe teórie riadenia. Na obr. 1 je zobrazená stránka aplikácie RECOLAB. Ako je možné vidieť, študent nemá možnosť vytvárať blokovú schému. Má možnosť len meniť určité parametre pomocou formulára. [1]

V diplomovej práci je opísaná tvorba webovej aplikácie, ktorá umožňuje vytvárať nové blokové schémy vo webovom rozhraní, s ktorými je následne možná simulácia vo virtuálnych laboratóriách. Užívateľ vie vytvoriť schému pomocou vkladania blokov z knižnice blokov do okna schémy. Výhodou tejto aplikácie je možnosť úpravy blokov, ako je ich pridávanie, zmazanie, či nastavenie parametrov bloku.

V práci som sa najprv venovala analýze problému. Zaoberala som sa štruktúrou súboru `blockdiagram.xml`, ktorý je podstatnou časťou komprimovaného súboru tvoriaceho simulinkovú schému. Študovala som zmeny štruktúry súboru po pridaní bloku, po jeho odstránení, zmene parametrov, po pridaní spojenia a pod. Zmeny v štruktúre som preštudovala od verzie MATLAB R2012a až po R2014a.

V kapitole Použité technológie sú opísané dostupné riešenia pre online tvorbu simulinkovej schémy. Je tu opísaný spôsob ukladania dát, možnosti kreslenia tvarov, architektúra aplikácie, použité jazyky, knižnice a technológie v rámci tvorby aplikácie.



Obr. 1: RECOLAB - Stránka pre riadenie vo vzdialenom laboratóriu [1]

V praktickej časti je opísaná samotná činnosť webovej aplikácie, ktorá umožní užívateľovi vytvárať jednoduchú schému v okne webového prehliadača. Ďalej sú tu opísané využité funkcie jQuery či iných knižníc a pozadie celej aplikácie. V tejto kapitole sa nachádza aj podrobný opis pridávania nového bloku do knižnice blokov.

2 Súčasný stav danej problematiky

V tejto kapitole je opísaný súčasný stav danej problematiky na základe publikácií o vyvinutých a vyvíjajúcich sa podobných aplikáciách.

Na internete je možné nájsť množstvo online dostupných kresliacich aplikácií, ktoré umožňujú užívateľovi zostrojiť schému/diagram. Napríklad kresliace nástroje: Gliffy [3], Insight Maker [4], Draw.io [5], a iné. Insight Maker je dokonca vo výsledku vyhľadávania nazvaný aj ako online Simulink. Táto aplikácia umožňuje užívateľovi kresliť schémy, simulovať, výsledky pozorovať v grafoch.

V nájdených aplikáciách ale neexistuje knižnica blokov, akú obsahuje Simulink, bloky majú možnosť byť spájané viacerými spojeniami a pod. Aplikácie sú vhodné na vykresľovanie iných typov diagramov. Napríklad Draw.io. Tento nástroj je vhodný pre tvorbu UML diagramov.

Pri porovnávaní existujúcich aplikácii so Simulink-om, som porovnávala aj formát vygenerovaných schém online aplikácií a Simulink-u. Vygenerovaný súbor v uvedených aplikáciách nie je kompatibilný so Simulinkom. Simulink požaduje ako vstupný súbor formát SLX, ktorý je komprimovaným súborom. Tento súbor obsahuje obrázky PNG a súbory XML, v ktorých sú uložené údaje o schéme. Štruktúre tohoto súboru sa bližšie venujem v podkapitole 3.1. Žiadna internetová aplikácia neexportuje schémy v súbore tohto typu.

Internetové aplikácie sú založené na javascriptových knižniciach ako napríklad MxGraph, jsDraw2D, Raphaël, JointJs, Fabric atď. Knižnice ponúkajú množstvo už zadefinovaných funkcií, ktoré je možné využiť v aplikácii. Výber vhodných knižníc sa pri vývoji mojej aplikácie menil.

V prvej verzii mojej aplikácie bola grafická časť vytváraná pomocou Canvasu. Dáta o schéme boli ukladané do databázy. Prvou zmenou, ktorá ovplyvnila vývoj aplikácii bol prechod na knižnicu Raphaël, no vzhľadom na nedostatok dokumentácií k tejto knižnici som zmenila Raphaël za Fabric, ktorý má omnoho viac funkcií potrebných pre aplikáciu. Ďalšou zmenou bolo ukladanie dát. Dáta sú v súčasnej verzii ukladané do JSON-u.

V roku 2010 bola na Fakulte elektrotechniky a informatiky vyvinutá podobná aplikácia, ktorá nie je verejne dostupná. Aplikácia umožňovala zostrojenie simulinkovej

schémy online. Výstupom z tejto aplikácie bol súbor mdl. Vzhľadom na to, že v roku 2012 MATLAB inovoval formát ukladania súboru z mdl na SLX, sa dá považovať táto aplikácia za zastaralú. [6], [7]

Hlavnou úlohou mojej práce je vytvoriť aplikáciu pre tvorbu simulinkových schém, ktoré budú generované v súbore typu SLX. Po preštudovaní už vzniknutých programov a porovnaní s návrhom simulinkových schém som zistila, že nie je dostupný nástroj, ktorý plnohodnotne nahradí grafické rozhranie Simulink-u.

3 Analýza problému

Táto časť je zameraná na opis štruktúry uloženej schémy programu Simulink a rozdiely medzi jednotlivými verziami programu Simulink vo verziách MATLAB-u od verzie R2012a po verziu R2014a.

3.1 Simulink/MATLAB

Simulink je súčasť programu MATLAB, ktorý je využívaný k simulácii dynamických systémov. Simulink je možné spustiť jednoducho z príkazového riadku príkazom `simulink`. Užívateľovi je zobrazená knižnica blokov. Bloky je možné jednoducho vložiť pomocou presúvania z knižnice do okna nového modelu.

Schémy, ktoré sú vytvárané v Simulink-u od verzie MATLAB-u R2012a, sú ukladané vo formáte SLX. Tento formát je komprimovaný súbor pozostávajúci z XML súborov a obrázkov vo formáte PNG [8].

3.1.1 Parametre schémy

Parametre schémy sú uložené v `blockdiagram.xml`. V tomto súbore sú zapísané všetky údaje o vytvorenej schéme. Všetky dáta sú umiestnené do elementu `Model`, ktorý sa nachádza v koreňovom elemente `ModelInformation`. Obsahuje údaje o nastavení editora, simulačné nastavenia, údaje o prednastavených parametroch blokov, o nastavení blokov a iné.

Každý blok má svoje prednastavené parametre a parametre nastaviteľné užívateľom. Hodnoty parametrov sú vkladané do elementov `p`. Elementy `p` obsahujú atribúty `Name`, ktorých hodnota je názov parametra. Ak schéma neobsahuje žiadne bloky, prednastavené parametre nie sú vložené do `blockdiagram.xml`. V kóde 1 je zobrazená časť zápisu prázdnej schémy.

```
<BlockParameterDefaults>
</BlockParameterDefaults>
<System>
  <P Name="Location">[100, 100, 900, 600]</P>
  <P Name="Open">on</P>
```

```
  <P Name="ModelBrowserWidth">200</P>
  <P Name="TiledPaperMargins">
    [1.270000, 1.270000, 1.270000, 1.270000]</P>
  <P Name="TiledPageScale">1</P>
  <P Name="ZoomFactor">100</P>
  <P Name="ReportName">simulink-default.rpt</P>
</System>
```

Kód 1: Prázdna schéma

Knižnica blokov v Simulink-u obsahuje množstvo blokov, ktoré umožňujú vytvárať schémy. Diplomová práca je zameraná na vývoj aplikácie, ktorá dokáže vytvoriť jednoduchú blokovú schému, ktorú je následne možné vygenerovať v súbore SLX. Vygenerovaná schéma musí byť spustiteľná v programe Simulink.

Pre nasledujúce názorné ukážky bude využívaný blok `Step`.

3.1.2 Pridanie bloku do simulinkovej schémy

Ak by bol do schémy vložený blok typu `Step`, do elementu `BlockParameterDefaults` sa zapíšu jeho preddefinované hodnoty (kód 2).

```
<Block BlockType="Step">
  <P Name="Time">1</P>
  <P Name="Before">0</P>
  <P Name="After">1</P>
  <P Name="SampleTime">-1</P>
  <P Name="VectorParams1D">on</P>
  <P Name="ZeroCross">on</P>
</Block>
```

Kód 2: Pridanie bloku `Step` do elementu `BlockParameterDefaults`

Tieto hodnoty nebudú duplikované po vložení ďalšieho bloku `Step` do schémy.

Súčasne sa do elementu `System` zapíšu informácie o pozícii, z-súradnici a parametri `"Sample Time"` (kód 3). Tieto dáta môžu byť opätovným presúvaním bloku, či zmenou parametrov prepísané. Po vložení viacerých blokov typu `Step`, budú do XML pridané

viaceré elementy `Block` s atribútom `BlockType` s hodnotou „Step“, ale rôznou hodnotou atribútu `Name` a `SID`.

```
<Block BlockType="Step" Name="Step" SID="1">
  <P Name="Position">[90, 90, 120, 120]</P>
  <P Name="ZOrder">1</P>
  <P Name="SampleTime">0</P>
</Block>
```

Kód 3: Pridanie bloku `Step` do elementu `System`

Ak by boli zmenené parametre bloku `Step`, boli by vložené ako ďalší riadok v elemente `Block` (kód 4). Preddefinované hodnoty zostávajú nezmenené.

```
...
  <P Name="Before">2</P>
  <P Name="After">10</P>
...
```

Kód 4: Zmenenie preddefinovaných parametrov bloku `Step`

3.1.3 Pridanie spojenia do simulinkovej schémy

Po spojení blokov pribudne okrem blokov v elemente `System` element `Line` (kód 5).

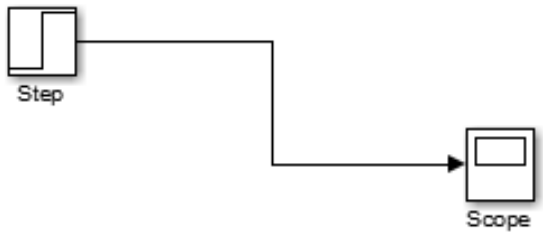
```
<Line>
  <P Name="ZOrder">2</P>
  <P Name="Src">1#out:1</P>
  <P Name="Dst">2#in:1</P>
</Line>
```

Kód 5: Pridanie spojenia do elementu `System`

Toto spojenie hovorí, že je to druhá vytvorená čiara v schéme, ktorá spája zdrojový blok, ktorý bol pridaný ako prvý do schémy a cieľový blok , ktorý bol pridaný ako druhý v poradí. Okrem toho je možné z kódu zistiť, z ktorého výstupu spojenie vychádza, a

do ktorého vstupu spojenie vchádza. Toto spojenie je tvorené rovnou čiarou, pretože sú bloky v schéme umiestnené v jednej rovine.

Ak je jeden z blokov umiestnený vyššie alebo nižšie (obr. 2), čiara v Simulink-u je vygenerovaná s hranami. Komplikovanejšie spojenie, to znamená viac hrán, je v prípade, ak cieľový blok má x-ovú súradnicu menšiu než zdrojový blok, teda je umiestnený naľavo od zdrojového bloku. Simulink sa snaží spájať bloky tak, aby spojenie neprechádzalo blokom.



Obr. 2: Simulink schéma - spojenie

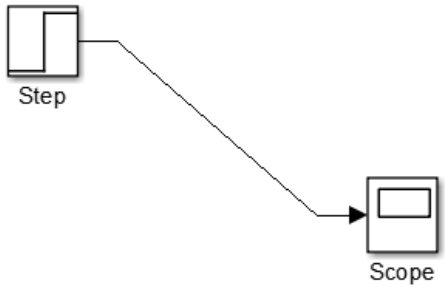
Do elementu `<p>` (s atribútom `Name`=”Points”) sú zapísané relatívne súradnice zlomov (hrán) vzhľadom na predchádzajúci zlom.

V kóde 6 má prvý zlom súradnice [83, 0], vzhľadom na bod, z ktorého spojenie vychádza. Zmenila sa pozícia len v smere x, preto [83, 0]. Druhý zlom je v [0, 55]. Vzhľadom k predošlému bodu [83, 0] je zmenená len súradnica v smere y. Keď Simulink vygeneruje spojenie s viacerými zlomami, do elementu `p` (s atribútom `Name`=”Points”) pribudnú súradnice ďalších zlomov.

```
<Line>
  <P Name="ZOrder">1</P>
  <P Name="Src">1#out:1</P>
  <P Name="Points">[83, 0; 0, 55]</P>
  <P Name="Dst">2#in:1</P>
</Line>
```

Kód 6: Pridanie spojenia so zlomami do elementu `System`

Simulink umožňuje takto vzniknutému spojeniu upravovať jeho tvar. Pre diagonálne spojenie je potrebné kurzor myši nastaviť nad zlom, ktorým chceme pohybovať, stlačiť klávesu SHIFT so stlačeným ľavým tlačidlom myši a posunúť kurzor nad miesto, kam chceme zlom presunúť. Príklad diagonálneho spojenia je na obr. 3.



Obr. 3: Zmena tvaru spojenia

3.2 Porovnanie verzií Simulink-u

V tejto kapitole sú porovnávané súbory `blockdiagram.xml`, ktoré sú vygenerované od verzie programu MATLAB R2012a až po R2014a. Súbor `blockdiagram.xml` môžme rozdeliť na dve časti. Prvá časť opisuje nastavenie editora, simulačných vlastností a pod. a druhá časť opisuje na údaje o vložených blokoch.

Porovnávané sú vždy dve po sebe nasledujúce verzie. Rozdiely v týchto súboroch sú minimálne. V prvej časti sa líšia len verziami MATLAB-u a každá vyššia verzia MATLAB-u obsahuje viac nastavení schémy. Druhá časť sa zmenila vo verzii R2013b. Pri spojeniach (elementoch `<line>`) sa objavuje parameter `ZOrder`. Predchádzajúce verzie obsahovali `ZOrder` len pri blokoch.

V práci je sústredená pozornosť na druhú časť súboru, to znamená na údaje o nastavení blokov. Preto sú zanedbané rozdiely v prvej časti súboru, ktoré vznikli vývojom novších verzií MATLAB-u.

Online aplikácia pracuje s verziou MATLAB R2014a. Vo virtuálnych laboratóriách je MATLAB umiestnený na serveri. Aby sa predišlo komplikáciám, vygenerovaná schéma by mala byť otváraná v takej verzii, aký je MATLAB na serveri.

4 Použité technológie

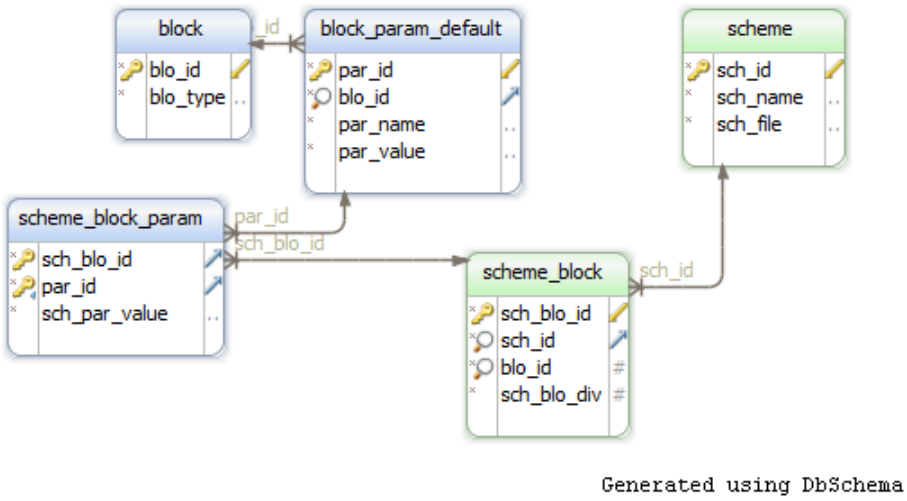
V tejto kapitole uvádzam dostupné riešenia pre tvorbu simulinkovej schémy. Zaoberám sa riešením grafickej časti, možnosťami uloženia dát, využitými technológiami a jazykmi.

4.1 Spôsob ukladania dát

Počas vytvárania schémy, je potrebné zachovávať dáta o pridaných blokoch a nastavených parametroch. To zahŕňa názov bloku, pozíciu, poradie bloku a už samotné vlastnosti, ktoré ponúka Simulink, ako napríklad pri bloku `Step` je to nastavenie hodnoty Before, After, Sample Time, atď. Počas vývoja aplikácie som sa zaoberala dvomi spôsobmi ukladania dát.

MySQL databáza

V prvej verzii aplikácie, ktorú som vyvíjala v semestrálnej práci, boli tieto dáta ukladané do databázy. Bolo vytvorených niekoľko tabuliek, do ktorých sa ukladali dané parametre. ERD diagram je na obr. 4.



Obr. 4: Entitno-relačný diagram

Počas diplomovej práce som databázu nahradila za JSON.

JSON

JSON (JavaScript Object Notation) je spôsob zápisu dátového formátu, ktorý je určený pre prenos dát. Tento formát je vybraný na základe toho, že v práci je využitý JavaScript, a pretože dáta potrebné pre Simulink majú byt' vo formáte XML. JSON je považovaný za odľahčenú a jednoduchšiu alternatívu k XML. JSON je podrobnejšie opísaný v podkapitole 4.5.8. [9]

4.2 Grafická časť

V rámci HTML stránky sú najznámejšie a najpoužívanejšie technológie pre vykresľovanie grafických prvkov Canvas a SVG. HTML5 Canvas ako aj SVG sú technológie, ktoré umožňujú vložiť do HTML stránok grafické prvky, ale sú od seba od základov odlišné.

V prvej verzii bol využitý HTML Canvas, ale len čiastočne, pretože knižnica blokov v aplikácii pozostávala z PNG obrázkov a Canvas bol využitý len na vykresľovanie spojení medzi blokmi. Obrázky sa do Canvasu pridávali ako klony originálov pomocou metódy `drag&drop` z knižnice jQuery UI. Nakoľko by bolo potrebné pri pridávaní d'alších blokov ich obrázok v PNG, rozhodla som pre dynamické vykresľovanie blokov. Pri dynamickom generovaní knižnice je potrebné vyriešiť otázku, či požiadavkám aplikácie postačuje Canvas alebo je potrebné SVG.

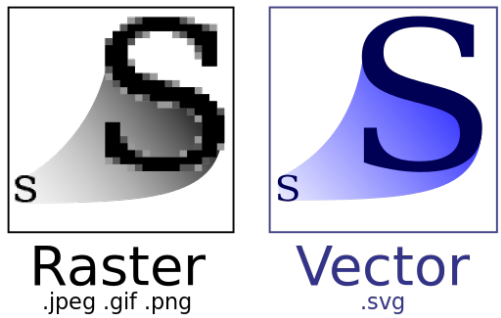
4.2.1 HTML5 Canvas

Canvas je súčast'ou HTML5. Využíva sa na web stránkach pomocou skriptovania väčšinou za pomoci jazyka JavaScript. Canvas vytvára bitmapový obrázok, ktorý je definovaný jednotlivými pixelmi, ktoré sú jednoznačne určené ich polohou a farbou (RGB). V podstate sa dá Canvas nazvat' ako plátno, kde sa pridávajú grafické prvky, ale po ich prekrytí dochádza k zmazaniu časti prekrytia prekrytého prvku. [10]

4.2.2 SVG

SVG (Scalable Vector Graphics) je značkovacím jazykom a používa sa na definovanie vektorovej grafiky. Pre definovanie grafických prvkov sa využíva formát XML. Grafika SVG neobsahuje obraz pixel po pixely, ale obsahuje zoznam súčastí - teda grafických objektov, pomocou ktorých je obrázok vykreslený. Veľkou výhodou SVG prvkov je, že nestrácajú pri zmene veľkosti kvalitu a každý prvok v SVG môže byť animovaný. [11]

Porovnanie bitmapového obrázku a svg obrázku je na obr. 5.



Obr. 5: Porovnanie kvality obrázkov [2]

4.3 Použité knižnice

Ako bolo spomínané v podkapitole 2, existuje množstvo knižníc podporujúcich kreslenie grafických prvkov. Počas vývoja aplikácie som sa stretla s viacerými, ale množstvo z nich nespĺňalo požiadavky aplikácie. V nasledujúcich podkapitolách uvediem tie knižnice, ktoré boli využité pri vývoji.

4.3.1 Knižnica Raphaël

Raphaël je knižnica JavaScriptu, ktorá zjednodušuje prácu s vektorovou grafikou na webe. Táto knižnica dokáže pracovať s vytvoreným obrázkom ako napríklad ho otáčať či orezávať. Každý grafický objekt, ktorý je vytvorený, je tiež DOM objektom, takže je možné využiť JavaScript udalosti alebo modifikovať grafický prvok neskôr. Raphaël v súčasnosti podporuje: Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+, Internet Explorer 6.0+. [12]

Aplikácia bola vyvíjaná pomocou knižnice Raphaël. Dokumentácia k Raphaëlu je od februára 2016 nedostupná na oficiálnej stránke, čo viedlo ku knižnici Fabric. Táto knižnica spĺňa požiadavky aplikácie a má možnosť taktiež pracovať s vektorovou grafikou.

4.3.2 Knižnica Fabric

V aktuálnej verzii online aplikácie je využitá knižnica Fabric.

Fabric je výkonná a jednoduchá javascriptová knižnica, ktorá umožňuje vytvárať a manipulovať s objektami v Canvas plátne. Obsahuje tiež funkcie na prevod z Canvasu do SVG a naopak, ako aj funkciu do JSON. [13]

Fabric ponúka [13]:

- jednoduché vkladanie objektov do plátna. Napríklad matematické tvary (čiara, obdĺžnik, trojuholník, kruh, ...), obrázky, text, cesty a iné. Všetky objekty sú pridávané príkazom `add(Objekt)` do plátna. Typy objektov sú vytvárané špecifickým príkazom. Pre názornú ukážku uvediem najčastejšie používaný objekt v práci, ktorým je obdĺžnik. Príkaz: `new fabric.Rect();`
- nastavenie vlastností pridávaným objektom. Pozíciu v rámci Canvasu, zväčšovanie, zmenšovanie, označovateľnosť, atď. Príkazy sú zadávané v tvare:
  vlastnosť : hodnota. Hodnoty môžu byť definované ako čísla, reťazce, logické dátové typ , polia a objekty;
- zoskupovanie objektov do skupín a so skupinou pracovať ako s objektom.

4.4 Architektúra aplikácie

Aplikácia je postavená na modeli tzv. dvojvrstvovej architektúry (klient-server). Je to softvérová architektúra, ktorá je zložená z dvoch vrstiev: prezentačnej (klientskej) a aplikačnej vrstvy, ktorá obsahuje funkčnú/procesnú logiku (obr. 6). Jednotlivé vrstvy plnia samostatné úlohy a je možné ich vyvíjať, udržiavať a meniť nezávisle, za predpokladu zachovania nemenného rozhrania medzi jednotlivými vrstvami. [14]

- Prezentačná/klientská vrstva - vrstva užívateľského rozhrania slúži na interakciu s obsluhou – zberom informácií a naopak aj ich zobrazenie užívateľovi.
  Táto vrstva zahrňuje:
  - informačnú architektúru
  - užívateľské rozhranie, ktoré je optimalizované pre rozličné verzie cieľového zariadenia, najčastejšie v podobe HTML/CSS/JavaScript kódu
  - podporu pre užívateľské interakcie ako je vkladanie dát a ich zobrazovanie
  - funkcionalitu z hľadiska SEO
  - rozhranie umožňujúce integráciu v štrukturovanej podobe – formy XML výstupu a iné
- Aplikačná vrstva – zabezpečuje zber dát, ich transformáciu a distribúciu informácií.

4.5 Prezentačná/klientská vrstva

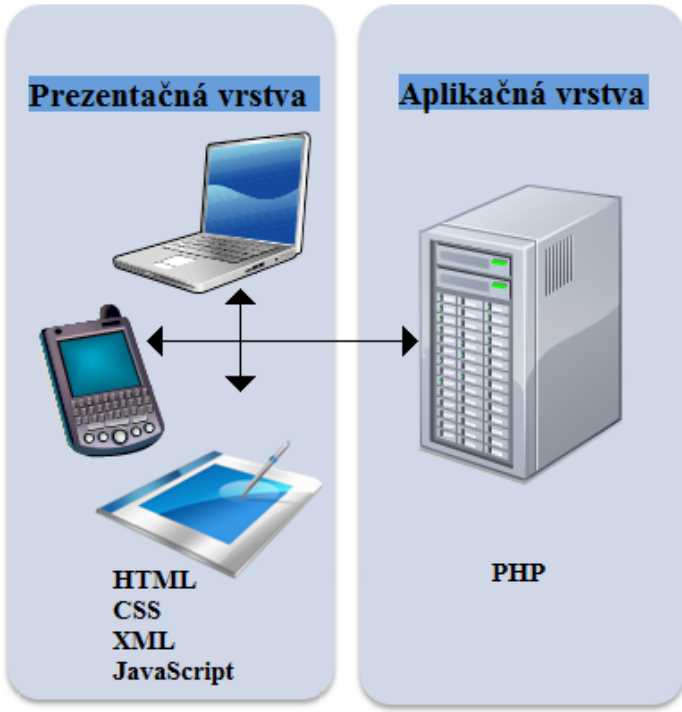
Na klientskej vrstve je spustená aplikácia, teda stránka, ktorá je tvorená pomocou jazykov HTML, CSS a JavaScript a pomocou knižníc jQuery a jQueryUI. Zabezpečuje užívateľské rozhranie. Táto vrstva tiež zabezpečuje zobrazenie výsledkov z aplikačnej vrstvy. Do tejto vrstvy by som zaradila aj AJAX, nakoľko je založený na JavaScript-e a XML. [14]

4.5.1 XML

[15], [16]

Simulink ukladá všetky dáta o schéme do formátu SLX. Ako bolo spomenuté v podkapitole 3.1, je to komprimovaný súbor obsahujúci podpriečinky so súbormi XML. V online aplikácii Simulink-u je potrebné uložiť parametre o blokoch do súboru XML s rovnakou štruktúrou, akou ukladá Simulink.

XML (eXtensible Markup Language) je značkovací jazyk, ktorý bol vyvinutý a štandardizovaný konzorciom W3C (World Wide Web Consortium). Tento jazyk je textový formát zápisu ľubovoľných dát, ktorý je veľmi ľahko čitateľný.



Obr. 6: Dvojvrstvová architektúra

Základná štruktúra XML

Dokumenty XML sú zložené z kombinácií značiek a znakov. Znaky v XML dokumente predstavujú dáta, zatiaľ čo značky tvoria štruktúru XML dokumentu. Ak XML spĺňa isté pravidlá, potom je považované za správne štrukturované (well-formed).

1. Elementy sú párovými značkami, preto musia mať začiatočnú aj koncovú značku. Existuje aj skrátený zápis, ktorý je použitý v prípade, keď obsahom elementu nie sú žiadne dáta. Namiesto `<empty></empty>` je možné použiť `<empty />`.

2. Dokument XML musí obsahovať koreňový element dokumentu, ktorého obsahom sú všetky ostatné elementy. Tieto elementy sú doň vnorené.

3. Každý vnorený element musí mať začiatočnú aj koncovú značku obsiahnutú vo svojom nadradenom elemente. Tým je zaistená hierarchická štruktúra dokumentu XML.

Jednoduchý príklad správne zapísaného XML súboru je uvedený v kóde 7.

```
<korenovy_element>
  <vnoreny_element>
    Priklad XML
  </vnoreny_element>
</korenovy_element>
```

Kód 7: Štruktúra XML súboru

Ak je dokument XML napísaný podľa týchto pravidiel, tak má správnu štruktúru. Dokument XML je označovaný za platný, keď je správne štruktúrovaný a spĺňa požiadavky DTD.

DTD (Document Type Definition), po slovensky definícia typu dokumentu, je jazyk pre opísanie štruktúry XML. DTD napríklad vymedzuje jazyky HTML a XHTML.

4.5.2 HTML

Užívateľská časť aplikácie je tvorená v jazyku HTML.

HTML (HyperText Markup Language) je textový značkový jazyk, používaný na tvorbu základu webových stránok, ale aj iných informácií, ktoré sú zobrazované vo webovom prehliadači.

Pomocou jazyka HTML sú vytvárané dokumenty, ktoré môžu obsahovať text, odkazy, modálne okná, obrázky, tabuľky a iné. [17]

Základné parametre HTML stránky

HTML stránka je tvorená kombináciou textu a značiek. Značky HTML sú párové a nepárové. Párové značky sa využívajú pri zobrazení objektu a nepárové značky sa zavádzajú v mieste, kde je objekt volaný. Všetky značky sú zapisované do lomených zátvoriek <>. Je to z dôvodu odlíšenia značiek od textu. [18]

Základná štruktúra HTML stránky

Základná štruktúra HTML stránky pozostáva z hlavičky a tela dokumentu. Na začiatku dokumentu HTML by mala byť zadefinovaná verzia jazyka HTML. Koreňový element

je tvorený párovou značkou `<html>` a `</html>`, ktorá vymedzuje celý HTML dokument. Hlavička je tvorená párovými značkami `<head>` a `</head>`. Obsahuje metadata, ktoré sa vzťahujú na celý dokument. Metadata nie sú zobrazované užívateľovi. Hlavička môže obsahovať odkazy na externé súbory. Napríklad odkaz na kaskádové štýly. Kaskádové štýly môžu byť definované aj priamo v hlavičke. Telo dokumentu je reprezentované značkami `<body>` a `</body>`, medzi ktorými je celý obsah stránky. [18]

Príklad základnej štruktúry HTML 5 dokumentu je v kóde 8.

```
<!DOCTYPE html>
<html lang="sk">
  <head>
    <meta charset="utf-8">
    <style type="text/css">
      Definovanie stylov
    </style>
    <title>Titulok</title>
  </head>
  <body>
    Telo dokumentu
  </body>
</html>
```

Kód 8: Štruktúra HTML súboru

4.5.3 CSS

Kaskádové štýly (Cascading Style Sheets) rozširujú vlastnosti HTML o vizuálne formátovanie. Pomocou CSS sú vytvárané štrukturované dokumenty, čo znamená, že obsah je oddelený od vzhľadu. CSS je samostatný súbor, ktorého odkaz je umiestnený v hlavičke HTML dokumentu. Výhodou oddeleného CSS je menšia dátová veľkosť a veľmi jednoduchá zmena celkového dizajnu stránky. [19]

4.5.4 JavaScript

JavaScript je interpretovaný programovací jazyk, ktorý má základné objektovo orientované schopnosti. JavaScript beží na strane klienta, čo znamená, že všetky aplikácie sú spustené vo webovom prehliadači u užívateľa. Toto je veľkým rozdielom oproti serverovým jazykom, ktoré sú vykonávané na serveri a klientovi sú odosielané len výsledky.

Pomocou JavaScriptu je možné pridať na statickú web stránku dynamické prvky, ktoré komunikujú s užívateľom, riadia prehliadač a dynamicky vytvárajú obsah HTML stránky. [20], [21]

4.5.5 jQuery

jQuery je javascriptová knižnica, v ktorej je kladený dôraz na interakciu medzi JavaScriptom a HTML. jQuery obvykle existuje ako jeden javascriptový súbor, ktorý obsahuje všetky funkcie pre DOM, Ajax, udalosti a efekty. Tento súbor je možné jednoducho vložiť ako odkaz do HTML kódu. [22]

4.5.6 jQueryUI

JQuery UI je javascriptová knižnica, ktorá je zameraná na rozhranie pre klienta. Bola vytvorená s cieľom uľahčenia implementácie efektov a funkcií HTML prvkov na webových stránkach. Má podporu u širokého spektra prehliadačov. Túto knižnicu je možné jednoducho stiahnuť s voliteľnými súčasťami.
jQuery UI je možné rozdeliť do 4 základných častí [23]:

- Core - jadro knižnice, ktoré sa skladá z jednotlivých modulov: core, widget, mouse, position
- Interactions - metódy, ktoré sú implementované k interakciám medzi klientom a webovým rozhraním. Týmito metódami sú draggable, droppable, resizable, selectable a sortable.
- Widgets - obsahuje pokročilejšie elementy. Patrí sem napr.: button, slider, dialog, menu, atď.
- Effects - grafické efekty ako show, hide, add class, remove class, effect.

V aplikácii sú využívané efekty a funkcie knižnice jQueryUI. V praktickej časti sa nachádza ich presnejší opis.

4.5.7 AJAX

AJAX (Asynchronous JavaScript and XML) je obecne označovaná technológia vývoja interaktívnych webových aplikácií. Táto technológia umožňuje meniť obsah celej web stránky bez nutnosti znovunačítavania za pomoci javaskriptovej knižnice. [24]

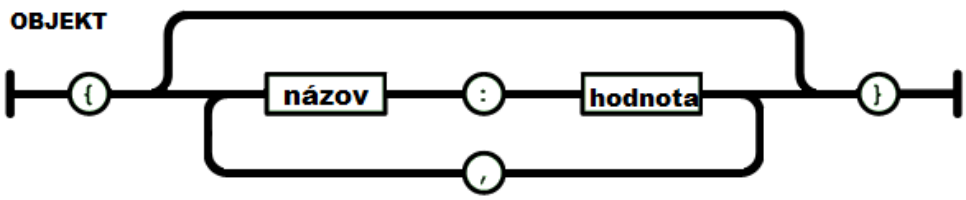
4.5.8 JSON

[9]

JSON (JavaScript Object Notation) je formát pre výmenu dát na webe. JSON sa objavil v čase, keď sa pre výmenu dát začal využívať jazyk XML. Nakoľko práca s ním bola zložitá, vyvinul sa JSON. Pri použití krátkych štrukturovaných dát, ktoré sú posielané webovými aplikáciami je používaný JSON. JSON je ľahko čitateľný, zrozumiteľný i zapisovateľný človekom.

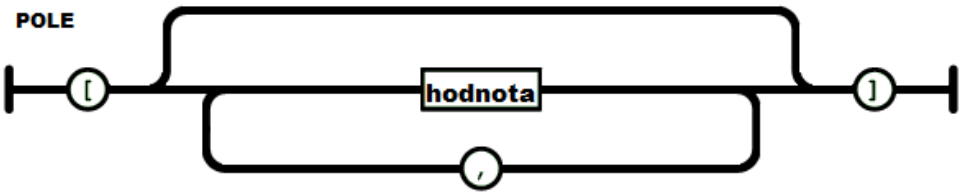
Zápis JSON je zápisom javascriptového jazyku. Podporuje dve štruktúry:

1. Štruktúru, ktorá reprezentuje objekt. Objekt je chápaný ako množina párov: názov a hodnota. Hodnota je uzatvorená úvodzovkami.



Obr. 7: JSON - objekt

2. Štruktúru, ktorá reprezentuje pole.



Obr. 8: JSON - pole

Do formátu JSON je možné zapisovať nasledujúce typy dát:

- String – textový reťazec
- Number – číslo (celočíselné/reálne)
- Boolean – logická hodnota
- Null – hodnota null
- Array – pole
- Object – objekt

Príklady formátu JSON

Prvým príkladom je objekt a opisuje jednu hodnotu formulára bloku `Step`.

```
{
    "title": "Step time",
    "name": "Time",
    "size": "20",
    "id": "Time",
    "value": "0"
}
```

Kód 9: Štruktúra objektu

Druhým príkladom je pole. Tento príklad je uvedený z bloku `Scope` a opisuje možnosti výberového zoznamu.

```
[
    {
        "value": "on",
        "name": "all"
    },
    {
        "value": "off",
        "name": "none"
    },
    ...
]
```

Kód 10: Štruktúra poľa

4.6 Aplikačná vrstva

Aplikačná vrstva je kľúčovou z pohľadu ukladania a správy dát. Mala by byť zvolená silná platforma umožňujúca zber, transformáciu a následnú distribúciu informácií. Hlavnou úlohou je prijímať požiadavky klienta a po spracovaní odoslať požadované informácie naspäť klientovi, ktorý ich zobrazí používateľovi. Rozdielom medzi klientom a serverom je, že server musí byť spustený neustále, pretože klient môže odosielať požiadavky v rôznom čase. V online aplikácii Simulink-u je využívaný na tejto vrstve jazyk PHP. [14]

4.6.1 Jazyk PHP

PHP (Hypertext Preprocessor pôvodne Personal Home Page) je skriptovací programovací jazyk. Tento jazyk je určený predovšetkým pre programovanie dynamických stránok a webových aplikácií. Pri použití PHP sú skripty spracované na strane serveru a užívateľovi je odoslaný len výstup.

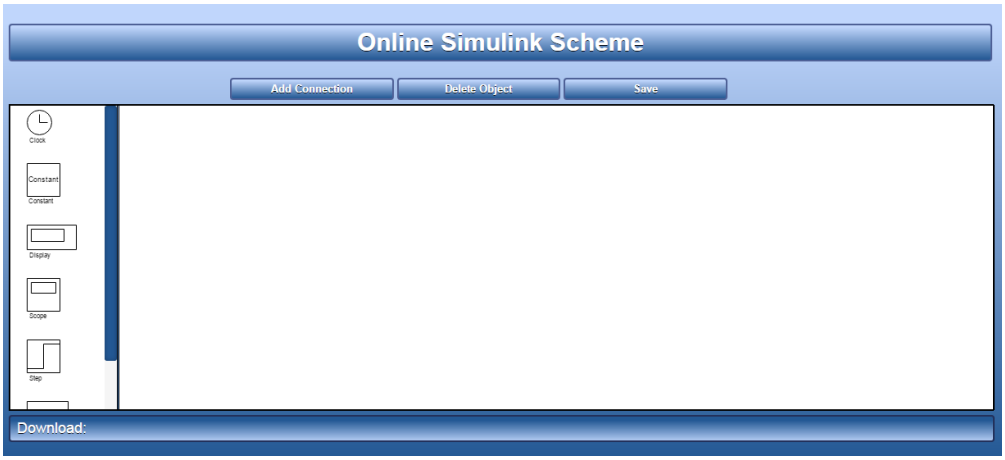
PHP podporuje množstvo knižníc. V práci je PHP využívané pri odosielaní dát na server, ktorý naspäť posiela súbor SLX, ktorý je možné stiahnuť. [25]

5 Praktická časť

V praktickej časti sú opísané užívateľské možnosti aplikácie, činnosti, ktoré sa vykonávajú v pozadí a opis pridania nového bloku do knižnice blokov krok po kroku. Ďalej tu sú vysvetlené a opísané prvky, funkcie, použité metódy a efekty z technológií a knižníc, ktoré sú v práci využité.

5.1 Opis aplikácie

Po spustení aplikácie je užívateľovi zobrazená stránka, na ktorej sa nachádza vytvorená aplikácia pre tvorbu schém. Je v nej umiestnená knižnica blokov (ľavá časť) a plátno (pravá časť), ktoré je určené pre tvorbu schémy (obr. 9). Nad plátnom sú umiestnené tlačidlá, ktoré umožňujú užívateľovi vytvoriť spojenie medzi dvomi blokmi, zmazať objekt alebo vygenerovať schému vo formáte SLX. Vygenerovaný súbor je zobrazený užívateľovi pod schémou ako odkaz na dokument, ktorý je možné stiahnuť.



Obr. 9: Online aplikácia Simulink

5.1.1 Knižnica blokov

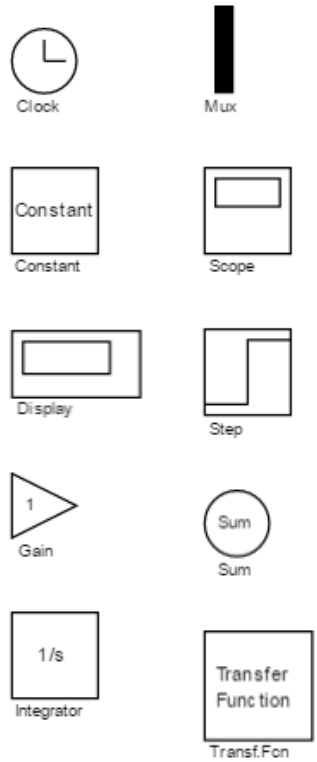
Aplikácia obsahuje adresár `library`. V tomto adresári sa nachádzajú podadresáre, ktoré predstavujú jednotlivé bloky. Každý podadresár obsahuje štyri súbory. Dva súbory sú typu JSON a dva sú typu XML.

Súbor s parametrami potrebnými pre vykreslenie bloku, je nazvaný `display.json`. Druhý súbor obsahuje parametre, ktoré slúžia na vytváranie dynamického formulára pre každý typ bloku. Tento súbor je nazvaný `param.json`.

Druhé dva súbory sú vkladané do výstupného XML. Prvý súbor, `dafault.xml`, obsahuje dáta, ktoré sa zapisujú do elementu `BlockParameterDefaults` v súbore `blockdiagram.xml`. Druhý súbor, `system.xml`, obsahuje dáta, ktoré sa zapisujú do elementu `System` v `blockdiagram.xml`. Formát výstupného súboru, `blockdiagram.xml` je opísaný v podkapitole 3.1.

Ak by užívateľ chcel pridať nový typ bloku do knižnice blokov, postupoval by podľa návodu v podkapitole 5.2.

V aplikácii sa momentálne nachádzajú nasledovné bloky:



Obr. 10: Aktuálna knižnica blokov v aplikácii

5.1.2 Generovanie knižnice

Knižnica blokov je dynamicky generovaná pri spustení stránky. Grafická časť každého bloku je zadefinovaná v externom súbore `display.json`.

Vlastnosti každého bloku majú v súbore rovnakú štruktúru. Koreňový element je typ bloku (Step, Scope, ...). Jednotlivé tvary, z ktorých je blok zložený, sú súčasťou poľa a odlišujú sa typom objektu. V kóde 11 je uvedená časť kódu pre zobrazenie bloku typu `Constant`. Koreňovým elementom je názov `Constant`, ktorý obsahuje pole objektov. Sú od seba odlíšené typom tvaru, ktorý je potrebné vykresliť. Blok typu `Constant` je zložený z obdĺžnika a textu. Obdĺžnik, ktorý má vlastnosti zadefinované v objekte `data`, bude mať rozmery 40×40, bude vyplnený bielou farbou, jeho ľavý roh bude mať v Canvase súradnice [50, 100] a rám obdĺžnika bude čiernej farby.

```
{
"Constant":
  [{
  "type": "rect",
  "data":
      {
      "width": 40,
      "height": 40,
      "fill": "white",
      "left": 50,
      "top": 100,
      "stroke": "black"
      }
  },
  ...
  ]
}
```

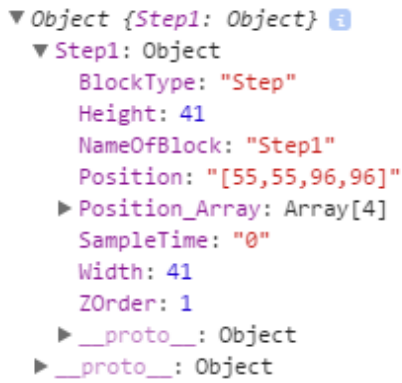
Kód 11: Kód pre vykreslenie časti bloku `Constant`

Takto sú postupne pridané do `display.json` aj ostatné časti, ktoré sú potrebné pre vykreslenie celého bloku. Viac o vykresľovaní tvarov a textu v podkapitole 5.2.

5.1.3 Pridanie bloku do schémy

Blok z knižnice blokov je ľahko pridaný do schémy pomocou dvojkliku na ľavé tlačidlo myši. Blok je automaticky vložený do schémy, kde má užívateľ možnosť s ním pohybovať, spájať s iným blokom, meniť mu vlastnosti, či zmazať ho.

Keď užívateľ pridá blok z knižnice blokov do schémy, v pozadí aplikácie je do premennej `scheme` pridaný nový objekt. Názov objektu je tvorený typom bloku a číslom. Číslo je poradie pridaného bloku do schémy. Tento objekt obsahuje informácie o vloženom bloku (obr. 11).



Obr. 11: Pridanie bloku `Step` do premennej `scheme`

5.1.4 Nastavenie parametrov bloku - modálne okno

Súčasťou aplikácie je možnosť nastavenia parametrov bloku. Po dvojkliku na blok v Simulink-u sa nám otvára okno s týmito možnosťami. Rovnako je to aj v aplikácii. Je na to využívaný tzv. widget - dialog(), ktorý je súčasťou knižnice jQueryUI. V nasledujúcom kóde je zobrazená udalosť dvojkliku myši. Táto udalosť spúšťa widget dialog, ktorý otvorí dialógové okno nad pôvodným obsahom stránky (kód 12).

```
block.on('object:dblclick', function (options) {
  $("#dialog").html(result)
  $("#dialog").dialog("open");})
```

Kód 12: Zobrazenie dialógového okna

Obsah dialógového okna je formulár napísaný v jazyku HTML. Tento obsah je tvorený dynamicky na základe externého súbora `param.json`, ktorý je zadefinovaný pre každý typ bloku. Štruktúra je podobná ako v prípade zobrazenia bloku. Koreňový element je názov bloku a obsahuje pole objektov. Prvý objekt poľa opisuje blok všeobecne, kým ostatné opisujú jednotlivé položky formulára. Môžu to byť textové polia, zaškrtávacie polia (checkbox) alebo výberové polia (selectbox). Pre názornú ukážku uvádzam časť kódu pre formulár bloku `Constant`, konkrétne pre prvý objekt a pre jeden textový prvok v jeho formulári (kód 13).

```
{
  "io": "out",
  "NumberOfInputs": 0,
  "NumberOfOutputs": 1,
  "BlockType" : "Constant",
  "title": "Source Block Parameters Constant"
},
{
  "type": "text",
  "data":
  {
    "title": "Constant Value",
    "name": "Value",
    "size": "20",
    "id": "Value",
    "value": "0"
  }
}
```

Kód 13: Časť kódu pre formulár bloku `Constant`

Prvý objekt je tvorený všeobecnými informáciami o bloku. Tento konkrétny blok je výstupný blok, ktorý nemá žiadne vstupy a jeden výstup. Uvedený je tu aj jeho typ a nadpis, ktorý je použitý pri vypisovaní.

V kóde je ďalej uvedené textové pole, ktorého nadpis vo formulári opisuje textové pole, `name`, `size` a `id` sú atribúty značky input. Nakoniec obsahuje samotnú hodnotu,

ktorá je zobrazená vo formulári ako prvá, preddefinovaná.

Dialógové okno, v ktorom je zobrazený formulár parametrov, má svoje nastaviteľné vlastnosti, metódy a udalosti. Jedným z efektov je napríklad možnosť modálneho okna. Pri otvorení tohoto okna užívateľ stráca možnosť kliknutia mimo modálneho okna. To znamená, kým toto okno nebude zatvorené, nemôže vykonávať iné zmeny v schéme. Modálnemu oknu je možné nastaviť aj iné vlastnosti, ako napríklad: jeho šírku, výšku, v hlavičke okna nadpis, možnosť zväčšovania a zmenšovania okna a iné. Do modálneho okna je možné pridať tlačidlá.

```
$('#dialog').dialog({
  modal: 'true',
  resizable: true,
  autoOpen: false,
  width: "auto",
  height: "auto",
  title: 'Source Block Parameters '+type ,
  buttons: {
    "Submit": addParam,
    "Close": function() {
    $(this).dialog( "close" );
    }
});
```

Kód 14: Nastavenie dialógového okna

V modálnom okne sa nachádzajú dve tlačidlá. Tlačidlo „Close“ a „Submit“. Po stlačení tlačidla „Submit“ sa vykoná funkcia `addParam` a zatvorí dialógové okno. Tlačidlom „Close“ sa zatvorí dialógové okno bez uloženia.

Funkcia `addParam` slúži na pridávanie parametrov do objektu v premennej `scheme`. V `scheme` sa nájde označený blok (objekt) a v ňom je hľadaný názov pridávanej vlastnosti. Ak v objekte neexistuje názov vlastnosti, táto vlastnosť je pridaná aj s hodnotou. Ak existuje, je prepísaná jej hodnota hodnotou z formulára. Po odoslaní údajov je modálne okno zatvorené a dáta sú uložené v premennej `scheme`.

Po otvorení modálneho okna a prvom odoslaní dát, je do objektu v premennej `scheme` pridaná vlastnosť „Reload“ s hodnotou `Yes` (kód 15). Vlastnosť „Reload“ slúži pri znovuotvorení modálneho okna k tomu, aby sa parametre nenačítavali z `param.json`, ale už z premennej `scheme`. Je to ošetrené podmienkou uvedenou v kóde 16.

```
scheme[selectedElement.type][subPar.data.id] =
$("#"+[subPar.data.id]).val();
scheme[selectedElement.type].Reload = 'yes'
```

Kód 15: Pridanie vlastnosti do premennej `scheme`

```
if( true == scheme[type].hasOwnProperty("Reload")){
  ... }
else{...
}
```

Kód 16: Ošetrenie načítavania hodnôt do formulára

Modálne okno pre blok typu `Scope` je zobrazené na obr. 12. Na tomto príklade je možné vidieť všetky spomínané súčasti formulárov.

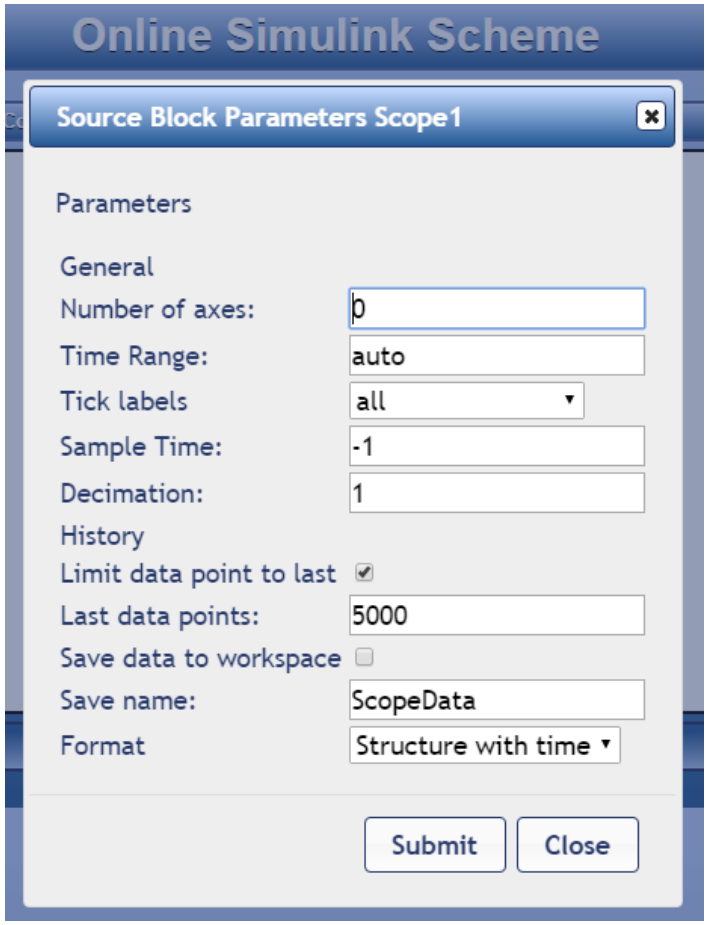
Ak by sme porovnali okno nastavení pre block `Scope` v Simulink-u a v aplikácii, formuláre nie sú úplne rovnaké. V aplikácii nie sú pridané všetky možnosti nastavenia bloku, ale iba tie, ktoré som považovala za najpodstatnejšie, teda najčastejšie nastavované. Ak by bolo potrebné pridať do formulára nové možnosti nastavenia, je potrebné pridať do `param.json` ďalší objekt do poľa.

5.1.5 Spojenie blokov

V aplikácii je možné spojiť dva bloky pomocou tlačidla `Add Connection`. Ako prvé je potrebné označiť blok, z ktorého bude spojenie vychádzať (zdrojový alebo vstupno-výstupný blok). Potom je potrebné stlačiť tlačidlo `Add Connection` a následne označiť blok, s ktorým chceme vytvoriť spojenie.

Bloky s viacerými vstupmi

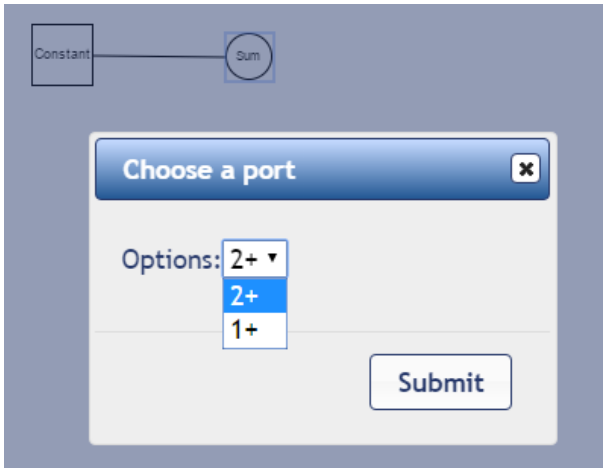
V knižnici blokov sa nachádzajú bloky, ktoré môžu byť spojené s viacerými blokmi.



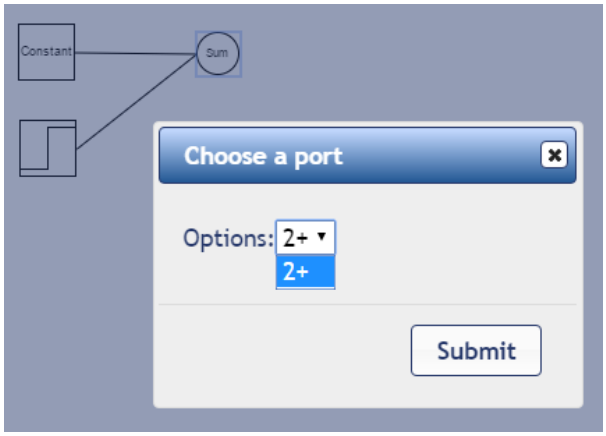
Obr. 12: Dialógové okno pre block `Scope`

Napríklad blok `Sum` a blok `Mux`. Ak by užívateľ chcel spojiť zdrojový blok s takýmto typom bloku, na stránke sa zobrazí modálne okno (obr. 13). V modálnom okne, musí užívateľ vybrať číslo vstupu, do ktorého bude spojenie pripojené. Užívateľ má na výber len zo vstupov, ktoré nie sú obsadené (obr. 14). Pri bloku `Sum` má užívateľ možnosť vidieť, či daný vstup je „+“ alebo „-“.

Po zmazaní spojenia, je opäť užívateľovi pri novom spojení ponúknutý voľný vstup. V nastaveniach blokov `Sum` a `Mux` má užívateľ možnosť menit' počet vstupov. Pri bloku `Sum` aj ich typ, pretože počet vstupov je menený na základe počtu znamienok „+“ a „-“.



Obr. 13: Dialógové okno pre výber prvého vstupu



Obr. 14: Dialógové okno pre výber druhého vstupu

Spojenia, ktoré sú vytvárané v aplikácii, sa líšia od spojení v programe Simulink. Aplikácia vytvára spojenia pomocou rovných čiar.

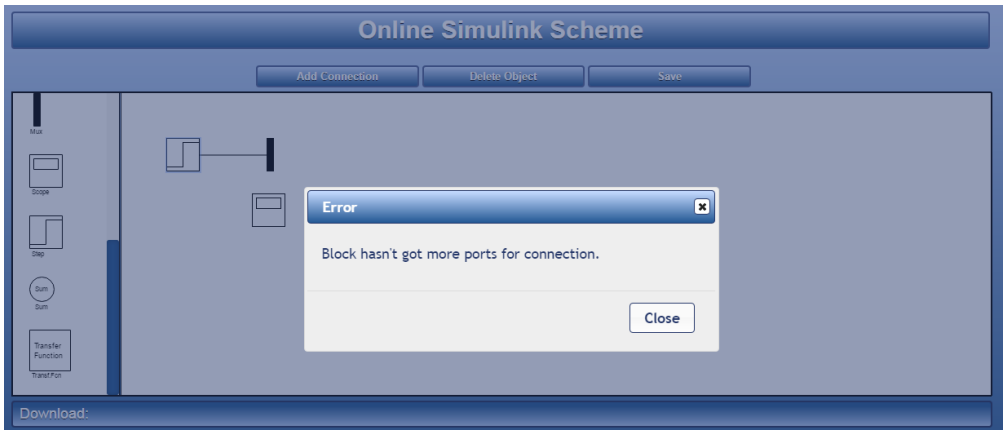
Kontrola možnosti spojenia blokov

Online aplikácia kontroluje vybrané bloky, ktoré chce užívateľ spojiť. Užívateľ nemá možnosť:

- spájať dva zdrojové bloky

- vytvoriť spojenie medzi dvomi spojenými blokmi
- vytvoriť spojenie bloku, ktorý už nemá voľné vstupy/výstupy (obr. 15)

Na obr. 15 je zobrazený blok `Step` spojený s blokom `Mux`. Blok `Step` má jeden výstup. Po pokuse spojiť blok `Step` s blokom `Scope`, je zobrazené modálne okno s chybovým hlásením.

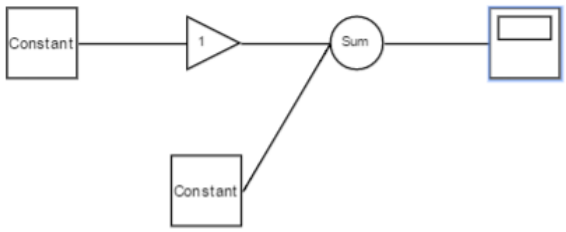


Obr. 15: Príklad vypísania chybovej správa v modálnom okne

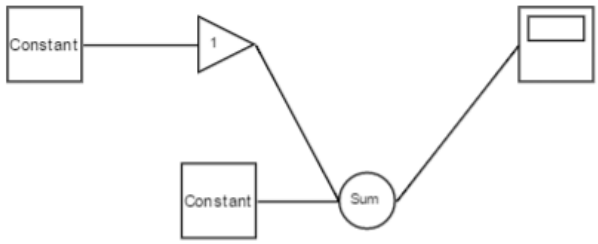
Ak by užívateľ spravil rovnakú vec v programe Simulink, program nevypisuje žiadne hlásenie ani nevytvorí spojenie.

Presúvanie bloku

Ak užívateľ vo vytvorenej schéme pohybuje niektorým blokom, spojenia sú zachované. Pri pohybe je v údajoch o bloku prepisovaná jeho pozícia a spojeniam sú prepisované súradnice bodov pripojenia. Na obr. 16 je znázornená schéma pred pohybom bloku a na obr. 17 je posunutý blok `Sum`.



Obr. 16: Pohybovanie s blokom - pred pohybovaním



Obr. 17: Pohybovanie s blokom - po pohybovaní

5.1.6 Uloženie schémy

Po kliknutí na tlačidlo „Save“, sú údaje z premennej `scheme` spracované PHP skriptom (asynchrónne). Prenos je realizovaný cez AJAX pomocou metódy `POST`.

V poli `$_POST` sa nachádzajú všetky informácie o blokoch a spojeniach. Skript prechádza cez celé pole a hľadá informácie o použitých blokoch. Na základe nájdenia bloku kontroluje, či v adresári bloku existujú súbory: `default.xml` a `system.xml`. Ak pre niektorý blok neexistuje jeden zo súborov, užívateľovi je zobrazená chybová správa. Príklad hlásenia chýbajúceho súboru pre blok `Clock` je na obr. 18.

Obr. 18: Hlásenie chyby pri generovaní súboru

Ak oba súbory existujú, do premennej `$default` je načítaný obsah zo súboru `default.xml`. Do premennej `$xmlDoc` sa vytvorí nový DOM document, do ktorého sa načíta obsah zo súboru `system.xml`. Pomocou funkcie `setSystemBlock()` sú hodnoty prepísané/doplnené hodnotami z premennej `$_POST` a vložené do premennej `$system`.

Ak sa v premenej `$_POST` nenachádzali len bloky, ale aj spojenia, do premennej `$line` je postupne pridávaný v XML jazyku kód (kód 17), ktorý má štruktúru, akú vyžaduje Simulink.

```
else if('true' == array_key_exists( 'Src', $_POST[$k]))
 {
   $line = '<Line>';
   $line .= '<P Name="ZOrder">'.$_POST[$k]['ZOrder'].'</P>';
   $line .= '<P Name="Src">'.$_POST[$k]['Src'].'</P>';
   $line .= '<P Name="Dst">'.$_POST[$k]['Dst'].'</P>';
   $line .= '</Line>';
   $system .= $line;
  }
```

Kód 17: Pridanie spojenia do výstupného dokumentu

Keďže v rovnakom čase je aplikácia prístupná viacerým užívateľom, musí byť názov schémy jedinečný, aby nedošlo k vzájomnému prepisovaniu pri generovaní. Názov schémy je zložený zo slova model a času, ktorý je reprezentovaný počtom sekúnd od 1.1.1970. (kód 18)

```
$filename = 'model' . time() . '.slx';
```

Kód 18: Vytvorenie názvu schémy

V koreňovom adresári aplikácie je vytvorená šablóna s názvom `untitled.slx`. Pri generovaní novej schémy je šablóna skopírovaná a uložená s aktuálnym názvom. V tom istom adresári sa nachádza aj súbor `blockdiagram.xml`, v ktorom sú nahradené elementy `<blockParametersDefautls>` a `<system>` reťazcami `$default$` a `$system$`. Tieto reťazce budú prepísané premennými `$default` a `$system`.

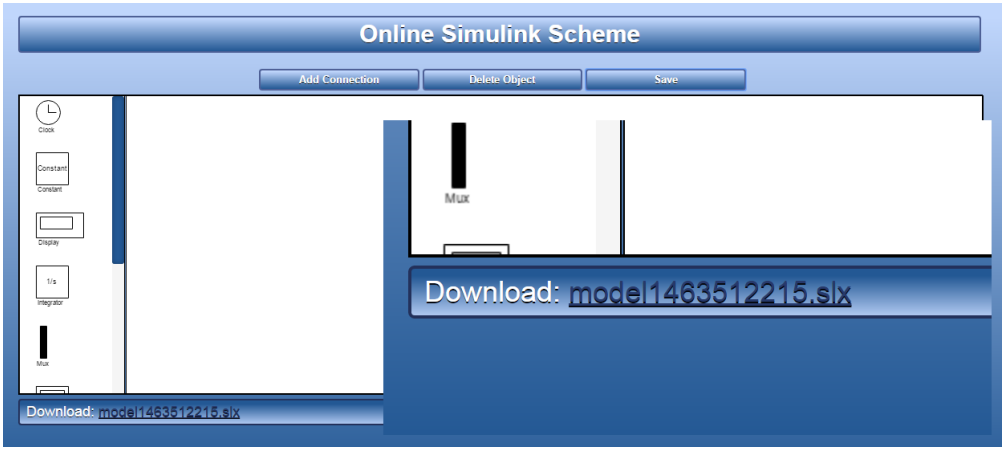
Pomocou jazyka PHP je možné pristupovať do existujúceho archívu zip. Je na to

potrebná trieda `ZipArchive`. V skopírovanej prázdnej schéme s novým názvom je nahradený pôvodný súbor `blockdiagram.xml`, novým súborom `blockdiagram.xml` (kód 19), ktorý už obsahuje hodnoty premenných `$default` a `$system`.

```
$zip = new ZipArchive;
if ($zip->open($dir_temp . $filename) === TRUE)
{
 $zip->addFromString('simulink/blockdiagram.xml', $filesource);
 $zip->close();
 echo '<a href="'.$dir_temp.$filename.'">'.$filename.'</a>';
}
```

Kód 19: Tvorba SLX schémy

Na stránke je vygenerovaný odkaz na súbor uloženej schémy (obr. 19).

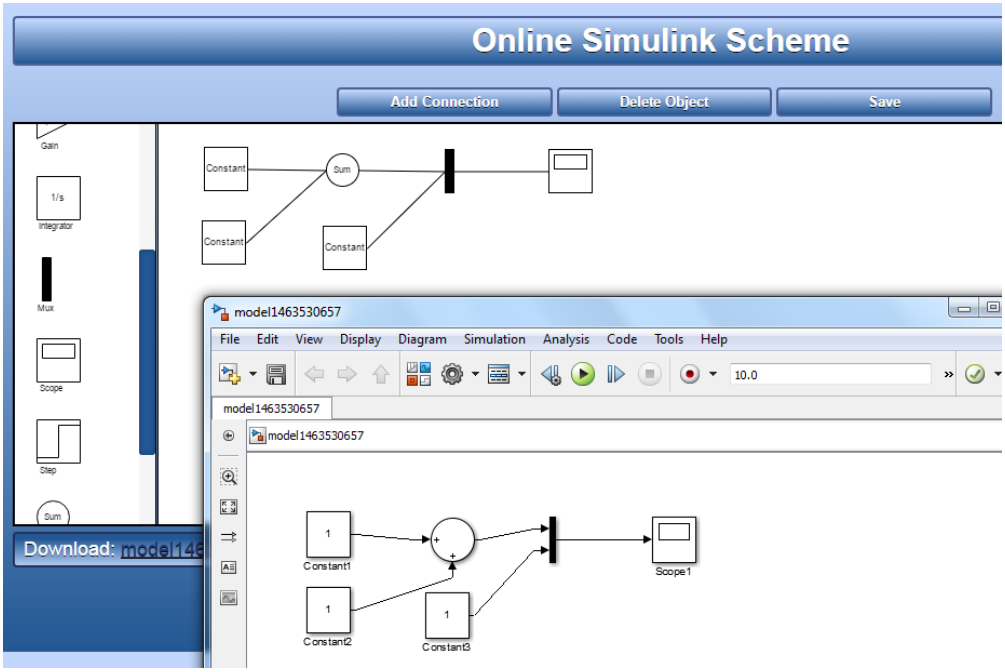


Obr. 19: Vygenerovaný odkaz na stiahnutie schémy

5.1.7 Porovnanie vytvorenej schémy so zobrazením v Simulink-u

Na obr. 20 je nakreslená schéma v online aplikácii a je porovnávaná po vygenerovaní so zobrazením v programe Simulink.

Na obrázku je možné vidieť, že schémy nie sú úplne totožné. Avšak schéma vygenerovaná v online aplikácii je funkčná v programe Simulink. V rámci diplomovej práce sme sa nezaoberali presnou kópiou vzhľadu schém v porovnaní so Simulink-om.



Obr. 20: Porovnanie schém

5.2 Pridanie nového bloku do knižnice blokov

Online aplikácia Simulink-u je navrhnutá tak, aby po jej dokončení, bolo možné pridávať užívateľom bloky do knižnice bez toho, aby musel zasahovať do kódu. V nasledujúcich krokoch je podrobne popísaný návod na to, ako pridať vstupno-výstupný blok `Gain` (obr.21).

Obr. 21: Blok `Gain` v Simulink-u

5.2.1 Vytvorenie adresára

Ako prvé je potrebné v adresári `library` vytvorit' nový adresár s názvom bloku. Tento adresár bude obsahovat' 4 súbory: `display.json`, `param.json`, `default.xml` a `system.xml`.

5.2.2 Vykreslenie bloku

Blok je zložený z trojuholníka a textu. Do `display.json` pridáme údaje o zobrazení bloku. Koreňový element bude názov bloku a bude obsahovat' pole objektov. Keďže je to pole objektov, nevyhnutná je čiarka medzi jednotlivými objektami v poli.

```
{
"Gain": []
}
```

Kód 20: Koreňový element v súbore `display.json`

Do tohto poľa budeme postupne pridávat' objekty. Prvý je trojuholník, z ktorého sa skladá blok `Gain`. Type je `triangle`. Tento blok bude mat' rozmery 30×30, vyplnený bude bielou farbou, umiestnený v Canvase na pozícii [30, 0], uhol otočenia trojuholníka je 90 stupňov a farba rámu bude čierna. Pozícia je podstatná len pri kreslení d'alších tvarov, pretože knižnica je generovaná automaticky, teda pozícia bloku je v nej pevne daná a vloženie do schémy je opät' pevne dané. Užívateľ môže tieto hodnoty l'ubovoľne nastavit' (kód 21).

```
{
  "type": "triangle",
  "data":
  {
    "width":30,
    "height":30,
    "left": 30,
    "top": 0,
    "strokeWidth": 1,
    "stroke": "black",
```

```
    "fill": "white",
    "angle": 90
  }
},
```

Kód 21: Pridanie tvaru trojuholníka

Druhým prvkom v bloku je text. Blok obsahuje dva rozličné texty. Ako prvý pridáme text, ktorý sa nachádza vo vnútri bloku (kód 22). Text, ktorý sa nachádza vo vnútri bloku má nastavený type ako `text`.

```
{
  "type": "text",
  "Text": "1",
  "data":
  {
    "fontFamily": "Arial",
    "left": 5,
    "top": 10,
    "fontSize": 8,
    "fill": "black"
  }
},
```

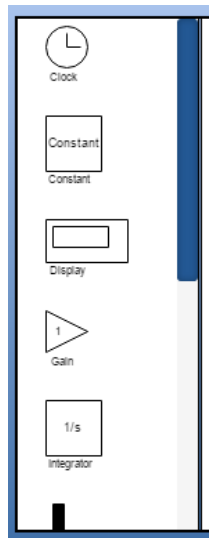
Kód 22: Pridanie textu do vnútra trojuholníka

Text, ktorý je pomenovaním bloku a nachádza sa pod ním, má type nastavený na `name` (kód 23). Pridávanému textu môžeme nastavit' font, veľkost' a farbu.

```
{
  "type": "name",
  "Text": "Gain",
  "data":
  {
    "fontFamily": "Arial",
    "left": 2,
```

```
    "top": 32,
    "fontSize": 8,
    "fill": "black",
  }
}
```

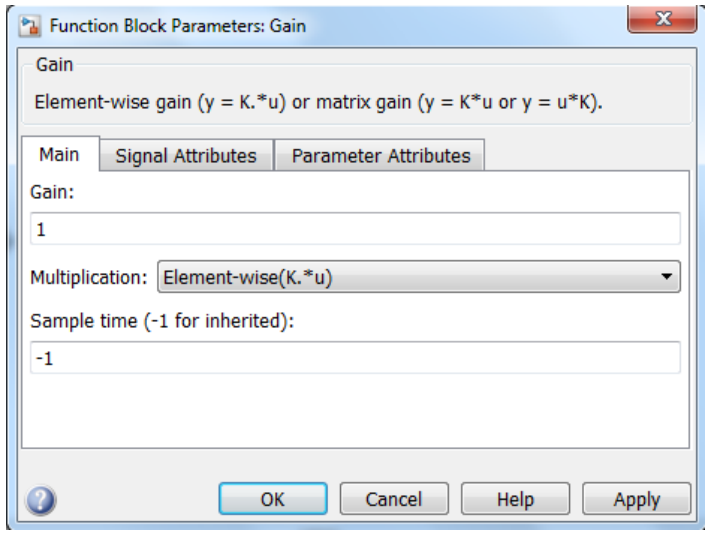
Kód 23: Pridanie názvu bloku `Gain`



Obr. 22: Blok `Gain` v knižnici online aplikácie

Blok v knižnici blokov je zobrazený na obr. 22. Do knižnice blokov je možné pridávať bloky, ktoré sa skladajú z nasledovných typov:

- `rect` - pridanie obdĺžnika
- `path` - pridanie cesty
- `triangle` - pridanie trojuholníka
- `circle` - pridanie kruhu
- `text` - pridanie textu v bloku
- `name` - pridanie názvu pod blok



Obr. 23: Formulár pre blok `Gain`

Podrobný popis atribútov, ktoré je možné nastaviť jednotlivým typom, je možné naštudovať v dokumentácii Fabric [13].

5.2.3 Nastavenie parametrov formulára

Pri nastavení formulára parametrov pre bloky je potrebné vedieť, ako vyzerá formulár parametrov v Simulink-u a aké názvy a hodnoty nadobúdajú prvky formulára v `blockdiagram.xml`.

Na obr. 23 je zobrazený formulár pre blok `Gain`. Do formulára online aplikácie som sa rozhodla pridať časť main, ktorá obsahuje:

- textové pole - Gain
- selectbox - Multiplication
- textové pole - Sample Time

V Simulink-u je potrebné postupne ukladať schémy so všetkými možnosťami nastavenia parametrov, aby sme zistili názvy a hodnoty parametrov, ktoré generuje Simulink.

Do súboru `param.json` vložíme koreňový element `Gain`, ktorý bude obsahovať pole objektov kód 24.

```
{
"Gain": []
}
```

Kód 24: Koreňový element v súbore `param.json`

Prvý objekt poľa sú všeobecné informácie o bloku. Tento blok je vstupno-výstupným blokom, preto `io` bude mať hodnotu `both`. Ak by bol blok len vstupný, napr. `Scope`, hodnota by bola `in` a naopak ak výstupný, hodnota by sa rovnala `out`.

```
{
  "io": "both",
  "NumberOfInputs": 1,
  "NumberOfOutputs": 1,
  "BlockType" : "Gain",
  "title": "Source Block Parameters Gain"
},
```

Kód 25: Pridanie všeobecných informácií o bloku `Gain`

Nasledujúce objekty v poli slúžia na definovanie formulára. Tieto objekty musia byť vložené v takom poradí, v akom ich chceme vykresliť. Objekty sú rozlíšené typom. Pre formulár je možné zadefinovať 4 typy.

1. `text` - použitie tohto typu vytvorí vo formulári textové pole (obr. 24)

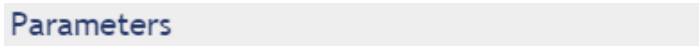
Obr. 24: Formulár - textové pole

2. `selectbox` - slúži na vytvorenie výberového poľa (obr. 25)

Obr. 25: Formulár - výberové pole

3. `checkbox` - vytvorí zaškrtávacie pole (obr. 26)

Obr. 26: Formulár - zaškrtávacie pole pole



Obr. 27: Formulár - nadpisy

4. `describe` - tvorí podnadpisy jednotlivých častí formulára (obr. 27)

Ďalším objektom v poli podľa obrázku 23 je nadpis Main. Do poľa objektov vložíme nasledovný kód .

```
{
  "type" : "describe",
  "text": "Main"
},
```

Kód 26: Pridanie podnadpisu do formulára bloku `Gain`

Vo formulári sa pod nadpisom nachádza textové pole. Ako typ zvolíme `text`. `Title` popisuje pole, `name`, `id` a `size` sú atribúty v `HTML` kóde a value je preddefinovaná hodnota, ktorá sa zobrazí vo formulári.

```
{
  "type": "text",
  "data":
  {
    "title": "Gain",
    "Name": "Gain",
    "size": "10",
    "id": "Gain",
    "value": "1"
  }
},
```

Kód 27: Pridanie textového poľa do fomulára bloku `Gain`

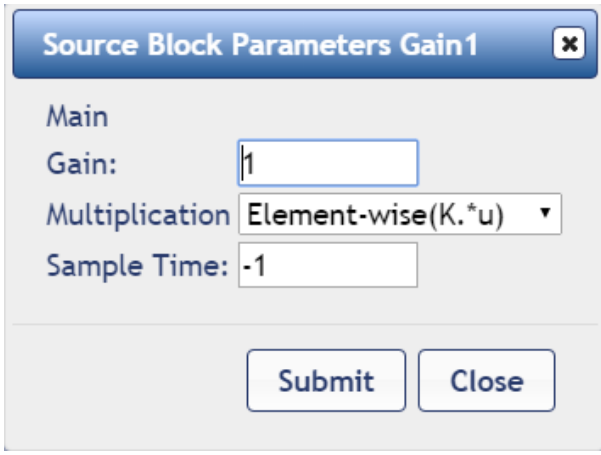
Pod textovým poľom je vo formulári výberový zoznam. V objekte `data` sa nachádza `title`, ktorý popisuje výberový zoznam. `Name` a `id` sú atribúty v HTML kóde a objekt `opt` obsahuje položky výberového zoznamu. Sú tvorené hodnotou a názvom.

```
{
    "type": "selectbox",
    "data":
    {
      "title": "Multiplication",
      "Name": "Multiplication",
      "id": "Multiplication",
      "opt":
      [
        {
          "value": "Element-wise(K.*u)",
          "name": "Element-wise(K.*u)"
        },
        {
          "value": "Matrix(K*u)",
          "name": "Matrix(K*u)"
        },
        {
          "value": "Matrix(u*K)",
          "name": "Matrix(u*K)"
        },
        {
          "value": "Matrix(K*u) (u vector)",
          "name": "Matrix(K*u)(u vector)"
        }
      ]
    }
},
```

Kód 28: Pridanie výberového zoznamu do formulára bloku `Gain`

Poslednou súčast'ou formulára je d'alšie textové pole. Toto pole je pridané rovnako ako prvé pole.

Výsledný formulár po pridaní všetkých častí je zobrazený na obr. 28.



Obr. 28: Vytvorený formulár v online aplikácii pre blok `Gain`

5.2.4 Súbor `default.xml`

Tento súbor je napísaný v jazyku XML a nič sa v ňom v priebehu aplikácie nemení. Súbor je potrebný pre naplnenie elementu `<BlockParameterDefaults>` v `blockdiagram.xml`. Ak je blok pridaný do schémy, po uložení schémy sa do daného elementu vloží celý obsah tohto súboru. Pre blok `Gain` sú preddefinované parametre v kóde 29.

```
<Block BlockType="Gain">
  <P Name="Gain">1</P>
  <P Name="Multiplication">Element-wise(K.*u)</P>
  <P Name="ParamMin">[]</P>
  <P Name="ParamMax">[]</P>
  <P Name="ParamDataTypeStr">Inherit: Same as input</P>
  <P Name="OutMin">[]</P>
  <P Name="OutMax">[]</P>
  <P Name="OutDataTypeStr">Inherit: Same as input</P>
  <P Name="LockScale">off</P>
```

```
  <P Name="RndMeth">Floor</P>
  <P Name="SaturateOnIntegerOverflow">on</P>
  <P Name="SampleTime">-1</P>
</Block>
```

Kód 29: Preddefinované hodnoty v `default.xml` pre blok `Gain`

5.2.5 Súbor `system.xml`

Tento súbor musí obsahovať dáta, ktoré majú možnosť byť menené nielen v online aplikácii, ale v programe Simulink.

Element `<Block>` obsahuje atribúty `BlockType`, `Name` a `SID`. Atribút `BlockType` má preddefinovanú hodnotu. `Name` a `SID` sú prázdne (kód 30). Do týchto atribútov je vložená ich hodnota po stlačení tlačidla `Save`.

V kóde 30 sú elementy `<p>` s atribútami `Name`, ktorých hodnoty sú názvy parametrov. Hodnota parametrov je ukladaná do elementov `<p>`. Tieto elementy majú v súbore `system.xml` preddefinované hodnoty. Ak neexistuje preddefinovaná hodnota (napr. `Zorder` a `Position`), elementy sú prázdne. Naplnené sú hodnotami po uložení schémy.

Obsah súboru `system.xml` je zapisovaný do elementu `<System>` v `blockdiagram.xml`.

```
<Block BlockType="Gain" Name="" SID="">
 <P Name="Position"></P>
 <P Name="ZOrder"></P>
 <P Name="Gain">1</P>
 <P Name="Multiplication">Element-wise(K.*u)</P>
 <P Name="ParamDataTypeStr">Inherit: Inherit via internal rule
 </P>
 <P Name="OutDataTypeStr">Inherit: Inherit via internal rule</P>
 <P Name="SaturateOnIntegerOverflow">off</P>
</Block>
```

Kód 30: Nastavované hodnoty v `system.xml` pre blok `Gain`

Opakovaním týchto piatich krokov vie užívateľ pridať množstvo blokov, ktoré potrebuje vo svojej schéme. Štruktúru jednotlivých súborov je potrebné zachovať pre správny chod aplikácie.

6 Záver

Cieľom diplomovej práce bolo vytvoriť internetovú verziu programu Simulink, ktorá umožňuje užívateľovi vytvárať jednoduchú blokovú schému v okne webového prehliadača. Schému je možné uložiť a stiahnuť vo formáte SLX, ktorý je podporovaný programom Simulink.

Prvou úlohou bol výber a naštudovanie použitých technológií, ktorý sa počas vývoja aplikácie výrazne menil. Najpodstatnejšou zmenou bola zmena v ukladaní dát, ktorá spôsobila zmenu architektúry aplikácie. Aplikácia pri vývoji prešla zmenou z trojvrstvovej architektúry na dvojvrstvovú, pretože databáza MySQL bola vynechaná a nahradená formátom JSON, ktorý beží na strane klienta. To znamená, že nie je potrebná častá komunikácia so serverom.

Ďalšou úlohou bolo naštudovanie ukladania blokových schém v programe Simulink. Konkrétne štruktúry súboru `blockdiagram.xml` pri zmenách v blokovej schéme ako je napríklad zmena parametrov, zmena pozície, tvorba spojení medzi jednotlivými blokmi a pod.

Po naštudovaní vhodných technológií a štruktúry schém v Simulinku, som začala vytvárať grafické prostredie aplikácie. Využila som Canvas a knižnicu Fabric, ktoré sú pre túto aplikáciu postačujúce.

Ako prvé som vytvorila knižnicu blokov, ktorá je generovaná pri spustení web stránky automaticky. Bloky z knižnice blokov je možné pridať do schémy dvojklikom. Po pridaní bloku do schémy má užívateľ možnosť každému bloku meniť jeho parametre. Bloky je možné spájať pomocou tlačidla `Add Connection`, prípadne ich zmazať pomocou tlačidla `Delete`.

V aplikácii sa nachádzajú bloky, ktoré môžu mať: jeden výstup, jeden vstup, jeden výstup a jeden vstup a bloky, ktoré môžu mať viac vstupov a jeden výstup. Pri poslednom typu bloku je pri vytvorení spojenia zobrazované modálne okno, ktoré slúži na výber vstupu.

Spojenia blokov sú vytvárané pomocou jednoduchých čiar. Toto vykresľovanie nie je podobné aké generuje Simulink, no pre funkčnosť schém, teda ich možnosť simulácie v Simulink-u, je takéto spojenie postačujúce.

Užívateľ má možnosť rozšíriť knižnicu blokov. V podkapitole 5.2 je uvedený presný

manuál k pridaniu blokov. Pri pridaní blokov musí byť dodržaná štruktúra súborov. Na uvedenom príklade pridania bloku v podkapitole 5.2, je možné vidieť všetky štruktúry.

Vytvorenie online aplikácie Simulink-u má uplatnenie vo virtuálnych laboratóriách. V laboratóriách sa nachádzajú procesy, ktoré je možné ovládať prostredníctvom blokových schém vyrobených v Simulink-u. Nakoľko sú tieto schémy statické, teda študent má k dispozícii už pripravenú schému, v ktorej vie meniť len určité parametre, napríklad pomocou formulára, vyvinutá aplikácia má veľký význam.

Literatúra

[1] R. Puerto, L.M. Jiménez, and O. Reinoso. Remote control laboratory via internet using matlab and simulink. *Computer Applications in Engineering Education*, 18(4):694–702, 2010.

[2] Porovnanie bitmapových formátov s formátom SVG. `https://sk.wikipedia.org/wiki/Scalable_Vector_Graphics#/media/File:Bitmap_VS_SVG.svg`, 2006. [Online; prístupné 16.5.2016].

[3] Gliffy. `https://www.gliffy.com/`. [Online; prístupné 16.5.2016].

[4] Insight Maker. `https://insightmaker.com/`. [Online; prístupné 16.5.2016].

[5] Draw.ior. `https://www.draw.io/`. [Online; prístupné 16.5.2016].

[6] Z. Janík. Internetom podporovaná modifikácia blokových schém v matlab/simulinku. Diplomová práca, Ústav riadenia a priemyselnej informatiky (FEI) STU v Bratislave, 2010.

[7] Z. Janík and Žáková K. Online design of matlab/simulink and scilab/xcos block schemes. In *14th International Conference on Interactive Collaborative Learning - 11th International Conference Virtual University*, 2011.

[8] SLX. `http://pc.net/extensions/file/slx`, 2013. [Online; prístupné 30.5.2016].

[9] Introducing JSON. `http://www.json.org/json-cz.html`, 2016. [Online; prístupné 20.2.2016].

[10] HTML5 Canvas. `http://www.w3schools.com/html/html5_canvas.asp`. [Online; prístupné 6.5.2016].

[11] SVG. `http://www.w3schools.com/svg`. [Online; prístupné 6.5.2016].

[12] Raphael (JavaScript library). `https://en.wikipedia.org/wiki/Rapha%C3%ABl_(JavaScript_library)`. [Online; prístupné 20.2.2016].

[13] Fabricjs Home. `http://fabricjs.com/docs/`, 2016. [Online; prístupné 10.5.2016].

[14] Bc. Jakub Chovan. Architektúry informačných systémov klient-server a soa. Diplomová práca, Vvsoká škola báňská - Technická Univerzita Ostrava, 2013.

[15] XML. https://sk.wikipedia.org/wiki/XML, 2013. [Online; prístupné 5.5.2016].

[16] XML pro začátečníky. http://programujte.com/clanek/2007030501-xml-pro-zacatecniky-1-cast/, 2014. [Online; prístupné 5.5.2016].

[17] Hypertextový značkový jazyk. http://sk.wikipedia.org/wiki/Hypertextov%C3%BD_zna%C4%8Dkov%C3%BD_jazyk, 2014. [Online; prístupné 8.5.2016].

[18] HTML příručka. http://www.jakpsatweb.cz/html/. [Online; prístupné 8.5.2016].

[19] Kaskádové štýly. http://sk.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_%C5%A1t%C3%BDly, 2014. [Online; prístupné 8.5.2016].

[20] JavaScript. http://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk/, 2015. [Online; prístupné 4.4.2016].

[21] David Flanagan. *JavaScript*. Computer Press, 2.vydanie, 2002.

[22] jQuery. http://sk.wikipedia.org/wiki/JQuery, 2015. [Online; prístupné 8.5.2016].

[23] jQuery UI. https://cs.wikipedia.org/wiki/JQuery_UI, 2015. [Online; prístupné 8.5.2016].

[24] AJAX. https://cs.wikipedia.org/wiki/AJAX, 2015. [Online; prístupné 7.12.2015].

[25] PHP. https://cs.wikipedia.org/wiki/PHP, 2015. [Online; prístupné 8.5.2016].