PROHLÁŠENÍ

Předkládám tímto k posouzení a obhajobě bakalářskou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem předloženou bakalářskou práci vypracoval samostatně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

Plzeň dne 16. srpna 2013 podpis bakaláře

ANOTACE

Hlavním cílem této práce je vytváření grafického editoru blokových diagramů. Na začátku práce je uvedeno srovnání technologií pro zobrazování grafického uživatelského rozhraní. Dále je uvedeno řešení editoru blokových diagramů pro řídicí systém REX implementované v programovacím jazyce Java s pomocí grafické knihovny JavaFX. V editoru je možné vytvářet diagramy z bloků knihovny JavaREX a provádět jejich simulaci.

Klíčová slova

blokový diagram, grafický editor, JavaREX, JavaFX

ANNOTATION

The main objective of this thesis is the development of graphical editor for block diagrams. The first part of the thesis compares the technologies used for displaying graphical user interface. The second part is centered on solution of a block diagram editor for REX control system implemented in Java programming language with use of the JavaFX graphics library. The aforementioned editor allows to create diagrams of blocks from JavaREX library and to perform their simulation.

Keywords

block diagram, graphical editor, JavaREX, JavaFX

Obsah

1. Úvod

Bakalářká práce se zabývá návrhem editoru blokových schemat řídicího systému REX ve webovém prohlížeči. Práce bude sloužit pro demonstrační spouštění modelů a regulátorů vytvořených z bloků systému REX. Řešení problému lze rozdělit do několika částí. Základní dělení je na část teoretickou a část praktickou. Teoretická část se zabývá výběrem mezi dostupnými webovými technologiemi v návaznosti na možnosti kreslení blokových schemat a propojení do systému JavaREX.

1.1. Cíl práce

Práce umožní tvorbu diagramů řídicí systém REX v prostředí webového prohlížeči. Tento problém dosud není vyřešen v systému REX a ani další současné řídicí systémy nenabízí možnosti testovat jejich algoritmy přímo v otevřeném prostředí webového prohlížeče. Využití tohoto editoru bude především pro demonstrační účely systému REX, kde umožní zájemcům přímo testovat pokročilé algoritmy řízení na libovolné modely systémů. Editor také umožní rozšířit využití řídicího systému REX ve vzdělávacích virtuálních laboratořích. Navíc výchozí technologie vedené pro tuto práci jsou platformově nezávislé a tedy dalším rozvojem navrženého editoru bude možné navrhnout další sofistikovanější editor, který umožní např. pracovat i se soubory typu `*.mdl` současného systému REX.

Editor by měl počítat s možností spouštění z webového prohlížeče na libovolné platformě. Tedy se nabízí použití technologií Java nebo JavaScript.

1.2. Členění práce

V Úvodu je popsán cíl práce. Následuje kapitola Technologie, která rozebírá výhody a nevýhody různých technologií pro psaní editoru bloků. V kapitole Řešení je popsána navržená architektura aplikace. Následuje kapitola Uživatelský manuál, která popisuje použití programu uživatelem. Navíc je rozšířena o příklad použití na modelovém příkladu PID regulátoru s momentovým autotunerem [1]. Poslední je kapitola Závěr, kde jsou krátce shrnuty všechny výsledky práce.

2. Volba vhodné technologie

V této kapitole budou zpracovány podklady pro výběr vhodných technologií pro řešení problému. Vzhledem k požadavku k nasazení ve webovém prohlížeči byly zkoumány dvě základní technologie JavaScript a Java. Pro rozhodnutí o výběru technologie byl navržen test, který vykresluje velké množství grafických objektů a umožní tak vybrat nejrychlejší technologii.

2.1. JavaScript

JavaScript, standardizovaný jako ECMAScript [2], je skriptovací jazyk dostupný prakticky ve všech webových prohlížečích. Jedná se o tzv. *Prototype-based* objektově orientovaný jazyk s dynamickým typováním a automatickou správou paměti (Garbage collector). V současnosti je k dispozici verze ECMAScript 5 [3].

I přes podobnost názvu se jedná o zcela odlišnou technologii než Java.

Velkou výhodou jazyka JavaScript je jeho kompatibilita s webovými prohlížeči. Programy lze spouštět prakticky na všech zařízeních vybavených webovým prohlížečem. Kromě počítačů i mobilní telefony, tablety a další zařízení.

Práce s grafikou v JavaScriptu je možná prostřednictvím manipulace s DOM webové stránky (vytváření a upravování HTML elementů), případně pomocí technologií uvedených v HTML5 – Canvas a SVG [4].

Nevýhodou jazyka je odlišné chování některých funkcí na různých webových prohlížečích. Tento problém je však možné odstranit použitím nadstavbové knihovna např. jQuery. Zásadním problémem je dynamické typování, které znemožňuje provést efektivní typovou kontrolu před spuštěním programu, což činí jakoukoliv tvorbu rozsáhlejších aplikací velmi problematickou.

Tento jazyk nebyl zvolen jako vhodná varianta, vzhledem k relativní složitosti projektu editoru. JavaScript není vhodně vybaven k psaní tak rozsáhlých aplikací.

2.1.1. GWT

Google web toolkit (GWT) je systém pro převod programu v jazyce Java do JavaScriptu. Tento projekt však není vyvíjen v součinnosti s vývojem Javy a proto zaostává v podpoře nejnovějších verzí jazyka Java a podporuje jenom omezené množství standardních knihoven [5]. Tuto technologii používá Google pro vývoj svých aplikací jako jsou Google Docs apod.

Systém GWT byl zavržen především z důvodu omezené kompatibility se standardní knihovnou jazyka Java, kde chybí např. třídy pro reflexi, což znemožňuje pohodlné načítání údajů z knihovny JavaREX, stejně jako dynamické načítání rozšiřujících tříd.

2.2. Java

Dalším jazykem podporovaným v prostředí webových prohlížečů je jazyk Java. Zde je ovšem potřeba webový prohlížeč vybavený rozšířením pro Javu, což už není tak běžné

jako v případě JavaScriptu. [6]

Jazyk Java je *Class-based* objektově orientovaný programovací jazyk s automatickou správou paměti spouštěný ve virtuálním stroji Java Virtual Machine (JVM). Jazyk je tedy podporován standardně používanými platformami Windows, Linux a Mac. Naopak dnes populární systém Android využívá vlastní virtuální stroj Dalvik, navíc není vybaven knihovnou kompatibilní se standardní verzí JavaSE [7].

Nejčastěji je jazyk Java je používán v bussines aplikacích pro rozsáhlé projekty jako jsou webové servery, databázové systémy, firemní aplikace. [8]

Samotná knihovna JavaREX je implementována v jazyce Java a tedy využití jazyka Java pro aplikaci editoru usnadní propojení editoru s knihovnou bloků.

Jako nejvhodnější pro psaní editoru se tedy jeví použití jazyka Java i s přihlédnutím k autorovým předchozím zkušenostem.

Editor bloků je grafická aplikace a jazyk Java nabízí několik knihoven pro grafiku. Dalším rozhodnutím je volba grafické knihovny. Grafická knihovna je vybírána na základě možností jednotlivých knihoven, jejich plánovanému rozvoji do budoucna a nakonec i podle jednoduchého testu vykreslování (zdrojový kód, a spustitelný test se nachází na přiloženém CD). Grafické knihovny jsou Abstract Window Toolkit (AWT), Swing, Standard Widget Toolkit (SWT) a JavaFX. Popis jednotlivých knihoven je v dalších podkapitolách.

Volba nakonec padla na knihovnu JavaFX. Důvody k výběru jsou shrnuty v tabulce 2. Test výkonosti byl vyhodnocen subjektivně na základě počtu plynule zobrazovaných pohybujících se grafických objektů.

2.2.1. AWT

Abstract Window Toolkit je původní grafická knihovna Javy [9]. Knihovna je založena na tzv. Heavy-weight komponentách, které využívají přímo okna operačního systému. Hlavní nevýhodou tohoto systému je hardwarová náročnost vytváření grafického prostředí, každá součást si nárokuje systémové zdroje, které se navíc nedají automaticky spravovat Garbage collectorem Javy.

2.2.2. Swing

Swing je nadstavba nad jádrem AWT. Narozdíl od AWT je založena na tzv. Light-weight komponentách, které využívají jedno společné okno systému a veškerá správa komponent je řešená přímo v Javě.

2.2.3. SWT

Standard widget toolkit je knihovna, která vznikla pro potřeby vývojového prostředí Eclipse. U knihovny je kladen důraz na podporu specifických vlastností cílových operačních systémů a nativního vzhledu aplikací. SWT vychází ze zastaralé technologie multiplatformního grafického rozhraní pro OTI Smalltalk. [10]

Technologie	Jazyk	Požadavky	Poznámka
JavaScript	JavaScript	Webový prohlížeč	Standardní rozhraní JavaScriptu v prohlížeči
jQuery	JavaScript	Webový prohlížeč	Knihovna JavaScriptu
GWT	Java	Webový prohlížeč	Nástroj pro překlad jazyka Java do JavaScriptu a implementace části tříd z JavaSE pro JavaScript
AWT	Java	JRE	Standardní součást JRE od první verze [9]
Swing	Java	JRE	Standardní součást JRE od verze 1.2 [13]
SWT	Java	JRE + nativní knihovna	Knihovna vytvořená pro potřeby vývojového prostředí Eclipse [10]
JavaFX	Java	JRE	Samostatná knihovna přibalená k JRE, bude integrována do JRE ve verzi 8 [12]

Tabulka 1: Přehled technologií

2.2.4. JavaFX

JavaFX je nejnovější grafická knihovna Javy, která vnáší do grafických kompoment nové možnosti. Technologie využívá graf scény, což je n-ární strom grafických prvků. Tato technologie umožňuje dědičnost grafických vlastností ve stromě. [11] Všechna zobrazení jsou narozdíl od předchozích knihoven prováděna s plnou hardwarovou akcelerací. Knihovna je v současnosti aktivně vyvíjena a v Java 8 by se měla objevit jako hlavní grafická knihovna. [12]

Velkou výhodou použití knihovny do budoucna je zaměření na moderní multimediální technologie. Knihovna podporuje přehrávání zvuku, videa a zobrazení 3D grafiky. Vše s využitím hardwarové akcelerace.

Vzhledem k probíhajícímu vývoji a očekávaným novým verzím se knihovna jeví jako vhodná pro využití v budoucích projektech. Pro projekt grafického editoru byla zvolena právě knihovna JavaFX.

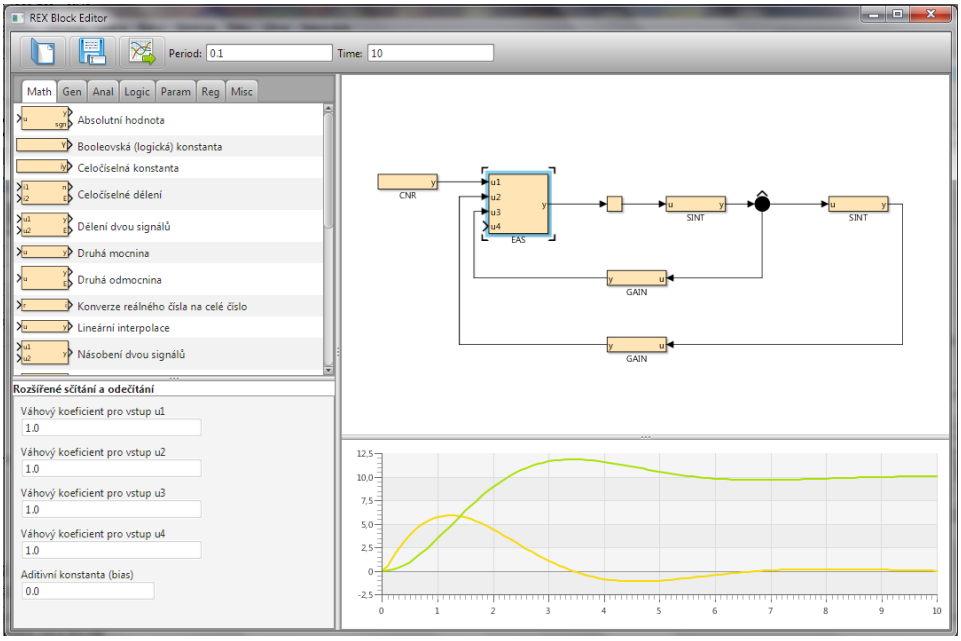
Technologie	Výhody	Nevýhody
JavaScript	Minimální velikost a správná funkce v současných prohlížečích	Problémy s podporou starších prohlížečů + Stejné jako u jQuery
jQuery	Široká podpora většiny verzí prohlížečů	Dynamicky typovaný jazyk
GWT	Použití staticky typovaného jazyka pro vývoj aplikací pro prohlížeče	Zastaralá verze jazyka Java a velmi omezená knihovna
AWT	Standardní součást JRE o jeho počátku	Zastaralá knihovna
Swing	Standardní součást JRE od verze 1.2	Zastaralá knihovna
SWT	Možnost integrace do vývojového prostředí Eclipse	Zastaralá architektura (počátek 90. let [10])
JavaFX	Vysoký grafický výkon a množství vizuálních efektů	Základ knihovny je stále ve vývoji

Tabulka 2: Vyhodnocení technologií

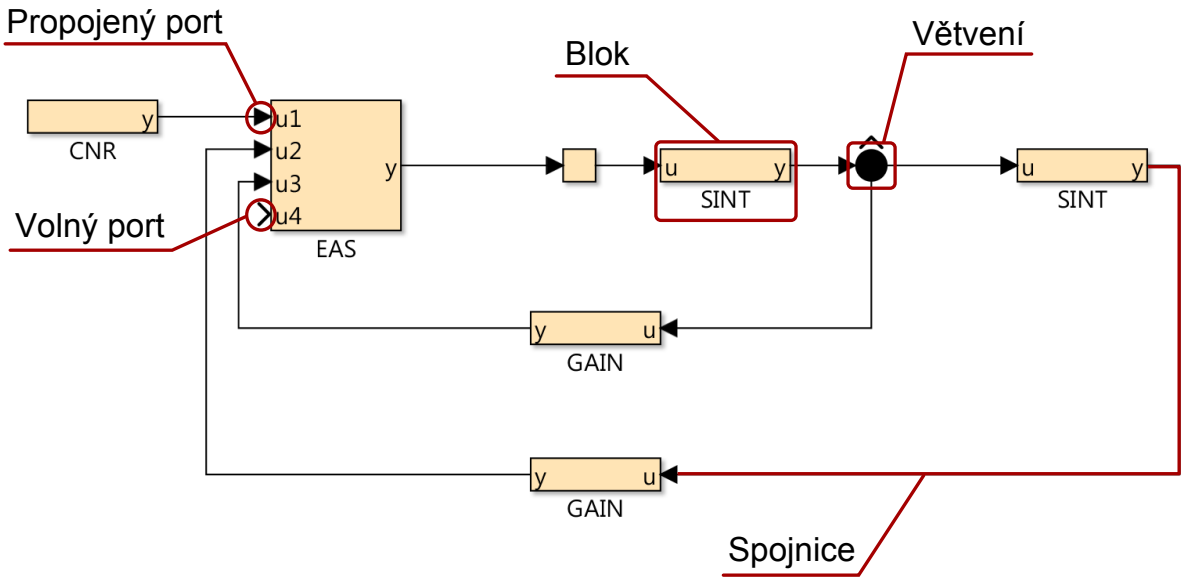
3. Řešení

Jak už bylo zmíněno výše samotné řešení problému je navrženo jako Java Applet s využitím knihovny JavaFX umožňující implementaci moderní rychlé grafiky. K programování samotnému byl zvolen editor Eclipse.

Myšlenkou aplikace je vytvořit kompaktní prostředí pro editaci schemat a spouštění simulovaných výsledků. Aplikace je soustředěna do jednoho okna, kde se kolem samotného editoru zobrazují panely nástrojů. Pro účely aplikace bylo vytvořeno několik samostatných komponent v knihovně JavaFX. Návrh editoru je na obr. 1



Obrázek 1: Okno editoru diagramů



Obrázek 2: Součásti diagramu

Hlavní lišta využívá komponentu `ToolBar` se sadou obrázkových tlačítek pro základní operace s editorem.

Editor diagramu byl navržen jako nová komponenta popsaná níže.

Knihovna bloků využívá standardní komponentu `ListView` ve které obsahuje seznam dostupných bloků z knihovny JavaREX. Pro každou položku je vytvořen prototyp bloku (`Block`) a komponenta jeho zobrazení (`BlockView`), která slouží jako náhled.

Editor paramatrů vytváří seznam komponent editorů jednotlivých parameterů vybraného bloku. Typ editoru závisí na typu parametru definovaném v knihovně JavaREX.

Graf využívá existující komponentu `LineChart` z knihovny JavaFX, která umožňuje zobrazování jednoho nebo více spojnicových grafů na základě výsledku simulace.

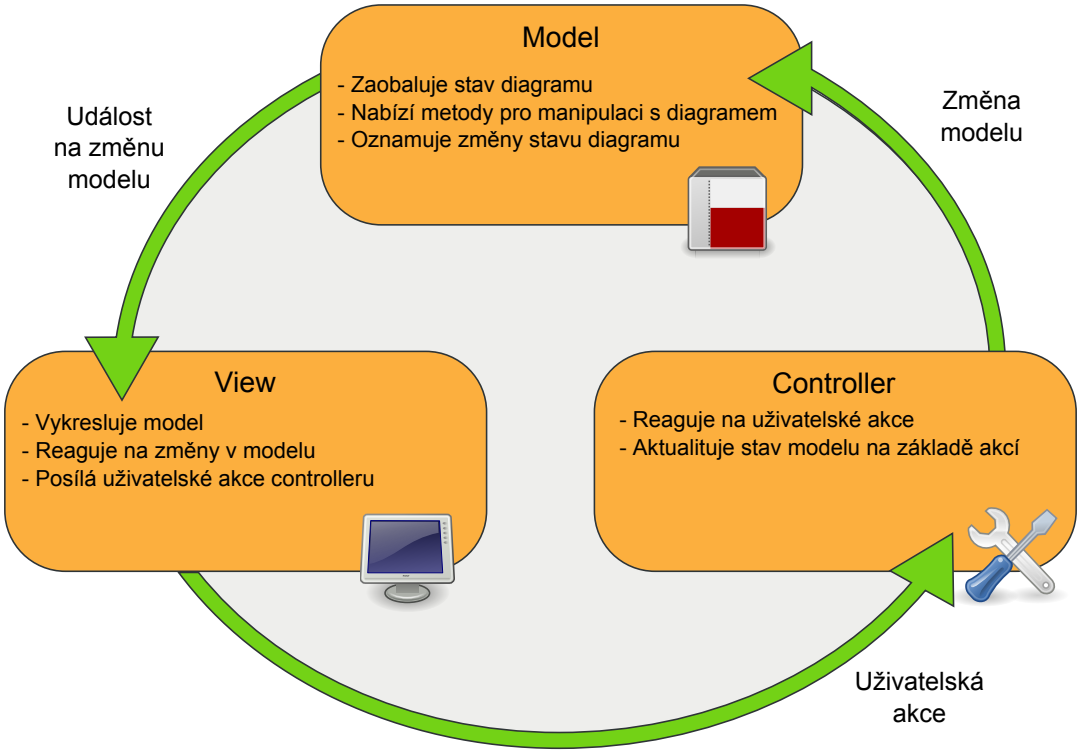
Informace o blocích systému JavaREX jsou automaticky načítány přímo z knihovny tříd. Z diagramu je vytvořena úloha JavaREX a provedena simulace, jejíž výsledky se graficky zobrazí.

3.1. Implementace editoru

Samotný editor byl hlavní a nejrozsáhlejší prací na bakalářské práci. Bylo potřeba vyřešit problém zobrazování bloků a spojnic a uživatelskou interakci s jednotlivými bloky. Pro návrh byl použit návrhový vzor Model-View-Controller (MVC). Samostatným problémem, řešeným v práci je pak systém plánování a správy spojnic. Jednoduchý způsob

kreslení spojnic se nechová příliš uspokojivě, např. vede spojnici přes již nakreslený blok, tyto problémy bylo třeba vyřešit.

Editor je v programu rozdělan na modelovou, zobrazovací a editační část v duchu návrhovému vzoru Model-View-Controller (MVC) [14] viz obr. 3.



Obrázek 3: Schéma použití MVC v editoru

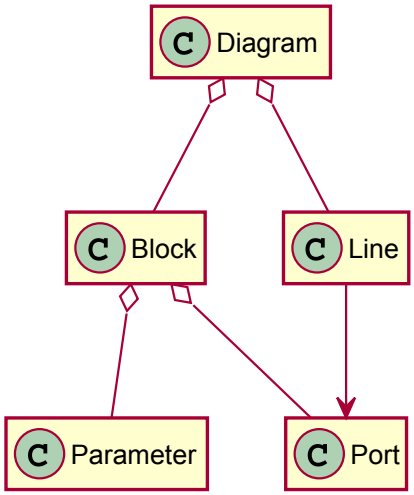
Model reprezentuje logickou strukturu diagramu.

View obsahuje zobrazovací komponenty

Controller obsahuje pomocné třídy pro uživatelskou editace modelu

3.1.1. Model

Model reprezentuje logickou strukturu diagramu. Základem modelu je třída `Diagram`, která slouží jako kontejner pro bloky (třída `Block`) a spojnice (třída `Line`). Každý blok je navíc složen z kolekcí portů a parametrů. Struktura modelových tříd je znázorněná na obr. 4.

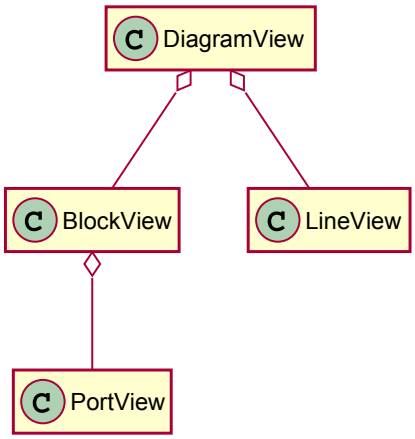


Obrázek 4: Diagram hlavních tříd modelu

Třídy modelu vedle prostého uchovávání dat zajišťují také jejich konzistenci, tj. správné provázání jednotlivých datových objektů.

3.1.2. View

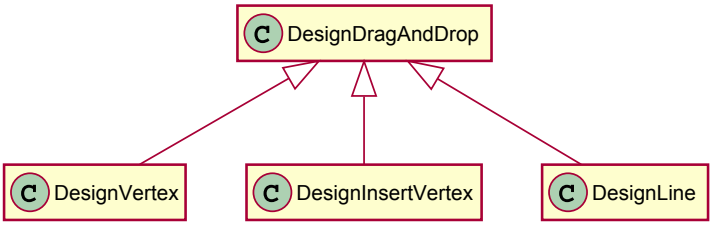
View reprezentuje grafickou podobu diagramu. Všechny třídy z této části jsou odvozeny od některé třídy JavaFX `Node` a je tedy možné je přímo umístit do grafu scény. Pro každou datovou třídu, která je v editoru zobrazována, existuje view třída se stejným názvem s přidanou koncovkou `View`. Instance view tříd se připojují na objekty modelu a reagují na jejich změny. Struktura modelových tříd je na obr. 5.



Obrázek 5: Diagram hlavních tříd viewu

3.1.3. Controller

Controller slouží pro uživatelskou interakci s modelem. Každá třída controlleru odpovídá jednomu ovládacímu gestu (tj. vytváření spojnice tažením, úprava spojnice).



Obrázek 6: Diagram hlavních tříd controlleru

Třídy controlleru dostávají informace od view objektů a mění stav modelových objektů.

DesignDragAndDrop základní třída pro akci tažení. Reaguje na tažení myši a na uvolnění tlačíka.

DesignVertex třída pro přesun vrcholu spojnice.

DesignInsertVertex třída pro přidání vrcholu na hranu spojnice.

DesignLine třída pro vytváření nové spojnice. Algoritmus plánování je uveden v následujícím textu

3.1.4. Plánování spojnice

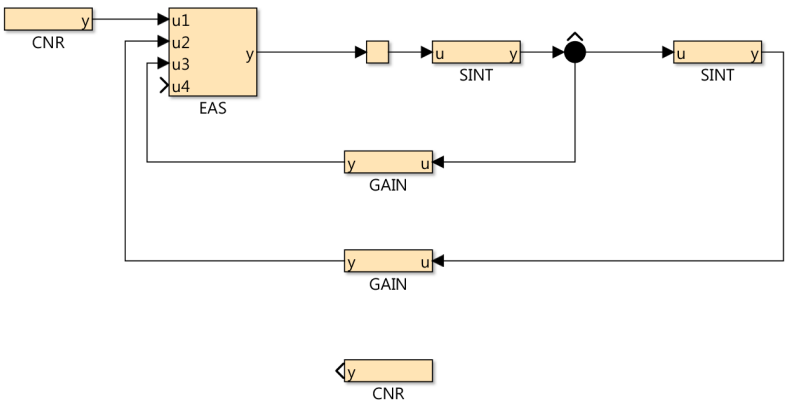
Nejsložitějším problémem v návrhu editoru bylo plánování spojnic. Nejjednodušší algoritmus přímého kreslení byl zavržen pro jeho neefektivitu. Autor prostudoval několik systémů plánování spojnic v editorech Matlab/Simulink, Scilab/Xcos a samozřejmě RexDraw. Na základě zjištěných informací byly empiricky určeny vlastnosti „kvalitní" spojnice:

- Nesmí protínat bloky
- Nesmí překrývat ostatní spojnice
- Musí mít minimální délku a minimální počet ohybů

Nakonec byl zvolen algoritmus vycházející z článku [15]. Algoritmus uvedený v článku je výhodný z důvodu, že umožňuje plánování tak aby se vyhnul blokům a spojnicím.
Následuje popis principu algoritmu. Algoritmus lze rozdělit do třech hlavních kroků.

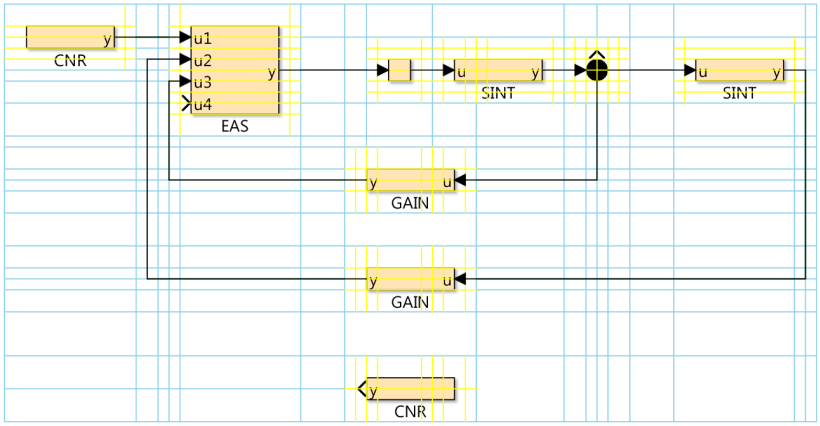
1. Vytvoření grafu viditelnosti
2. Nalezení cesty s nejmenší cenou

3. Estetická úprava spojnice



Obrázek 7: Výchozí situace

Pro vytvoření grafu viditelnosti určíme prostor zabraný jednotlivými bloky. Následně ze všech rohů každého bloku vedeme horizontální i vertikální hranu, dokud nenarazíme na prostor obsazený blokem, nebo na konec zkoumané oblasti. Obdobně postupujeme pro každý port, kdy vedeme jedinou hranu ve směru portu, a pro každý vrchol všech spojnic, kdy vedeme dvě horizontální a dvě vertikální hrany v těsném sousedství vrcholu. Výsledkem je graf z obrázku 8



Obrázek 8: Znázorněný graf viditelnosti

Nyní hledáme v grafu cestu mezi propojovanými porty. Pro nalezení cesty je použit algoritmus prohledávání stavového prostoru s minimální cenou. Stav je určen jako uspořádaná trojice (x, y, směr). Výchozí stav je pozice a směr zdrojového portu a cílový stav pozice a opačný směr cílového portu.

Pro procházení zavádíme tři operátory:

CW otočení o 90° doprava

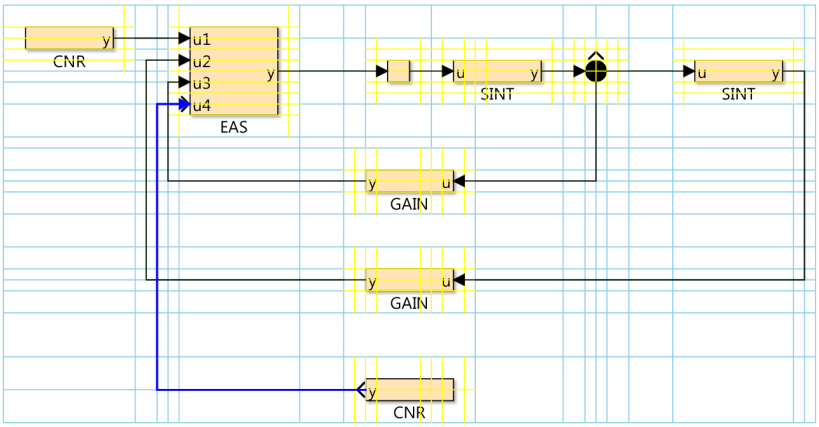
CCW otočení o 90° doleva

FORWARD přechod přes hranu ve směru natočení

Ohodnocení jednotlivých operátorů bylo stanoveno empiricky a je uvedeno v tabulce 3

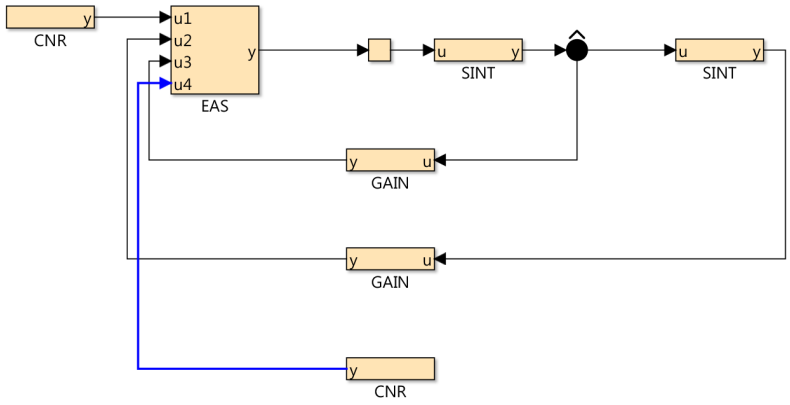
Otočení	1000
Přechod přes volnou hranu	1× délka hrany
Přechod přes existující spojnici	100× délka hrany
Přechod přes blok	10000× délka hrany

Tabulka 3: Ohodnocení operátorů

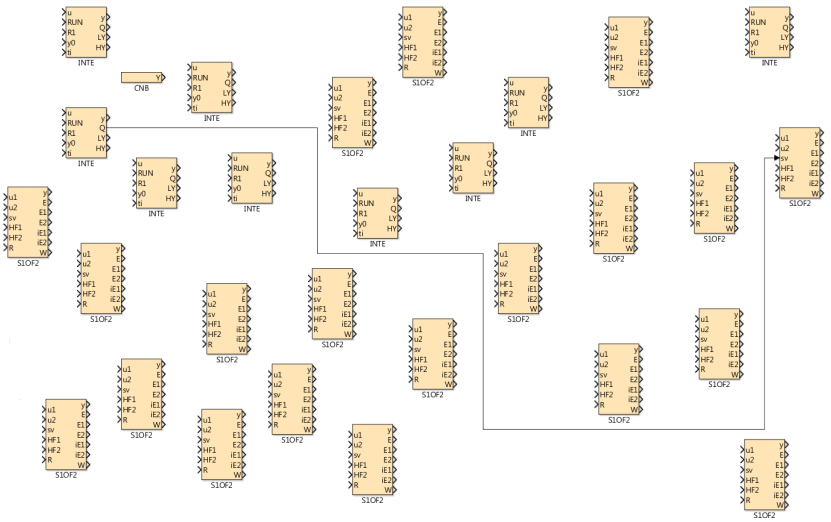


Obrázek 9: Nalezená spojnice s minimální cenou

Z vytvořené spojnice jsou na závěr odstraněny všechny přebytečné vrcholy.



Obrázek 10: Výsledný diagram



Obrázek 11: Příklad automaticky naplánované spojnice v komplexním diagramu

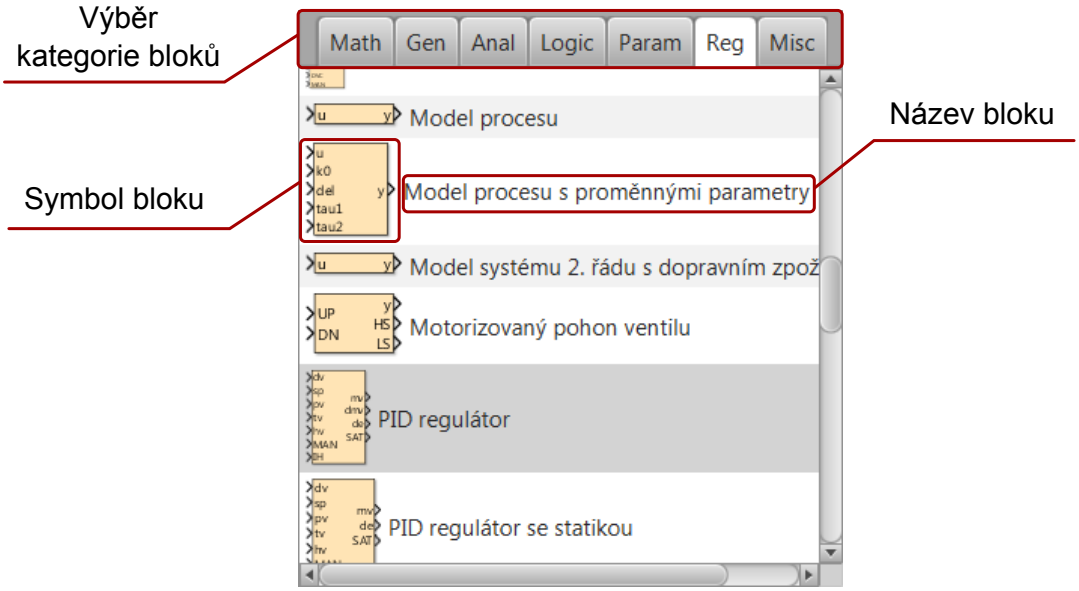
3.1.5. Ukládání diagramu

Aby bylo možné uchovávat vytvořené diagramy, byl navržen jednoduchý strukturovaný formát souboru. Formát je založen na technologii XML, která definuje standardní podobu značkovacího jazyka pro popis strukturovaných dat [16], což umožňuje snadné sdílení mezi různými programy a platformami. Příklad diagramu a jeho XML reprezentace je na obrázku 22 v příloze.

3.2. Ostatní komponenty

3.2.1. Knihovna bloků

Knihovna bloků slouží pro výběr nového bloku pro vytvoření. Bloky jsou v knihovně umístněny v záložkách `Tab` podle kategorií. Jednotlivé položky obsahují symbol bloku a název. Symbol bloku je realizovaný pomocí grafické komponenty `BlockView`, kterou knihovna sdílí s editorem.



Obrázek 12: Knihovna bloků

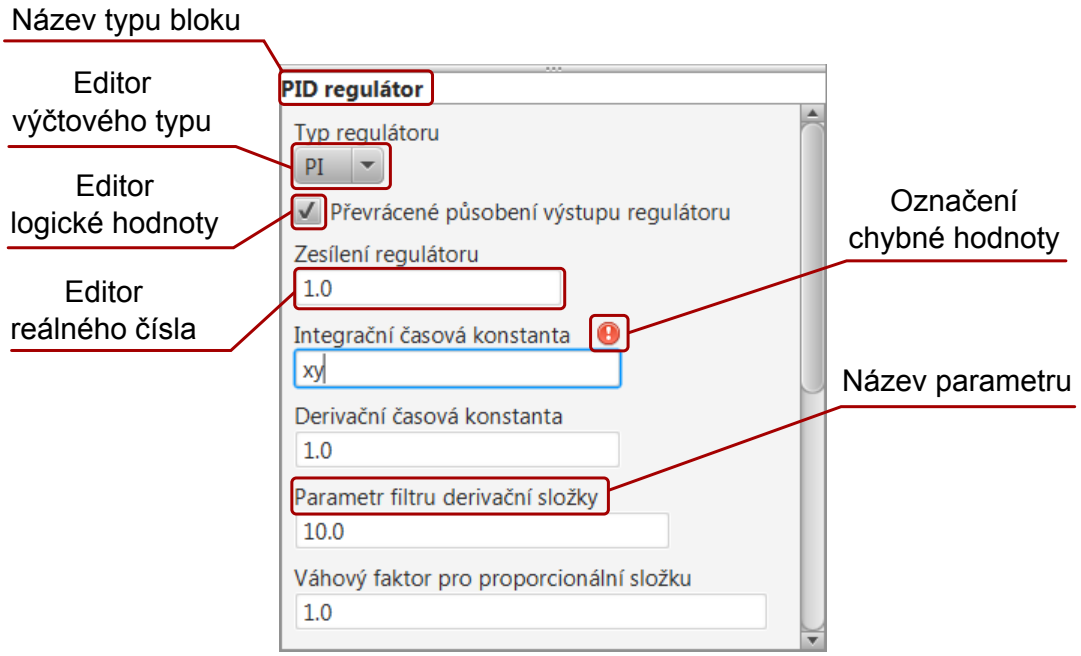
3.2.2. Editor parametrů

Editor parametrů slouží pro úpravy hodnot vnitřních parametrů jednotlivých bloků. Každý parametr má určen svůj datový typ, který definuje obor možných hodnot. Jednotlivé datové typy jsou uvedeny v tabulce 4.

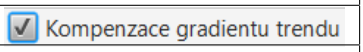
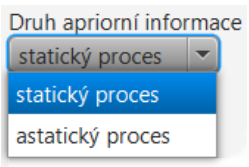
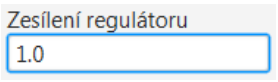
Kategorie	Datový typ	Rozsah hodnot
Logická hodnota	XBOOL	$\{TRUE, FALSE\}$
Celé číslo	XSHORT XLONG XLARGE	-32.768 .. 32.767 $(-2^{15}$ .. $2^{15}-1)$ $-2.147.483.648$ .. $2.147.483.647$ $(-2^{31}$ .. $2^{31}-1)$ -2^{63} .. $2^{63}-1$
Kardinální číslo	XBYTE XWORD XDWORD	0 .. 255 $(0$ .. $2^{8}-1)$ 0 .. 65.535 $(0$ .. $2^{16}-1)$ 0 .. 4.294.967.295 $(0$ .. $2^{32}-1)$
Reálné číslo	XFLOAT XDOUBLE	Single-precision floating-point Double-precision floating-point
Časový údaj	XTIME	Double-precision floating-point
Chybový kód	XERROR	-32.768 .. 32.767 $(-2^{15}..2^{15}-1)$
Textový řetězec	XSTRING	

Tabulka 4: Datové typy systému REX

Aby bylo možné hodnoty parametrů efektivně zadávat byla vytvořena sada speciálních editorů pro jednotlivé datové typy. V případě chybného zadání hodnoty parametru je v editoru zobrazena ikona varování.



Obrázek 13: Editor parametrů bloku

Editor	Datový typ	Ukázka
Zatrhávací pole (`CheckBox`)	XBOOL	
Výběr z výčtu (`ComboBox`)	XSHORT XLONG XLARGE XBYTE XWORD XDWORD	
Textové pole (`TextField`)	XSHORT XLONG XLARGE XBYTE XWORD XDWORD XFLOAT XDOUBLE XTIME XERROR XSTRING	

Tabulka 5: Editory pro datové typy

3.3. JavaREX

Jedná se o knihovnu vygenerovanou z algoritmů psaných v jazyce C++ pro REX. Knihovna tak obsahuje většinu bloků ze standardního řídicího systému REX implementovanými v jazyce Java a je snadné tyto bloky implementovat ve webovém prohlížeči s pluginem pro spouštění Java aplikací.

V současnosti se JavaREX využívá pro prezentaci pokročilých algoritmů systému REX v okně webového prohlížeče. Současné prezentace jsou, ale předem připravené a upravovat je možné pouze některé parametry.

V bakalářské praci je třeba vyřešit načtení bloků z knihovny do editoru a sestavení spustitelného souboru vygenerovaného na základě konkrétního schematu tzv. task systému JavaREX.

3.3.1. Integrace do editoru

Pro použití knihovny JavaREX v editoru bylo nejprve nutné vyhledat jednotlivé třídy bloků. Ty byly získány pomocí open-source Java knihovny Reflections. [17] Tato knihovna umožňuje vyhledávat třídy na základně anotací, nebo podle třídy jejich předka a to i ty, které ještě nebyly zavedeny do aplikace. Pro potřeby editoru byl použit druhý způsob vzhledem k faktu, že všechny třídy REX bloků dědí od třídy `javaRexCore.Block`.

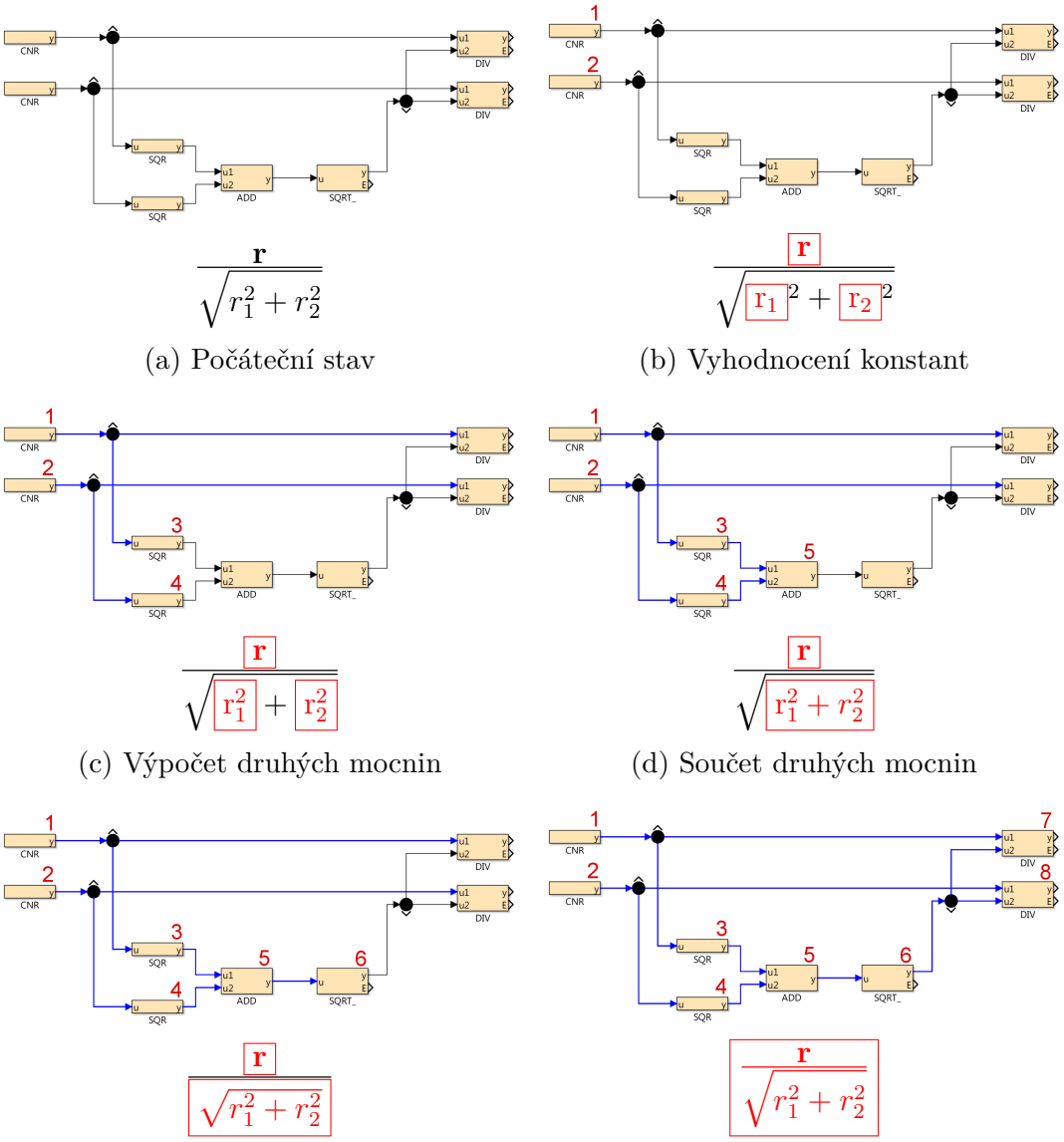
Jednotlivé bloky REXu jsou vytvářeny pomocí reflexe jazyka Java [18], což umožňuje vytvářet instance tříd, které nejsou známé při kompilaci. Díky této technice pro aktualizaci knihovny bloků stačí k aplikaci připojit novou knihovnu JavaREX.

3.3.2. Vytvoření tasku z diagramu

Na diagram bloků lze pohlížet jako na orientovaný graf, kde hrany určují podmíněnost vykonávání. Blok může být spuštěn teprve tehdy, když byly spuštěny všechny bloky připojené na jeho vstupy. Jedná se tedy o problém seřazení prvků acyklického grafu.

Řazení acyklického grafu bude demonstrováno na výpočtu normalizace 2-rozměrného vektoru:

$$\mathbf{n} = \frac{\mathbf{r}}{\|\mathbf{r}\|} = \frac{\mathbf{r}}{\sqrt{r_1^2 + r_2^2}}$$



(a) Počáteční stav

(b) Vyhodnocení konstant

(c) Výpočet druhých mocnin

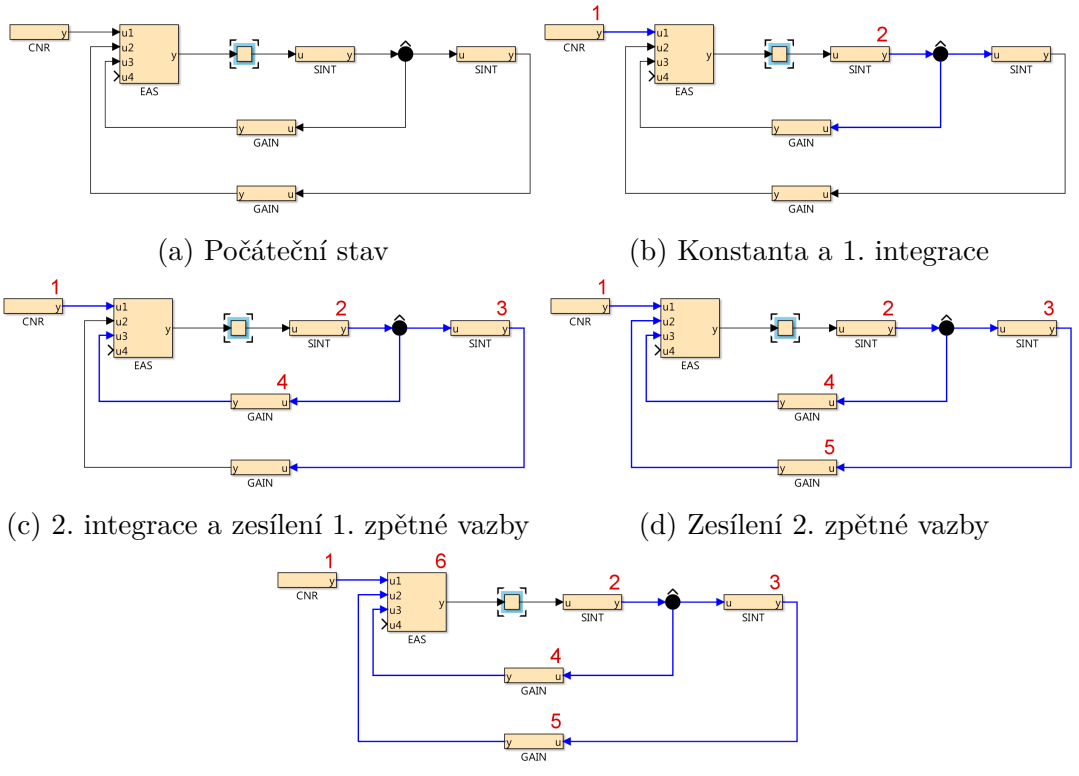
(d) Součet druhých mocnin

(e) Výpočet odmocniny

(f) Dokončení výpočtu

Obrázek 14: Normalizace 2-rozměrného vektoru

Reálné kybernetické systémy ovšem obvykle obsahují minimálně jednu zpětnou vazbu, což se v orientovaném grafu diagramu projevuje jako cyklus. V tomto případě musí dojít k úpravě grafu tak, aby opět vznikl acyklický graf a bylo možné použít uvedený postup. V systému REX je toto řešeno pomocí speciálního bloku LPBRK (Rozpojení zpětné vazby) [1], který potlačí závislost mezi spojnicemi na svém vstupu a výstupu.



(a) Počáteční stav

(b) Konstanta a 1. integrace

(c) 2. integrace a zesílení 1. zpětné vazby

(d) Zesílení 2. zpětné vazby

(e) Finální součet signálů

Obrázek 15: Výpočet lineárního systému 2. řádu (označený blok je LPBRK)

Vlastní task je vytvořen pomocí postupného volání JavaREX API dle algoritmu 1.

Algoritmus 1 Vytvoření JavaREX tasku

```
sortedBlocks ← sort(blocks)
for all block in sortedBlocks do
    block.createRexObject()
    declareBlock(block)
end for
allocateMemory()
for all block in sortedBlocks do
    addBlock(block)
    for all param in block.parameters do
        block.setInPar(param)
    end for
    block.setPeriod()
    block.init()
end for
for all c in portConnections do
    connectBlock(c.srcBlock, c.srcPort, c.dstBlock, c.dstPort)
end for
```

3.3.3. Spouštění JavaREX tasku

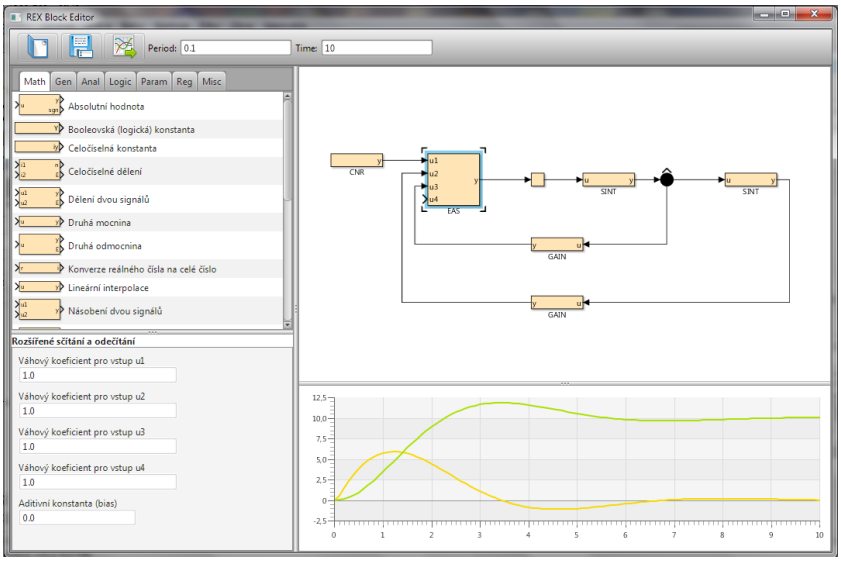
Spoštění tasku a vytváření grafu je prováděno offline, kdy je celá simulace vypočítána v jednom cyklu (algoritmus 2) a kompletní graf je ihned zobrazen.

Algoritmus 2 Provádění JavaREX tasku

```
task ← new RexTask()
task.init()
for t ← 0 to time by period do
    task.main()
    for all watched ports do
        addToChart(port, t, task.getPortValue(port))
    end for
end for
```

4. Uživatelský manuál

Uživatelský manuál se skládá z popisu aplikace a příkladu krok za krokem jak vytvořit schema systému JavaREX. Vlastní aplikace je na obr. 16.



Obrázek 16: Obrazovka editoru JavaREX s popisem oken

Samotná aplikace editoru se skládá ze 4 oken, u kterých lze měnit velikost viz 16. Části jsou hlavní lišta, prohlížeč knihovny bloků, editor parametrů, graf výstupu a vlastní okno editoru.

Hlavní lišta slouží k základním operacím s programem. Obsahuje tlačítka pro ukládání a načítání souborů, tlačítko pro provedení simulace a editovatelná pole pro nastavení délky simulace a periody simulace.

Editor diagramu slouží k editaci vlastního diagramu.

Knihovny bloků obsahují seřazené všechny podporované bloky.

Editor parametrů zobrazuje parametry aktuálně vybraného bloku. Umožňuje jejich editaci.

Graf zobrazuje graf po provedení simulace.

4.1. Ovládání editoru

Editor poskytuje základní funkce pro práci se soubory, editaci schematu a následné grafické zobrazení simulovaných hodnot.

Hlavní lišta

Horní lišta obsahuje tlačítka (zleva) načtení souboru, uložení souboru a spuštění simulace. Následuje editační pole pro nastavení periody vzorkování a další je nastavení délky simulace.

Systém ukládá a soubory typu .xml. Pro načtení souboru stiskneme tlačítko načtení souboru na horní liště. Objeví se dialogové okno kde požadovaný sobor nalezneme a načteme. Pro uložení vybereme tlačítko uložení souboru a postupujeme analogicky.

Pokud máme připravený diagram simulace nastavíme krok simulace a délku simulace a tlačítkem spuštění simulaci spustíme.

Tvorba diagramu

Diagram se skládá z bloků systému JavaREX, bloky a jejich vlastnosti odpovídají chování standardních bloků systému REX.

Pro vložení bloku do schematu vybereme v knihovně bloků požadovaný blok a dvojklikem do pole editoru blok umístíme. Blok můžeme kliknutím a tažením přesouvat libovolně po schematu. Rotaci bloku provedeme stiskem klávesy R, rotaci můžeme opakovat a tak blok natočit do libovolné polohy.

Nastavení parametrů u bloku provádím výběrem bloku, levým tlačítkem myši a editací příslušných parametrů v editoru parametrů. Změny parametrů jsou aplikovány okamžitě po změně.

Propojení bloků provádíme výběrem portu odkud chceme vést spojnici a kliknutím a tažením k cílovému portu táhneme spojnici, která se automaticky vytvoří. Pokud nejsme s podobou spojnice spokojeni můžeme ji editovat, najetím na spojnici ve zlomu se zobrazí ikona zlomového bodu spojnice a ten můžeme libovolně přesunout do požadované polohy. Najetím na spojnici v místě bez zlomů, vytvoříme nový zlomový bod a ten opět můžeme přetahnout na požadované umístění.

Zvláštním způsobem je řešeno rozvětvení spojnice. To se provádí umístěním speciálního bloku, který rozvětvení řeší. Tento blok umístíme dvojklikem na pravé tlačítko myši.

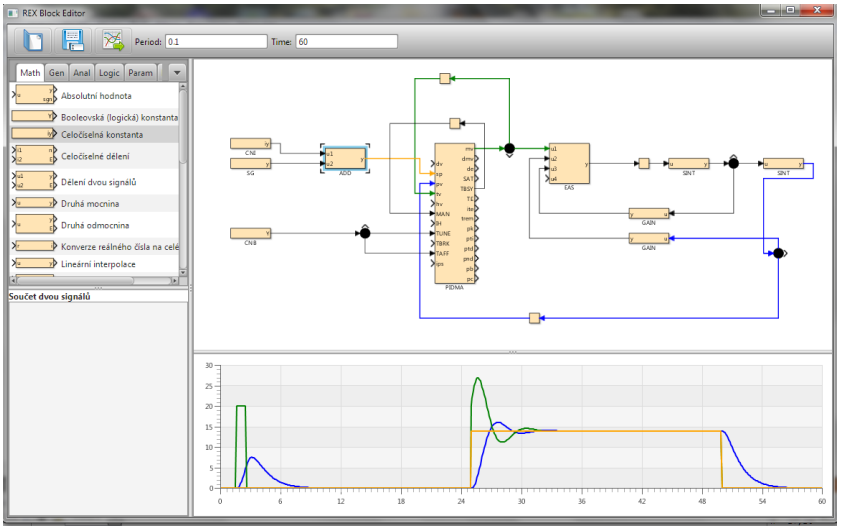
Doporučený postup na tvorbu diagramu je nejprve umístit požadované bloky a pak je prpojit pomocí spojnic. Přesouvání bloku s připojenou spojnicí v současné verzi není uspokojivě vyřešeno a vzniká pak nutnost upravit spojnice ručně.

Spouštění simulace

Vytvořené schema je možné spustit jako simulaci a v grafu zobrazit sledované hodnoty. Před spuštěním samotné simulace vybereme v diagramu spojnice odpovídající signálům, které chceme sledovat. Nejprve vybereme spojnici kterou chceme sledovat a stiskneme klávesu W (*watch*), spojnice se zvýrazní nějakou barvou. Toto můžeme udělat pro každou spojnici, kterou chceme sledovat a nebo vybereme více spojnic najednou a stiskneme tlačítko W a všechny vybrané spojnice se obarví. Nakonec stiskneme na hlavní liště tlačítko simulace a v grafu se zobrazí výsledné časové průběhy v barvách odpovídajících zvýraznění.

4.2. Modelový příklad

Jako jednoduchý modelový příklad je připravena simulace systému druhého řádu řízeného momentovým regulátorem s autotunereme, simulaci je možno najít na přiloženém CD v složce Examples. Připravená simulace je vidět na obr. 17



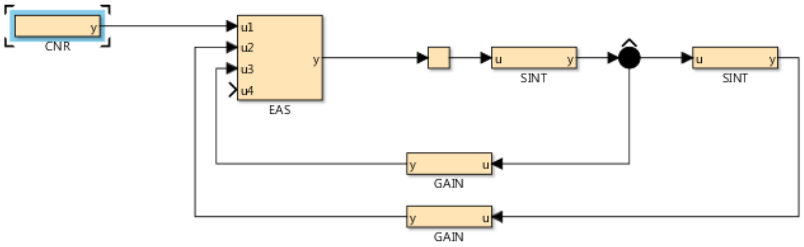
Obrázek 17: Obrazovka editoru JavaREX se simulačním schematem regulátoru s momentovým autotunerem

Následuje postup vytvoření. Nejprve je potřeba vytvořit samotný systém druhého řádu. Spustíme editor a v okně knihovny bloků vybereme záložku Reg a najedeme na Jednoduchý integrátor. Dvojklikem je vložíme do editoru parametrů. Klepneme na vložené integrátory a v editoru parametrů nastavíme hodnoty saturací na 1000 resp. -1000 pro horní a dolní mez. Dál vložíme bloky ze záložky math 2x násobení konstatou, blok pro rozšířené sčítání a odečítání a reálnou konstantu. Z knihovny Misc vložíme blok pro řešení algebraických smyček Loop break.

Mezi integrátory vložíme jeden blok pro rozvětvení spojnicí (dvojklikem pravým tlačítkem myši). Schema propojíme podle obrázku 18. Jednoduchým tažemním myši od výchozího do cílového portu. Porty konstant můžeme ještě před tažením spojnic otočit pomocí stisku klávesy R. Nastavíme hodnoty násobení konstantou, první hodnotu nastavíme na hodnoutu -2 a druhou nastavíme na hodnoutu -1. Což odpovídá systému s přenosem:

$$F(p) = \frac{1}{p^2 + 2p + 1}$$

Po zapojení nastavíme na hlavní liště dobu běhu na 20s a periodu vzorkování na 0.1s. Pro zobrazení grafu vybereme spojnici s výstupem systému (vystupující z druhého integrátoru) a stiskem klávesy W ji označíme pro zobrazení v grafu (Označení zrušíme výběrem označené spojnice a stiskem klávesy U (*unwatch*)). Po stisku tlačítka pro



Obrázek 18: Jednoduchý systém druhého řádu

spuštění na hlavním panelu bychom měli vidět výsledný průběh signálu odpovídající přechodové charakteristice systému druhého řádu s přenosem $F(p)$.

Nyní máme hotový systém druhého řádu, můžeme přejít k nastavení regulátoru s momentovým autotunerem. Z knihovny Reg vložíme PID regulátor s momentovým autotunerem, nastavíme jeho parametry, Doba odhadu gradientu na 0, doba odhadu šumu na 0, amplituda pulsu na 20 a Práh pro ukončení pulsu na 5. Z knihovny Gen vložíme blok Signálový generátor, nastavíme jeho parametry na typ generovaného signálu Čtverec, Amplituda 7, Frekvence 0.02 a fázový posuv 180. Z knihovny math vložíme ještě blok Reálná konstanta a jeho parametr nastavím na 7. Jeho výstup sečteme s výstupem generátoru. Schema zapojíme podle obrázku 17. Zpětné vazby na regulátoru slouží k vysledování integrační složky a ke spuštění regulátoru po provedení ladicího experimentu.

Vybereme příslušné spojnice s výstupem systému a s řízením (výstup z regulátoru) čas simulace na hlavní liště nastavíme na 60s a periodu vzorkování na 0.1s. Systém spustíme a získáme výsledné grafy jako na obrázku 17.

V případě problémů, je soubor s příkladem na přiloženém CD ve složce Examples, kde se nachází i další příklady použití editoru systému JavaRex.

5. Závěr

Práce prokázala, že zvolená technologie umožňuje nasazení pro editor diagramů. Tento editor by mohl být po úpravách použitý např. pro prezentaci bloků systému REX v online editoru. A nebo pro sofistikovanější virtuální laboratoře, použitelné ve výuce a školení uživatelů systému REX.

Všechny body zadání práce byly splněny. Autor prozkoumal možné způsoby a programovací jazyky pro editor blokových diagramů systému JavaREX. Autor implementoval editor diagramů, který je možné spustit ve webovém prohlížeči a po drobnějších úpravách může být editor pro prezentační a výukové účely např. jako virtuální laboratoř. Editor je navíc přímo propojený s knihovnou JavaREX a podporuje tak přímou simulaci vytvořených diagramů. Výstupy simulace je možné zobrazovat jako graf. V následujícím textu, bude podrobněji popsán autorův přínos.

Autor také zanalyzoval možnosti jednotlivých softwarových technologií, vzhledem k tomu, že se jedná o editor, který vykresluje bloky byl pro posouzení navržen testovací program vykreslující velké množství bloků a subjektivně byla vybrána nejrychlejší technologie. Na základě tohoto testu byla volena knihovna JavaFX, která je navíc nejlépe vybavena pro použití v grafickém editoru. Více informací o metodice výběru vhodné technologie je v části Volba vhodné technologie 2

S použitím vybranné technologie bylo přistoupeno k návrhu editoru. A byly řešeny metody kreslení a zobrazování diagramů. Pro tyto účely musel být navržený model zobrazovaných dat (prvky diagramu), byly navrženy způsoby zobrazování a interakce s uživatelem. Tyto části byly implementovány v jazyce Java a vznikl editor.

Významnou součástí editoru blokových schemat je systém propojování portů. Jednoduché algoritmy kreslení spojnic nesplňují požadavky na intuitivní ovládání a proto byl implementován systém plánování spojnic na základě [15]. Systém kreslení spojnic by zároveň potřeboval další vývoj, protože rozvětvení spojnice je řešeno speciálním blokem a přesouvání spojnic při jejich editaci není optimální. Srovnání použitého plánovacího algoritmu se dalšími editory (Simulink, Xcos, RexDraw) je ilustrováno na obrázku 21 v příloze.

Editor bylo potřeba propojit se systémem JavaREX. Byla vytvořena třída, která umožňuje automaticky vytvářet knihovnu bloků editoru na základě tříd systému JavaREX. To umožní měnit knihovnu JavaREX při zachování funkčnosti editoru.

Dalším úkolem v bakalářské práci bylo spouštění vytvořených blokových schemat. Pro spouštění systému bylo třeba z diagramu systému JavaREX vytvořit acyklický graf a ten potom samostatně spouštět.

Byla vytvořena aplikace, která integruje editor schemat a několik dalších komponent, které společně umožňují vytvářet, ukládat a načítat bloková schemata. Samozřejmě také editovat a spouštět jako simulaci. Výsledek simulace je možné zobrazit jako Graf. Digramy jsou ukládány jako XML soubory.

Pro potřeby webové aplikace bylo umožněno aplikaci zakomponovat do webové stránky. Aplikaci je možné spouštět v prohlížečích vybavených Java pluginem.

Zdrojový kód práce je na přiloženém CD spolu se spustitelnou aplikací a ukázkou zakomponování editoru do webové stránky.

5.1. Další práce

Další práce na editoru by se mohla zaměřit na systémy vytváření spojnic. Především systém ruční editace není kompletně dotažený, spojnice je možné přesouvat pouze ve vrcholu, ne na hraně. Totéž platí pro přesouvání již zapojených bloků, kdy se spojnice nepřesouvají intuitivně. Tyto problémy jsou však řešeny v jiných editorech (např. Simulink) uspokojivě.

V editoru by se navíc chybí možnost simulace v reálném čase a možnost krokování simulace. Tato možnost by se dala poměrně snadno doimplementovat.

Reference

[1] REX Controls, "Funkční bloky systému REX, Referenční příručka," 2013. [Online; accessed 10-August-2013].

[2] Wikipedia, "JavaScript," 2013. [Online; accessed 15-July-2013].

[3] Wikipedia, "ECMAScript," 2013. [Online; accessed 15-July-2013].

[4] World Wide Web Consortium, "HTML5," 2012. [Online; accessed 15-July-2013].

[5] Google, "The GWT Release Notes," 2013. [Online; accessed 15-July-2013].

[6] w3resource.com, "Java-support Statistics," 2013. [Online; accessed 15-July-2013].

[7] Google, "Android APIs," 2013. [Online; accessed 15-July-2013].

[8] Oracle, "Java EE at a Glance," 2013. [Online; accessed 15-July-2013].

[9] Wikipedia, "Abstract Window Toolkit," 2013. [Online; accessed 15-July-2013].

[10] Wikipedia, "Standard Widget Toolkit," 2013. [Online; accessed 15-July-2013].

[11] J. Žára, B. Beneš, J. Sochor, and P. Felkel, *Moderní počítačová grafika*. Computer Press, 2004.

[12] Oracle, "JavaFX Roadmap," 2013. [Online; accessed 15-July-2013].

[13] Sun Microsystems, "Sun delivers next version of the Java platform," 1998. [Online; accessed 15-July-2013].

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1 ed., 11 1994.

[15] M. Wybrow, K. Marriott, and P. J. Stuckey, "Orthogonal connector routing.," in *Graph Drawing* (D. Eppstein and E. R. Gansner, eds.), vol. 5849 of *Lecture Notes in Computer Science*, pp. 219–231, Springer, 2009.

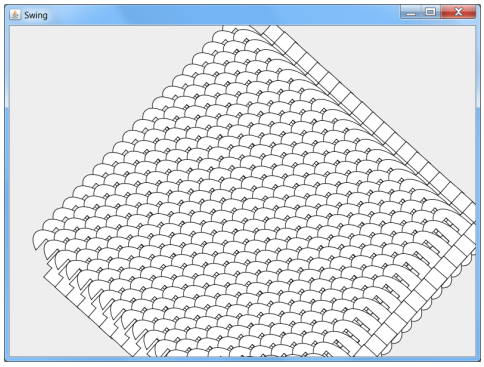
[16] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fifth Edition)," 2008. [Online; accessed 15-July-2013].

[17] ronmamo, "Reflections library project page," 2013. [Online; accessed 15-July-2013].

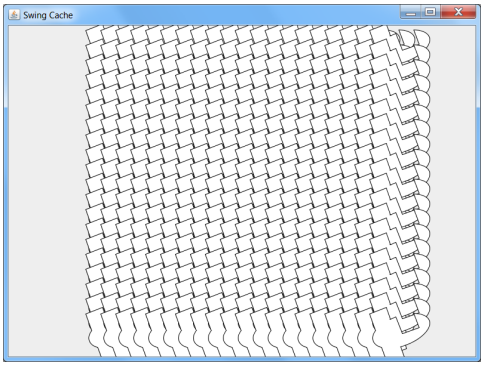
[18] Oracle, "Trail: The Reflection API," 2013. [Online; accessed 15-July-2013].

[19] MathWorks, "Documentation Center, Simulink," 2013. [Online; accessed 10-August-2013].

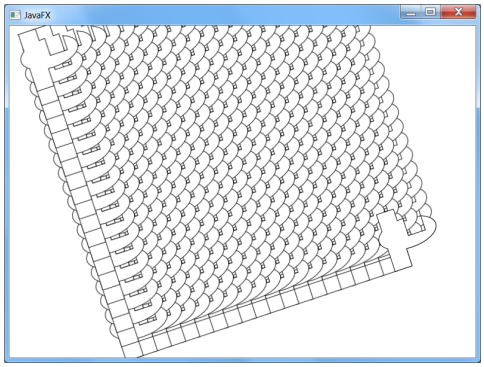
A. Příloha



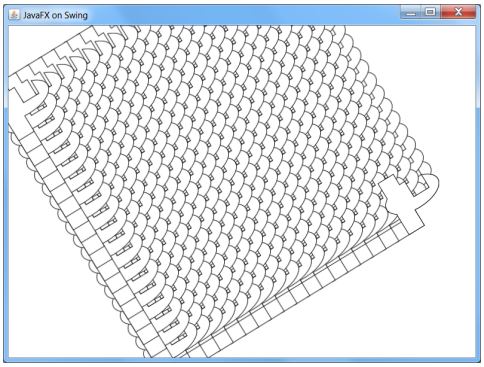
(a) Swing



(b) Swing s využitím cache

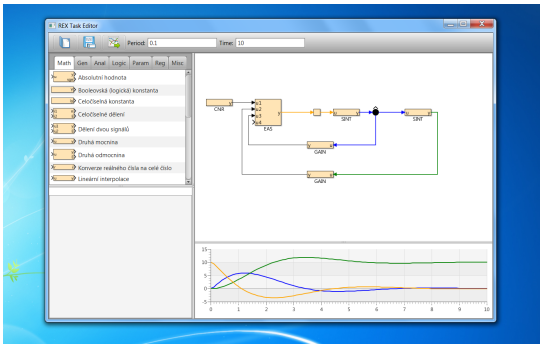


(c) JavaFX

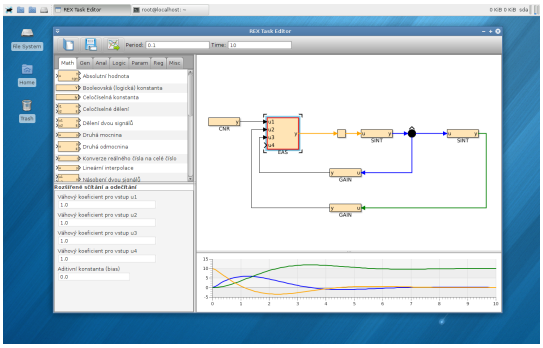


(d) JavaFX uvnitř Swing kontejneru

Obrázek 19: Testovací aplikace na porovnání grafického výkonu

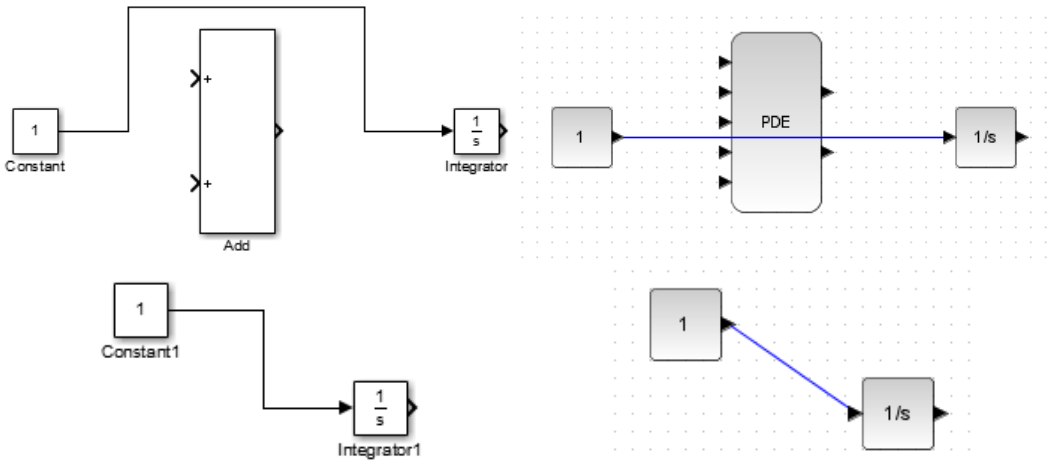


(a) Windows

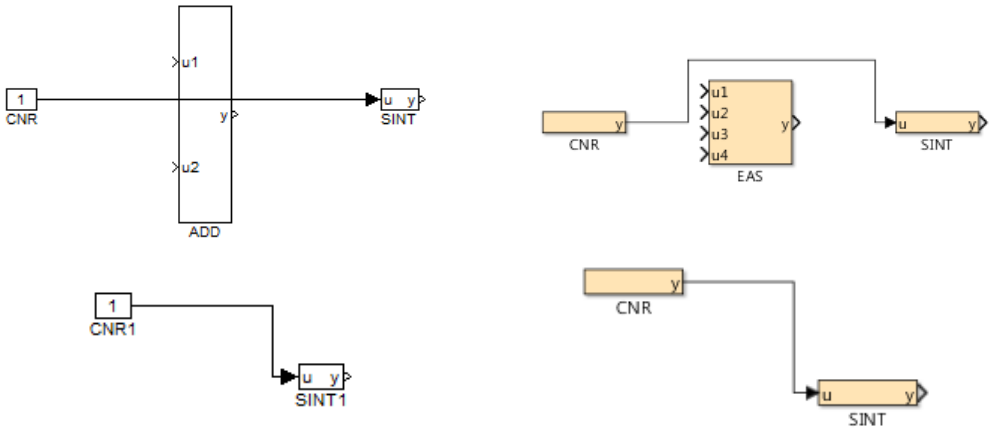


(b) Linux

Obrázek 20: Ukázka aplikace na různých platformách

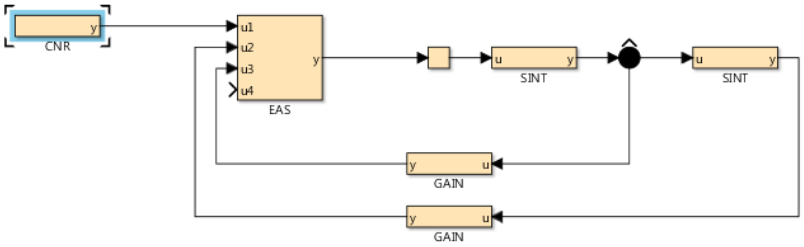


(a) Simulink (b) Xcos



(c) RexDraw (d) Tento editor

Obrázek 21: Srovnání plánovacích algoritmů




```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<diagram>
  <block bounds="420 190 80 20" ref="rex.reg.SINT">
    <p name="ti">1.0</p>
    <p name="y0">0.0</p>
    <p name="ymax">1000</p>
    <p name="ymin">-1000</p>
  </block>
  <block bounds="640 190 80 20" ref="rex.reg.SINT">
    <p name="ti">1.0</p>
    <p name="y0">0.0</p>
    <p name="ymax">1000.0</p>
    <p name="ymin">-1000</p>
  </block>
  <block bounds="180 160 80 80" ref="rex.math.EAS">
    <p name="a">1.0</p>
    <p name="b">1.0</p>
    <p name="c">1.0</p>
    <p name="d">1.0</p>
    <p name="y0">0.0</p>
  </block>
  <block bounds="340 290 80 20" ref="rex.math.GAIN" rotation="180">
    <p name="k">-1.0</p>
  </block>
  <block bounds="340 380 80 20" ref="rex.math.GAIN" rotation="180">
    <p name="k">-1.0</p>
  </block>
  <block bounds="540 190 20 20" junction="1"/>
  <block bounds="360 190 20 20" ref="LoopBreak"/>
  <block bounds="30 160 80 20" ref="rex.math.CNR">
    <p name="ycn">10.0</p>
  </block>
  <line end="0:0" points="" start="6:1"/>
  <line end="2:2" points="160 300" start="3:1"/>
  <line end="1:0" points="" start="5:2"/>
  <line end="6:0" points="" start="2:4"/>
  <line end="2:1" points="140 390 140 190" start="4:1"/>
  <line end="2:0" points="" start="7:0"/>
  <line end="4:0" points="740 390" start="1:1"/>
  <line end="3:0" points="550 300" start="5:3"/>
  <line end="5:0" points="" start="0:1"/>
</diagram>
```

Obrázek 22: Příklad diagramu a jeho XML reprezentace