Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

# BAKALÁŘSKÁ PRÁCE

Program pro elektroniku

2022

František Pavelka

Vedoucí práce:
RNDr. Arnošt Večerka

Studijní program: Aplikovaná informatika,
kombinovaná forma

**Bibliografické údaje**

| | |
|---|---|
| Autor: | František Pavelka |
| Název práce: | Program pro elektroniku |
| Typ práce: | bakalářská práce |
| Pracoviště: | Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci |
| Rok obhajoby: | 2022 |
| Studijní program: | Aplikovaná informatika, kombinovaná forma |
| Vedoucí práce: | RNDr. Arnošt Večerka |
| Počet stran: | 55 |
| Přílohy: | 1 CD/DVD |
| Jazyk práce: | český |

**Bibliographic info**

| | |
|---|---|
| Author: | František Pavelka |
| Title: | Program for electronics |
| Thesis type: | bachelor thesis |
| Department: | Department of Computer Science, Faculty of Science, Palacký University Olomouc |
| Year of defense: | 2022 |
| Study program: | Applied Computer Science, combined form |
| Supervisor: | RNDr. Arnošt Večerka |
| Page count: | 55 |
| Supplements: | 1 CD/DVD |
| Thesis language: | Czech |

## Anotace

*Práce je zaměřena na metodiku tvorby aplikací pro elektroniku. Jsou v ní rozebrány obecné aplikace, které se touto tématikou zaobírají a jejich funkcionality. Na základě analýzy referenčních aplikací byl sestaven návrh aplikace, který byl následně implementován. V implementačních detailech se práce zaměřuje především na metodiku tvorby grafického editoru elektronických obvodů a na problematiku simulace elektronických obvodů.*

*Závěrem je zhodnocena výsledná aplikace a jsou uvedeny klady a zápory konečného návrhu. Obsahem jsou také možnosti dalšího rozvoje aplikace.*

## Synopsis

*This bachelor thesis is focused on the methodology of development of the applications for electronic. There are described general applications which focus on this topic and there are also described basic functionalities of similar applications. Based on the initial analysis was created application design which was then implemented. The implementation section of the thesis is focused on the issues of the graphic editor creation and the general topic of the electronic circuit simulation.*

*The conclusion of the thesis includes validation of the overall application design and possibilities of the future application development.*



**Klíčová slova:** simulace; elektrické obvody; grafický edito; elektronický komponent

**Keywords:** simulation; electrinic circuits; graphical editor; electronic component

Tímto bych rád poděkoval vedoucímu práce RNDr. Arnoštovi Večerkovi za vedení práce a podnětné rady.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce podpis autora

# Obsah

# Seznam obrázků



# Seznam tabulek

# 1 Úvod

Elektronika je velice rozšířeným oborem, který má dosah do mnoha aspektů běžného života. Ovšem návrh a analýza elektronických obvodů je složitou disciplínou, která vyžaduje velké množství testování, zahrnujícího změny parametrů a konfigurace součástek. Pokud by se tyto změny měly provádět na fyzickém obvodu, bylo by to v mnoha případech velmi nákladné nebo zcela nemožné. Z tohoto důvodu začaly vznikat aplikace, které by umožnily snadný návrh obvodu a změny jeho parametrů, aniž by bylo nutné ho fyzicky sestavit.

S takovými aplikacemi je možné se setkat v širokém rozmezí komplexity a požadované uživatelské odbornosti. Téměř vždy se ale jedná o grafické editory elektronických obvodů, které v sobě současně implementují možnost analogové nebo digitální simulace sestavených obvodů.

Problematika tvorby aplikace pro elektroniku v sobě zahrnuje široké spektrum témat, mezi které patří například: problematika tvorby grafických editorů, problematika reprezentace a manipulace dat nebo problematika matematického modelování elektronických komponent.

Konečná aplikace by tedy měla uživateli umožnit sestavovat elektronické obvody a prostřednictvím simulace se lépe seznámit se vzájemnými vztahy jednotlivých komponent a celkovým chováním elektronických obvodů.

# 2 Cíl práce

Cílem práce bylo:

- seznámit se s tématem aplikací pro elektroniku a s metodikou jejich tvorby,
- vytvořit návrh aplikace na základě zkoumaných referenčních aplikací,
- implementovat aplikaci, který umožní uživateli provádět návrh a analýzu elektronických obvodů a současně bude poskytovat základní informace o některých komponentách,
- v závěru zhodnotit výslednou aplikaci a analyzovat její další možný rozvoj

# 3 Průzkum existujících řešení

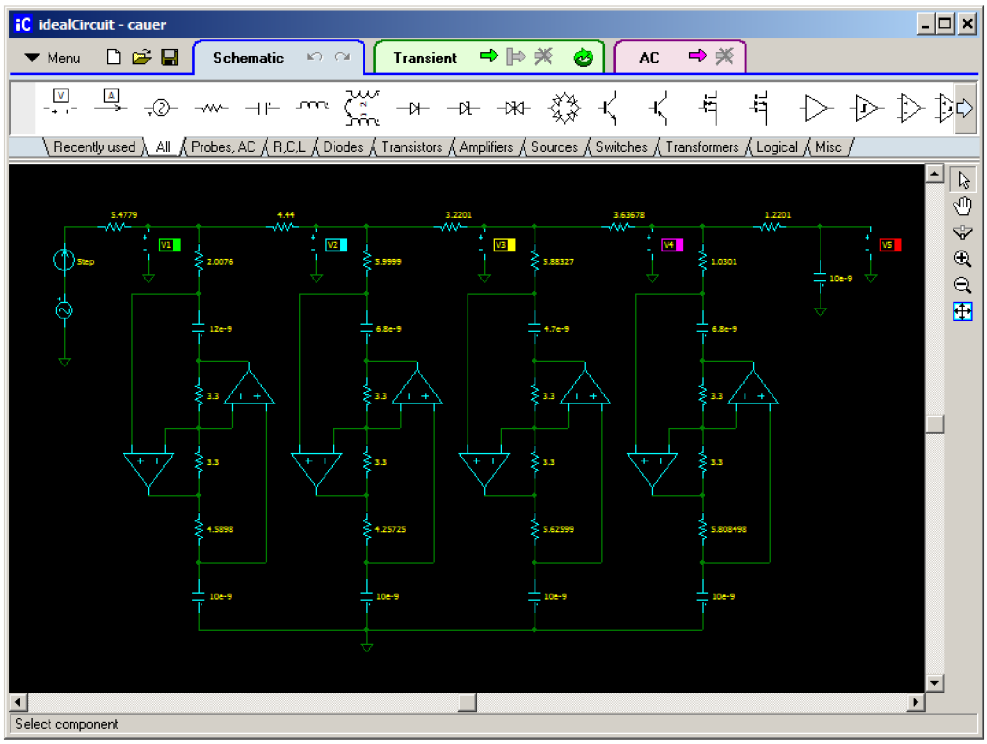
## 3.1 Určení funkcí aplikace pro elektroniku

Výsledný software by měl umožňovat snadnou tvorbu a editaci elektronických obvodů. Aplikace by měla být použitelná i bez předchozích zkušeností s podobným typem programu a bez hlubších znalostí principů elektrotechniky.

## 3.2 Analýza existujících řešení

Programů, které se zabývají problematikou elektroniky existuje velké množství a je možné se s nimi setkat v širokém spektru požadované odbornosti. V oblasti tvorby a analýzy elektronických obvodů spadá většina programů do kategorie určené pro odborné uživatele. Při návrhu aplikace se tedy očekává, že cílový uživatel má dobrou znalost fungování elektronických komponent a obvodů z nich sestavených. Skupina aplikací, které jsou cíleny pro uživatele laiky je značně omezena. Přesto se jedny z nejpopulárnějších aplikací, zabývající se touto tematikou, řadí právě do kategorie určené pro méně zkušené uživatele.

### 3.2.1 idealCircuit




Obrázek 1: Snímek z aplikace idealCircuit

Aplikace **idealCircuit** je nativní, analogový simulátor elektronických obvodů založený na ideálních součástkách. Princip je podobný jako u programu **NL5 Circuit Simulator**, ovšem aplikace je výrazně redukována tak, aby obsahovala jen nezbytné komponenty, modely a funkce. Navíc zde bylo, oproti aplikaci **NL5 Circuit Simulator**, upraveno i uživatelské rozhraní tak, aby bylo jednoduché a intuitivní. [1]

Program umožňuje sestavování elektronických obvodů z dostatečně rozsáhlé knihovny komponent. Dostupné jsou základní součástky, jako například rezistory,

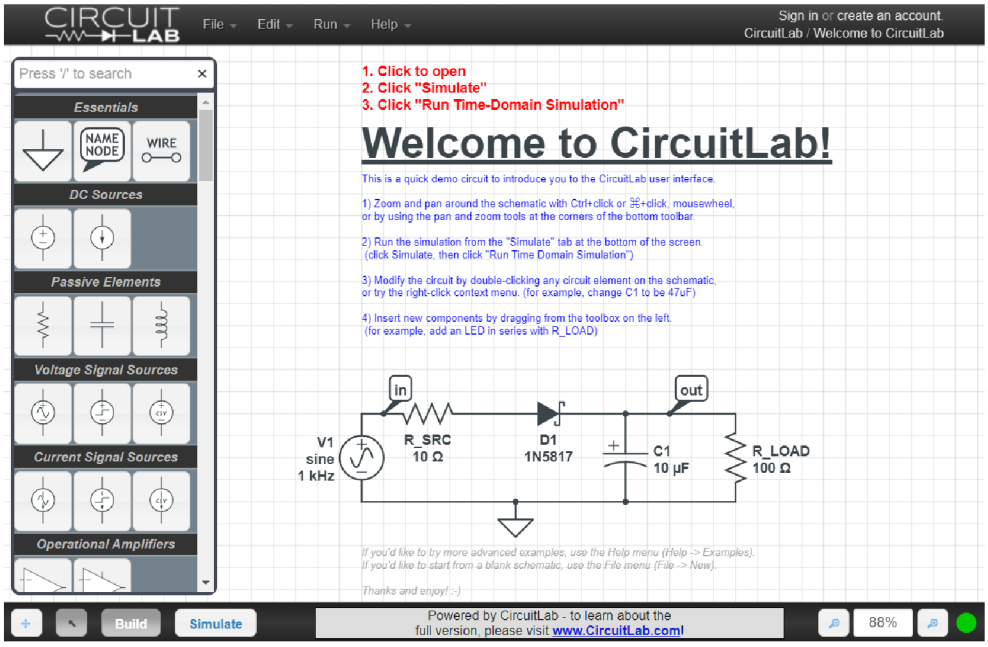
cívky (induktory), kondenzátory nebo tranzistory, ale také komponenty jako zesilovače, transformátory nebo logické brány. Použité modely jsou redukovány pouze na nezbytné, a to samé platí také pro nastavitelné parametry součástek.

Grafický editor je intuitivní a práce s ním je, po osvojení si základních principů, snadná. Drobnou nevýhodou editoru je absence všech klávesových zkratek.

Možnosti simulace jsou v programu omezeny na možnosti simulace přechodových jevů a možnost „AC sweep“ (lineární analýza malých signálů). Výsledky simulace jsou zobrazovány v grafu s nastavitelným rozsahem, který je přehledný a dobře čitelný. Průběh simulace je velmi rychlý a je možné ho i kontinuálně sledovat (na rozdíl od vygenerování výsledků v konkrétním časovém intervalu).

Aplikace nabízí také knihovnu ukázkových zapojení a dobře zpracovaný uživatelský manuál s popisem nastavitelných vlastností všech komponent.

### 3.2.2 CircuitLab




Obrázek 2: Snímek z aplikace Circuit Lab

CircuitLab [2] je webová aplikace, vytvořená především pro výukové účely. Jedná se o simulátor ideálních obvodů, stejně jako u aplikace idealCircuit. Knihovna komponent je zde přítomna v podobném rozsahu, ovšem doplněna o některé integrované obvody, jako je například časovač 555. Uživatelské rozhraní je pěkné a intuitivní. Grafický editor je možné ovládat pomocí klávesových zkratek.

Simulace je dostupná prakticky ve stejném rozsahu jako u aplikace idelaCircuit ovšem výsledky jsou trochu méně čitelné. Průběh simulace je oproti nativnímu idealCircuit delší, což je pravděpodobně způsobeno webovou podstatou aplikace.

Naopak značně propracovanější je aplikace v oblasti dostupných výukových materiálů. CircuitLab poskytuje podstatně rozsáhlejší knihovnu ukázkových zapojení, která jsou navíc i dobře zdokumentována. Na oficiálních stránkách je také k dispozici velké množství výukových textů a videí.

### 3.2.3 Simulátory využívající SPICE

Oba výše zmíněné programy mají své vlastní simulační jádro. Velké množství dostupných simulačních programů ovšem využívá jako simulační jádro program SPICE. Jedná se o program, který se do maximální možné míry snaží simulovat chování reálných komponent (na rozdíl od dvou výše zmíněných, které simulují komponenty ideální). Programy založené na SPICE jsou využívány především profesionály. Ti je používají především kvůli prakticky neomezené možnosti rozšíření o další komponenty a jejich modely. Výrobci elektronických součástek navíc často poskytují ke konkrétním komponentám modely ve SPICE kompatibilním formátu.

I přes svou komplexitu je ovšem program schopný simulovat i ideální komponenty a bylo by možné ho využít jako simulační jádro i pro velmi jednoduché ideální obvody.

## 3.3 Metodika práce s grafickými simulátory elektronických obvodů

Metodika práce s grafickými simulačními programy elektronických obvodů by mohla být rozdělena do několika, téměř nezávislých, kategorií. Jsou to:

- metodika práce se soubory a dokumenty,
- metodika tvorby a editace obvodů,
- metodika nastavení parametrů součástek,
- metodika simulace a interpretace výsledných hodnot.

### 3.3.1 Metodika práce se soubory a dokumenty

V této kategorii je metodika prakticky totožná s metodikou práce se soubory a dokumenty u grafických editorů (především z kategorie editorů vektorové grafiky). Grafické programy běžně umožňují vytvářet dokumenty a ukládat je jako soubory. Při načtení souboru by měl být obsah dokumentu shodný se stavem při uložení.

Soubory jsou nejčastěji ukládány v textové formě některého ze standardních datových formátů, jako je XML nebo JSON. Formát informace se poté různí program od programu, ale prakticky se vždy jedná o formu seznamu objektů a jejich vlastností. Soubor může být dále doplněn informacemi o datu vytvoření nebo verzi souboru.

Základními operacemi jsou poté:

- otevření dokumentu,
- uzavření dokumentu,
- uložení do souboru,
- uložení do souboru pod novým názvem,
- načtení dokumentu ze souboru.

Pokud program umožňuje práci s více dokumenty současně, bývá standardem možnost kopírování objektů napříč jednotlivými dokumenty.

### 3.3.2 Metodika tvorby a editace obvodů

Tvorba a editace elektronických obvodů se provádí u všech zkoumaných programů pomocí pracovní plochy grafického editoru. Základním úkonem je vložení komponenty do pracovní plochy. To je nejčastěji uskutečněno přetažením komponenty z knihovny na požadované místo.

Jakmile jsou součástky na místě, je možné mezi nimi vytvářet spojnice. Každý komponent má vyznačena aktivní místa, která odpovídají vstupům nebo výstupům jednotlivých schématických značek. Vytvoření spojnice mezi komponenty je pak provedeno tahem daného aktivního bodu do aktivního bodu jiného komponentu (nebo jen do prostoru pracovní plochy).

Editační funkce poté zahrnují přesouvání jednotlivých komponent a spojnic, rotaci a mazání komponent nebo kopírování a vkládání částí obvodů. Vlastností všech zkoumaných editorů byla dále možnost vrátit editaci zpět, anebo odstraněnou změnu znovu provést.

### 3.3.3 Metodika nastavení parametrů součástek

Parametry elektronických součástek jsou ve všech zkoumaných řešeních nastavovány pomocí samostatného dialogového okna. Samotnou hodnotu pak uživatel nastavuje vpisem do textového pole. Před nastavením se u hodnoty validuje platný datový typ a platný rozsah.

### 3.3.4 Metodika simulace a interpretace výsledných hodnot

Před provedením simulace uživatel musí zvolit zkoumané uzly nebo komponenty. V případě idealCircuit je to provedeno při návrhu obvodu, přidáním ampérmetru nebo voltmetru přímo do návrhu. CircuitLab problematiku řeší přidáním „sondy“ na dané uzly nebo komponenty. Aplikace CircuitLab také nabízí možnost generování výsledků pro všechna možná řešení.

Ke spuštění simulace, je v obou zkoumaných řešeních dostupný samostatný panel/dialog, který současně interpretuje její výsledky. Před spuštěním uživatel volí parametry simulace, které jsou různé podle jejího typu.

U simulace přechodových jevů je to vždy ekvivalent sledovaného časového intervalu a délky kroku. U analýzy malých signálů je to horní a spodní rozsah frekvence a počet generovaných bodů.

Po spuštění a dokončení simulace jsou výsledky dostupné ve formě spojnicového grafu. U aplikace idealCirciut jsou napětí a proud zobrazeny v jednom grafu. Aplikace CircuitLab zobrazuje obě sledované veličiny, každou ve vlastním grafu. Obě řešení nabízí možnost volby toho, které výsledky jsou v grafu zobrazeny.

## 3.4 Návrh funkcionalit aplikace na základě zkoumaných řešení

Na základě zkoumaných řešení byl vytvořen návrh funkcionalit, které by měla aplikace implementovat. Ty byly, pro přehlednost, rozděleny do kategorií, odpovídajícím zkoumaným metodikám práce s referenčními aplikacemi.

- Práce se soubory a dokumenty:
  - vytvoření nového dokumentu,
  - uzavření otevřeného dokumentu,
  - uložení dokumentu do souboru,
  - uložení dokumentu do souboru pod novým názvem,
  - načtení dokumentu ze souboru,
  - načtení dokumentu ukázkového zapojení,
  - validace správnosti otevřených dokumentů.
- Tvorba a editace obvodů:
  - vložení komponenty do obvodu,
  - posun komponent po pracovní ploše,
  - změna rotace komponenty,
  - vytvoření spojnice mezi komponenty,
  - kopírování a vkládání částí obvodu (i napříč dokumenty),
  - vrácení změny a znovu provedení vrácené změny (Undo/Redo),
  - mazání komponent,
  - možnost nastavení vlastností komponenty v samostatném dialogu.
- Simulace a interpretace výsledků:
  - simulace přechodových jevů,
  - nastavení parametrů simulace v rozsahu: čas počátku a konce simulace, délka kroku,

  - možnost zobrazit, kde se sledovaný uzel/komponent nachází v obvodu,
  - zobrazení výsledků v podobě spojnicového grafu,
  - možnost volby zobrazených výsledků.

- Informace o komponentách:
  - možnost zobrazení informací o jednotlivých komponentách.

# 4 Návrh aplikace

## 4.1 Určení uživatele aplikace

Aplikace je dle analýzy a návrhu určena pro uživatele, kteří mají minimální znalosti elektronických obvodů a komponent. Uživatel by měl být v aplikaci schopen sestavit návrh elektronického obvodu a tento návrh následně simulovat. Výstupem aplikace mohou být soubory reprezentující vytvořené dokumenty. V aplikaci budou také uživatelsky dostupné informace o jednotlivých elektronických komponentách.

## 4.2 Případy použití

Případy použití byly rozděleny na základě výše uvedených navrhovaných funkcionalit do kategorií:

- tvorba a editace elektronických obvodů,
- práce se soubory a dokumenty,
- simulace a interpretace jejich výsledků,
- zobrazení dostupných informací o komponentách.

### 4.2.1 Tvorba a editace elektronických obvodů

Za tvorbu a editaci elektronických obvodů by měla být zodpovědná třída kreslící plochy. Ta umožní jeho sestavení, posun v prostoru, přiblížení nebo oddálení a funkce spojené s editací. Třída by také měla podporovat možnosti používání klávesových zkratek pro úkony, pro které je to možné (například rotace, kopírování a mazání objektů).

Uživateli by měly být dostupné následující funkce:

**Vložení nového objektu** – vložení objektu do pracovní plochy bude provedeno přetažením ikony příslušné komponenty do kreslící plochy. Při tažení by mělo být indikováno, na jaké místo bude objekt, po uvolnění myši, vložen.

**Označení objektu nebo skupiny objektů** – mělo by být uživateli umožněno označit libovolnou skupinu objektů a následně s nimi provádět manipulace, kopírovat je nebo mazat.

**Mazaní objektů** – objekty budou odstraněny z kreslící plochy a nebude možné s nimi dále interagovat.

**Manipulace vlastností komponent** – uživateli bude umožněno měnit grafické vlastnosti objektu, jako je rotace nebo poloha v rámci pracovní plochy (ve smyslu absolutní poloha vůči počátku). Uživatel také bude schopen nastavit konkrétní vlastnosti dané komponenty.

**Kopírování a vkládání objektů** – v aplikaci bude umožněno kopírovat a vkládat označené objekty kreslící plochy. Vložené objekty by měly mít stejnou relativní pozici vůči sobě a měly by mít stejné grafické i parametrické vlastnosti (stejně nastavené parametry součástek). Kopírování by mělo být dostupné také mezi jednotlivými dokumenty.

**Vrácení a znovu provedení změn** – změny provedené na obvodu by mělo být možné vrátit zpět. Vrácené změny bude možné opět vyvolat a to tak, aby se stav před provedením sekvence undo->redo vždy shodoval se stavem po provedení. Pokud bude vykonána skupina změn současně (například smazání několika objektů), musí být tato editace brána jako jeden úkon v zásobníku undo/redo.

**Vytvoření spojnice mezi objekty** – vytvoření spojnice je ekvivalentní s vložením objektu linky. Vložení bude provedeno přímo z pracovní plochy pomocí aktivních bodů daných komponent. Vložený objekt linky bude možné manipulovat stejně, jako objekty komponent (s výjimkou rotace).

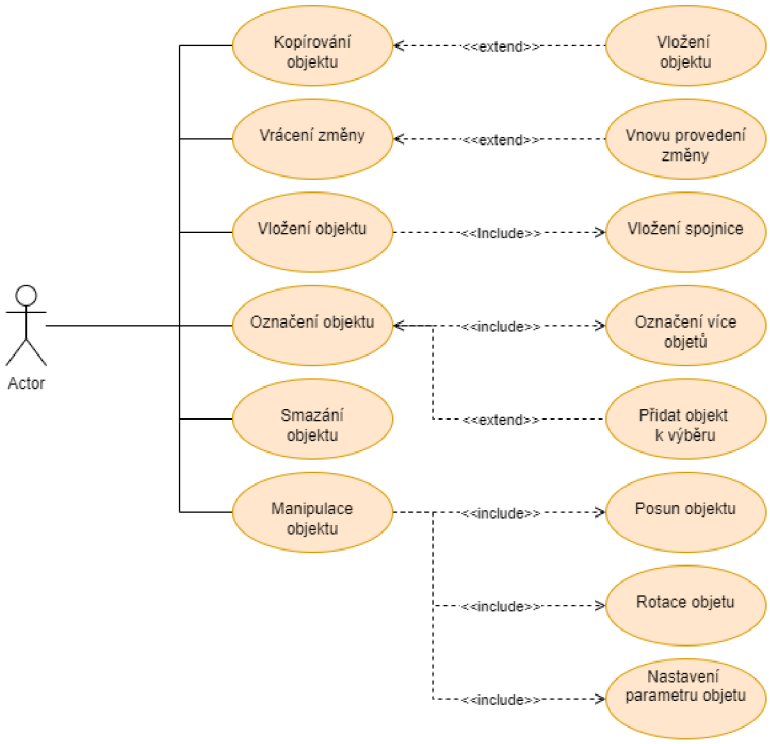
Práci s obvody dále popisuje diagram z obrázku 3.

### 4.2.2 Práce se soubory a dokumenty

V oblasti práce se soubory a dokumenty by měly být pro uživatele dostupné tyto funkcionality:

**Vytvoření nového dokumentu** – aplikace by měla umožnit tvorbu nových dokumentů. Vytvořený dokument by neměl obsahovat žádné komponenty ani žádná zapojení. Při spuštění programu by měl být pro uživatele automaticky vytvořen nový dokument.

**Uzavření otevřeného dokumentu** – otevřené dokumenty bude možné libovolně ukončit. Pokud budou v dokumentu neuložené změny, měla by aplikace vyžadovat potvrzení uzavření dokumentu.




Obrázek 3: Diagram práce s obvody

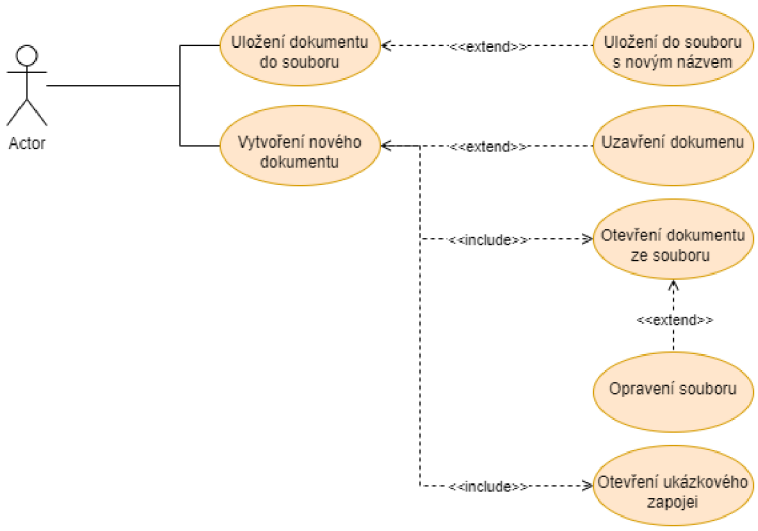
**Uložení dokumentu do souboru** – vytvořené dokumenty bude možné ukládat do souborů na disk počítače. Uživateli bude umožněno zvolit název a cestu souboru. Pokud by měl nově vytvořený soubor přepsat některý ze stávajících, bude vyžadováno potvrzení.

**Uložení dokumentu do souboru pod novým názvem** – uživatel bude moci ukládat dokument pod novým názvem. Pokud by měl nově vytvořený soubor přepsat některý ze stávajících, bude vyžadováno potvrzení.

**Načtení dokumentu ze souboru** – uživateli bude umožněno otevírat dokumenty uložené do souborů. Pokud se uživatel pokusí otevřít neplatný dokument, bude o tom informován. Pokud se uživatel pokusí otevřít platný, ale poškozený dokument, bude vyvolán dialog opravy chybného souboru.

**Načtení dokumentu ukázkového zapojení** – uživateli bude umožněno otevřít dokument ze souboru ukázkového zapojení. S takovýmto dokumentem bude možné běžně manipulovat, ale nebude možné soubor přepsat.

Práci se soubory a dokumenty popisuje diagram z obrázku 4.




Obrázek 4: Diagram práce s dokumenty

### 4.2.3 Simulace a interpretace výsledků

Simulace a interpretace jejich výsledků bude probíhat v dedikovaném panelu. Uživateli by měla být umožněna přehledná práce s výsledky simulace. Panel bude implementovat tyto funkcionality:

**Spuštění simulace pro zvolený dokument** – během simulace by měl program zůstat responzivní a měl by uživatele informovat o tom, že simulace probíhá.

**Nastavení parametru simulace** – uživateli bude dostupná možnost simulace přechodových jevů, u které bude možné dále nastavovat parametry: čas počátku, čas konce, délka kroku a maximální délka kroku (u automaticky generované délky kroku).

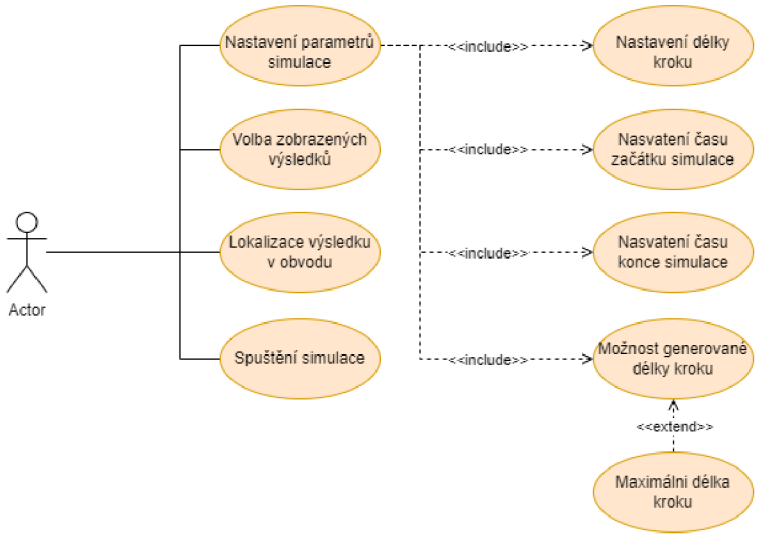
**Zobrazení sledovaného výsledky v obvodu** – uživateli bude umožněno lokalizovat sledovaný výsledek na konkrétním uzlu/komponentu v obvodu.

**Možnost volby zobrazených výsledků** – bude možné zvolit, které výsledky jsou zobrazeny ve spojnicovém grafu.

Práci s panelem simulace popisuje diagram z obrázku 5.

### 4.2.4 Informace o komponentách

Informace o komponentách budou zobrazeny v samostatném panelu. Panel by mělo být možné vyvolat z knihovny komponent přímo pro konkrétní součástku. Panel by měl implementovat tyto funkcionality:

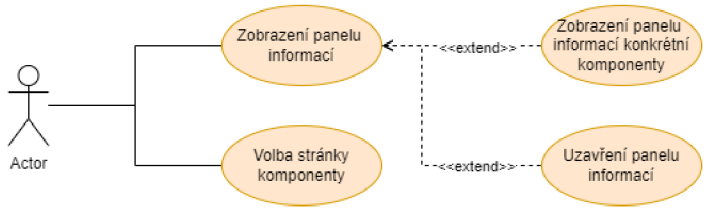



Obrázek 5: Diagram práce s panelem simulace

**Zobrazený panel informací** panel by mělo být možné zobrazit samostatně anebo již pro konkrétní komponentu.

**Volba ze seznamu komponent** v panelu by měl být dostupný seznam komponent, pro které jsou dostupné informace. Volbou ze seznamu se zobrazí požadované informace.

Práci s panelem komponentů popisuje diagram z obrázku 6.




Obrázek 6: Diagram práce s panelem informací o komponentách

# 5 Implementace aplikace

## 5.1 Použité technologie

Aplikace byla vyvíjena v objektově orientovaném jazyku Java 17. Pro tvorbu grafického uživatelského rozhraní byla zvolena knihovna JavaFX.

JavaFx je postavena na návrhu Model-View-Presenter a umožňuje dobré rozdělení uživatelského rozhraní na jednotlivé logické prvky. Rozložení komponent je

možné definovat ve formátu FXML. Vzhled komponent ve formátu CSS a logiku jednotlivých panelů v samostatné třídě ovladače. Uživatelské rozhraní je tedy, z hlediska struktury kódu, velmi dobře čitelné a je možné jej snadno implementovat a upravovat.

Jako build systém byl zvolen Maven a jako testovací nástroj byla zvolena knihovna JUnit.

## 5.2 Architektura aplikace

Návrh aplikace je postaven na návrhovém vzoru Model-View-Presenter s využitím agregátoru událostí (návrhový vzor Event Aggregator). Jednotlivé panely pak pomocí agregátoru komunikují vzájemně i s hlavním ovladačem (controller), který poté interpretuje události jako konkrétní příkazy na model.

### 5.2.1 (MVP) Model View Presenter

Návrhový vzor MVP je obecně definován tak, že každá grafická komponenta je rozložena na dvě vzájemně komunikující části, zobrazení (view) a moderátora (presenter). Zobrazení pak obsahuje jen minimum logiky, která je spojena čistě se zobrazením dané komponenty. Veškeré uživatelsky vyvolané události jsou pak předávány moderátorovi, který je zpracuje.

Pokud tedy například dojde k vyplnění textu do textového pole, je moderátor zobrazením informován, že došlo ke změně. Nyní může moderátor například provést validaci vložené hodnoty nebo změnit vzhled textového pole tak, aby indikoval změněnou hodnotu.[3]

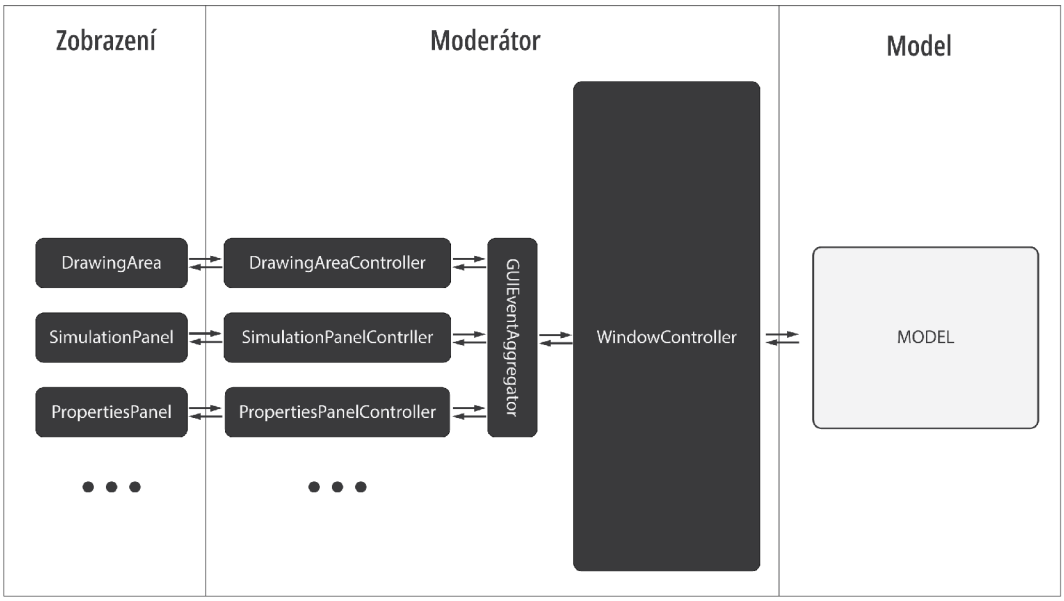
### 5.2.2 Agregátor událostí (Event Aggregator)

Agregátor událostí je struktura, která funguje jako prostředník pro sdílení eventů mezi jednotlivými objekty. Jednotlivé objekty mohou přes agregátor událostí publikovat nebo odebírat konkrétní eventy. Pokud některý z objektů publikuje událost, agregátor notifikuje všechny odběratele, kteří na událost déle reagují.

Využití agregátoru tak redukuje závislost komponent a umožňuje snadnější sledování událostí v aplikaci. Nevýhodou je, že není možné zcela kontrolovat odkud jsou události publikovány. V některých případech je tak nutné kontrolovat jejich platnost.[4]

### 5.2.3 Architektura grafického rozhraní

Grafické uživatelské rozhraní aplikace je rozděleno do jednotlivých panelů. Každý panel má poté svého vlastního operátora, který řídí chování prezentační logiky, zpracovává události a komunikuje je pomocí agregátoru událostí. Stručný diagram architektury poté nalezneme ne obrázku 7 (množství panelů bylo pro přehlednost redukováno):




Obrázek 7: Architektura grafického rozhraní

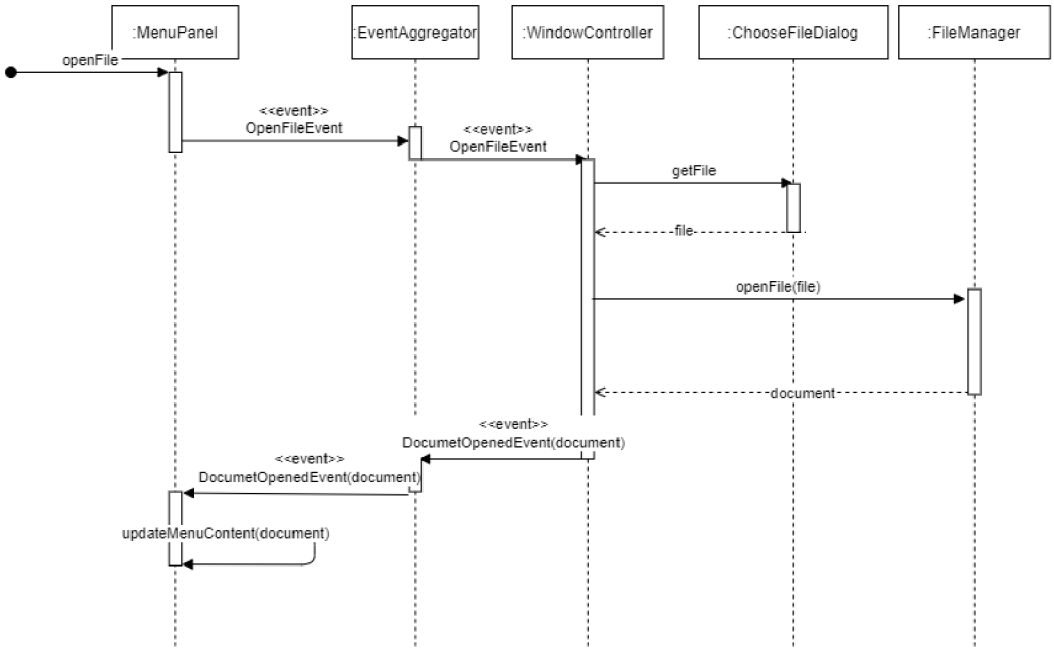
Každý panel má obvykle svoji vlastní sadu konkrétních událostí, které mohou obsahovat dodatečné informace. Řídící může například vyvolávat samostatnou událost **DOKUMENT_BYL_UZAVŘEN**, která nese i odkaz na zasažený dokument. Odběratelé události pak mohou na událost reagovat, aniž by si museli vyžádat dodatečné informace.

Řídícím moderátorem pro všechny panely je WindowController, který mění příslušné události na příkazy provedené na modelu. Současně komunikuje dění modelu zpět do zbytku zobrazení.

Konkrétním případem může být například nahrání dokumentu ze souboru (sekvenční diagram na obrázku č. 8). Uživatel v menu aplikace zvolí možnost „otevřít“. Moderátor panelu menu poté publikuje přes agregátor odpovídající událost. Agregátor ji předá třídě WindowContoroller, která vyvolá dialog výběru souboru. Jakmile uživatel vybere zvolený soubor, WindowController zvolený soubor obdrží a pošle žádost do třídy FileManger na jeho otevření (zde byl diagram mírně zjednodušen kvůli čitelnosti). Pokud je soubor platný, FileManager vytvoří odpovídající dokument a vrátí ho WindowControlleru. Ten poté publikuje odpovídající událost přes agregátor. Moderátor panelu menu poté upraví vzhled panelu tak, aby odpovídal novému dokumentu. Výše popsaný postup je znázorněn na diagramu 8.

### 5.2.4 Třída kreslící plochy (DrawingCanvas)

Za zvláštní zmínku také stojí třída kreslící plochy, která je zodpovědná za zobrazení a manipulaci obvodů. Reprezentaci komponent na kreslící ploše zajišťuje třída CanvasObject. Ta udržuje informace o lokaci v kreslící ploše, relativní pozici




Obrázek 8: Sekvenční diagram otevření dokumentu

vůči středu a způsobu, jakým se komponenta na kreslící ploše zobrazí.

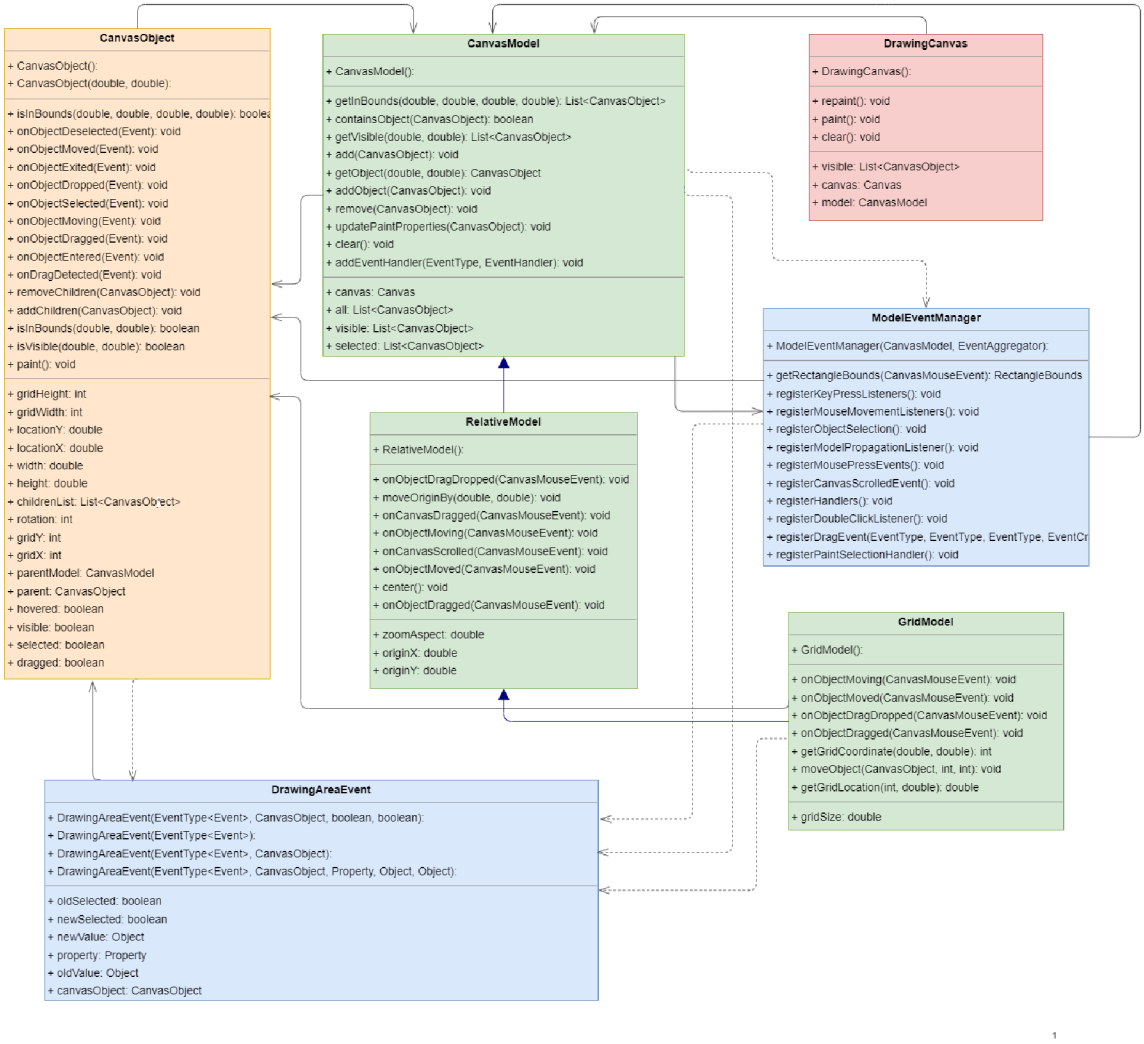
Objekty kreslící plochy jsou spravovány třídou CanvasModel, která udržuje seznam vložených objektů a zajišťuje jejich manipulace. Mezi manipulace patří změna pozice objektu, mazání nebo změna rotace. Třída je také schopna vrátit seznam označených objektů nebo seznam objektů, které se nacházejí v určitém rozsahu souřadnic.

Třída RelativeModel doplňuje třídu CanvasModel o možnosti posunu kreslící plochy a o možnost přiblížení/oddálení. Třída GridModel ji pak dále rozšiřuje o možnost umístění objektu do mřížky.

Vnější komunikace kreslící plochy je poté zajištěna třídou DrawingAreaEvent. Ta je rozšířením třídy Event standardní knihovny a nese informace o změně komponent, parametrů atd. Třídu kreslící plochy dále popisuje diagram na obrázku 9.

### 5.2.5 Architektura modelu aplikace

Struktura modelu je odvozena od datové základny aplikace. Ta musela být zvolena tak, aby bylo možné volně objekty manipulovat v kreslící ploše, ale současně aplikovat pouze některé provedené změny. Bylo tedy nutné vyřešit otázku, jak budou data reprezentována a jak s nimi bude manipulováno. Řešením bylo rozdělení reprezentace dokumentu na dvě části. Surovou netypovanou reprezentaci dat a typovaný dokument, který již obsahuje konkrétní typy objektů.




Obrázek 9: Diagram tříd pro kreslící plochu

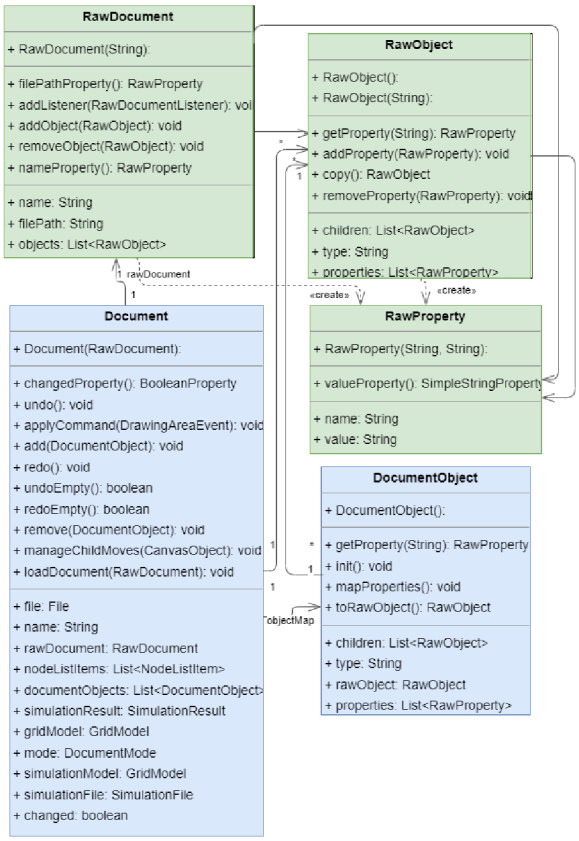
**5.2.5.1 Surový dokument** – netypovaný RawDocument je základem datové struktury aplikace. Jedná se o datovou strukturu, která obsahuje jen minimum logiky a je podobná reprezentaci dat například ve formátu XML nebo JSON. Dokument samotný pak udržuje pouze seznam komponent (RawComponent) a jejich jmenovitých vlastností. Pokud dojde k jakékoliv změně ve stavu surového dokumentu, je tato změna předána, za využití návrhového vzoru „pozorovatel“, všem odběratelům. Hlavním odběratelem těchto událostí je pak již typovaný dokument.

Surový dokument je také základní strukturou pro vytváření a načítání souborů.

**5.2.5.2 Typovaný dokument** – je vždy nástavbou konkrétního surového dokumentu. Obsahuje seznam objektů, které jsou reprezentací jednotlivých Ra-

wObject ze surového dokumentu. Dokument samotný pak reaguje nazměny v surovém dokumentu a automaticky upravuje data tak, aby změnám odpovídaly.

Typovaný dokument také obsahuje dodatečné informace o simulaci, výsledcích simulace, souboru a další uživatelská data, která nejsou persistentní. Datovou strukturu dále popisuje diagram na obrázku 10.




Obrázek 10: Diagram tříd datové struktury

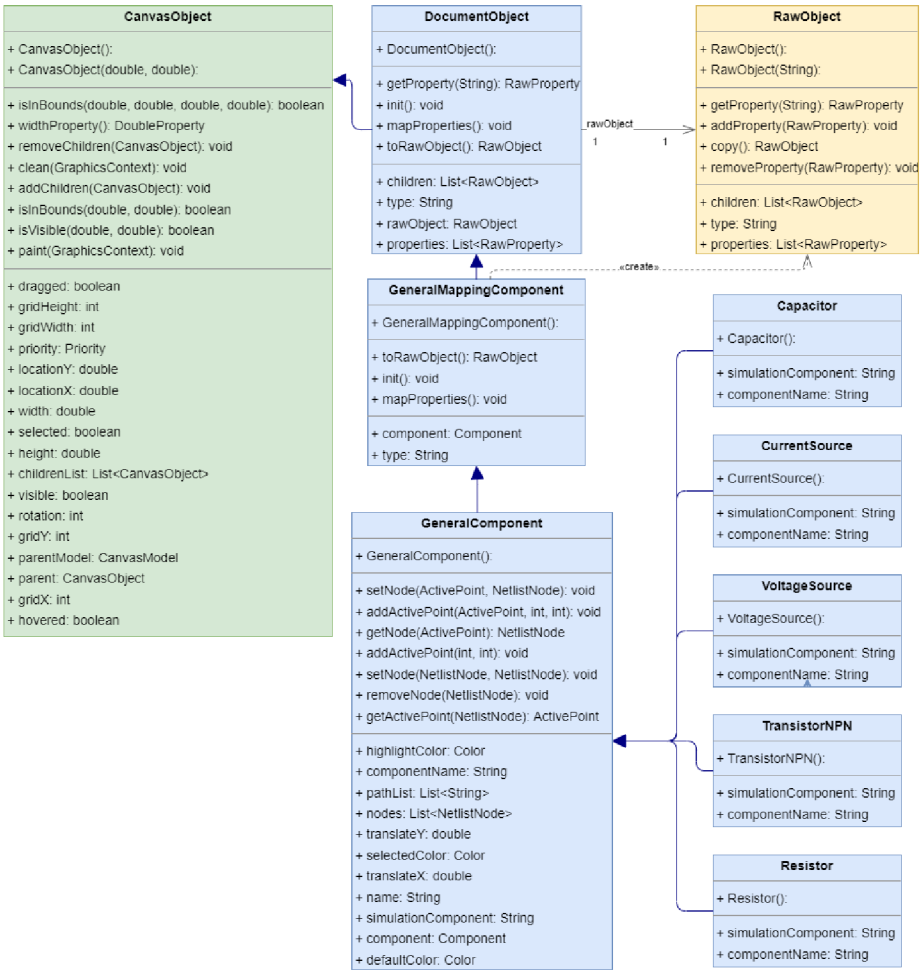
### 5.2.6 Propojení tříd DrawingCanvas a Document

Dokument je na třídu kreslící plochy napojen pomocí objektu DocumentObject. Ten rozšiřuje třídu CanvasObject a implementuje všechny požadované funkce. Každý dokument pak obsahuje GridModel a synchronizuje jeho obsah se surovým dokumentem. Při změně aktivního dokumentu, příslušný ovladač (v tomto případě DrawingAreaController) jen nastaví model kreslící plochy na model z dokumentu.

Třída DocumentObject je dále rozšířena třídou GeneralMappingObject, která zajišťuje snadné mapování vlastností DocumentObject a RawObject. Mapování je provedeno pomocí anotací, aby byl redukován zbytečně se opakující kód.

Konkrétní komponenty, jako například třída Capacitor, rozšiřují třídu GeneralComponent. Ta obsahuje informace potřebné ke snadné tvorbě jednotlivých

komponent a také informace potřebné k sestavení sítě komponent pro provedení simulace. Závislosti jednotlivých objektů jsou dále popsány na diagramu z obrázku 11.



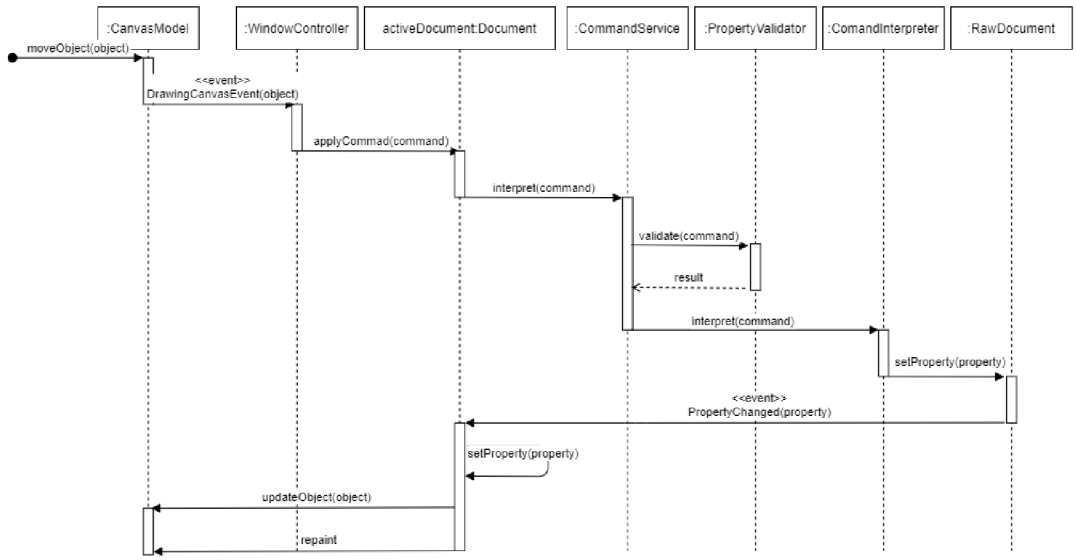

Obrázek 11: Diagram tříd pro závislosti objetků dokumentu

### 5.2.7 Propagace změn do surového dokumentu

Pokud dojde ke změnám vlastností objektu, které jsou mapované, je nutné je přepsat do surového dokumentu, aby se projevily. Tyto změny jsou vyvolávány především manipulací objektu v kreslící ploše nebo nastavením vlastnosti komponentu v panelu vlastností.

O propsání změny do dokumentu rozhoduje WindowController. Pokud shledá, že má být změna propsána, předá příkaz konkrétnímu dokumentu. Ten ji předá třídě CommandService. Ta je zodpovědná za validaci hodnot a interpretaci konkrétních příkazů. Pokud je změna validní, je propsána do surového dokumentu, který informuje všechny odběratele.

Konkrétní příklad změny popisuje sekveční diagram na obrázku 12 (diagram byl redukován o komunikaci v rámci GUI, ta byla popsána výše a její princip je zde stejný).




Obrázek 12: Sekvenční diagram změny surového dokumentu

### 5.2.8 Zbylá architektura modelu

V modelu se nacházejí ještě další pomocné třídy, zodpovědné za požadované funkcionality:

**FileService** – třída zodpovědná za ukládání a načítání souborů. Při načítání validuje hodnoty a je schopna opravit některé poškozené soubory.

**CommadService** – jednotný uzel, přes který procházejí veškeré změny dokumentu. Je zodpovědný za jejich validaci a provedení. Poskytuje funkcionalitu undo/redo.

**CommadInterpreter** – skutečně provádí změny na surovém dokumentu. Poskytuje také možnost provést reverzní změnu, která je využívána funkcí zpět.

**DocumentManager** – je zodpovědný za správu dokumentů.

### 5.2.9 Struktura souboru

Soubory jsou textového charakteru a jsou založeny na formátu JSON. Informace souboru je 1:1 shodná s daty surového dokumentu. Soubor tedy obsahuje informace o názvu dokumentu a seznam objektů. Každý objekt pak obsahuje seznam dceřiných objektů a svých vlastností.

Struktura dokumentu se pak řídí těmito pravidly:

- dokument musí vždy obsahovat pole „objects“, které reprezentuje seznam objektů dokumentu,
- dokument musí vždy obsahovat položku „name“,
- každý objekt dokumentu musí obsahovat pole dceřiných objektů „children“,
- každý objekt musí obsahovat pole vlastností „properties“,
- v poli vlastností objektu je vždy přítomna právě jedna položka „TYPE“.

Soubor pro dokument s názvem „resistor“, který obsahuje pouze objekt rezistoru, by pak struktura souboru vypadala takto:

```
{
    "objects": [
        {
            "children": [],
            "properties": [
                {"name": "gridHeight", "value": "2"},
                {"name": "gridX", "value": "-5"},
                {"name": "gridY", "value": "-9"},
                {"name": "rotation", "value": "0"},
                {"name": "gridWidth", "value": "2"},
                {"name": "TYPE", "value": "RESISTOR"},
                {"name": "resistance", "value": "250"}
            ]
        }
    ],
    "name": "resistor"
}
```

### 5.2.10 Kontrola souboru a načtení dokumentu

Za tvorbu dokumentu ze souborů a jejich validaci je zodpovědná třída FileService. Ta při načítání souboru kontroluje platnost výše zmíněných pravidel a validuje obsah vlastností přítomných objektů.

Validace vlastností probíhá prostřednictvím pomocné třídy DocumentObjectFactory. Tovární třída se pokusí vytvořit objekt daného typu a nastavit všechny vlastnosti podle seznamu ze souboru. Pokud daná vlastnost neexistuje nebo je jiného datového typu, je vrácena chybová hláška. Pokud je vše v pořádku, je vrácen konkrétní DocumentObject.

Oprava chybného objektu je provedena doplněním chybějících parametrů s výchozími hodnotami, případně nastavením chybných parametrů do výchozích hodnot.

# 6 Simulace

Počítačová analýza elektronických obvodů je často využívána v případech, kdy je potřeba analyzovat chování navrženého obvodu, aniž by bylo nutné ho fyzicky sestavit. Simulace umožňuje snadnou změnu parametrů obvodu i v takových případech, kde by to u fyzického zapojení nebylo možné.

Pokud má být simulace používána jako platný nástroj, musí být vykonána v rozumném čase. Klíč k efektivnímu provedení simulace je volba vhodné úrovně abstrakce pro daný problém. Pro podporu dané úrovně abstrakce modelu musí simulátor poskytovat vhodný algoritmus.

Historicky se u simulátorů setkáváme s možnostmi analogové nebo digitální simulace. Programy, které podporují oba algoritmy simulace se označují jako „mixed-mode“.[5]

## 6.1 Analogová simulace

Analogová simulace se zaměřuje na lineární a nelineární chování obvodu v průběhu času nebo v určitém rozmezí frekvencí. Odpověď okruhu je získána iterativním výpočtem Kirchhoffových zákonů pro daný obvod v krocích, které jsou zvoleny tak, aby se výsledek blížil stabilní hodnotě, a aby aproximace integrace byli přiměřeně přesné. Protože Kirchhoffovy zákony formulují soustavu rovnic, simulátory většinou operují na principu řešení matice rovnic pro každý určený bod v čase. Tento přístup většinou vede k delším časům simulace v porovnání s digitálními simulátory.[5]

## 6.2 Digitální simulátory

Digitální simulátory se liší od analogových v několika ohledech. Například, že nevyžadují řešení Kirchhoffových zákonů. Místo toho simulátor pouze rozhoduje, zda došlo ke změně logického stavu uzlu a propaguje tuto změnu připojeným elementům v podobě události.

Pokud dojde k události, simulátor prověří pouze ty elementy, které jsou událostí zasaženy. Díky tomuto přístupu není nutné v digitální simulaci řešit matice rovnic. Analogové simulátory oproti tomu musí iterativně počítat chování celého okruhu kvůli přímým a zpětným přenosovým vlastnostem analogových součástek. Výsledkem je, že digitální simulátory mají výraznou výpočetní výhodu, která se promítá do výrazně vyšší rychlosti digitální simulace.[5]

## 6.3 Simulace smíšených signálů

Moderní obvody často obsahují směs analogových a digitálních dílčích obvodů. Pokud mají být takové obvody simulovány efektivně a přesně, je potřeba použít spojení analogových a digitálních technik. Pokud je spojena analogová a digitální simulace, výsledek se označuje jako „mixed-mode“ simulace.

Jsou používány dvě základní metody pro implementaci „mixed-mode“ simulace, a to „native-mode“(nativní režim) a „glued-mode“(lepený režim). Simulátory používající nativní režim implementují digitální i analogové algoritmy v jednom spustitelném souboru. Simulátory s lepeným režimem mají dva samostatné spustitelné soubory pro analogovou a digitální simulaci. Tyto typy simulátorů musejí definovat input/output protokol tak, aby spolu mohli spustitelné soubory efektivně komunikovat. Omezení komunikace často limitují rychlost a v některých případech i přesnost simulace. Na druhou stranu tento typ usnadňuje vývoj jednotlivých modelů komponent.[5]

## 6.4 NGSpice

Aplikace využívá jako simulační jádro program NGSpice. Je to open source simulátor založený na programu SPICE. Umožňuje využití standardních SPICE knihoven a modelů. Simulaci okruhů provádí v nativním „mixed-mode“ režimu a je díky tomu poměrně rychlý a přesný. Pro aplikaci byl zvolen především kvůli tomu, že poskytuje možnost pohodlného ovládání z příkazové řádky a přesměrování výsledků simulace do samostatných souborů.[5]

NGSPice je v aplikaci využit pomocí nativní Java funkce Runtime.exec, která umožňuje asynchronní spuštění nativních programů, jejich ovládání pomocí příkazů a současně možnost detekce ukončení procesu. To vše zajišťuje snadné spuštění programu s potřebnými parametry a vyčkání výsledků, které jsou následně zpracovány.

Simulace probíhá v následujících krocích:

- vytvoření net-listu komponent,
- vytvoření simulačního souboru,
- vytvoření vstupního souboru pro NGSpice,
- spuštění programu NGSpice pomocí Runtime.exec a vyčkání ukončení,
- načtení výsledků simulace a jejich nastavení do panelu simulace.

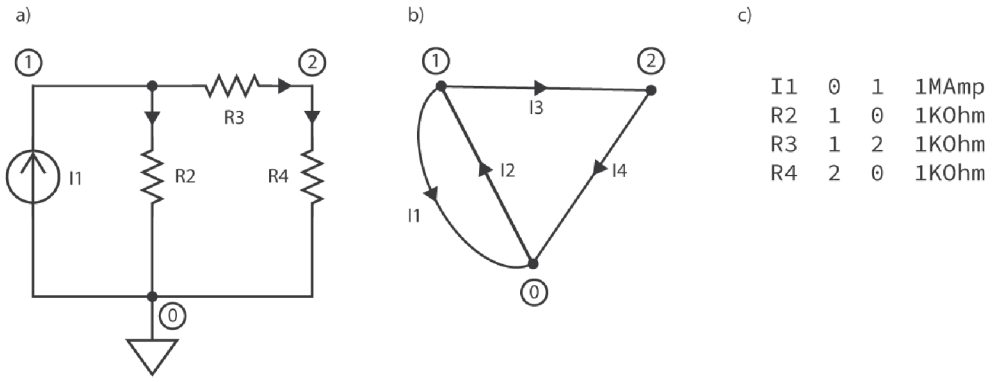
## 6.5 Topologie obvodu a net-list

Abychom mohli popsat princip sestavení net-listu, musíme si nejprve určit pojmy net-list a topologie obvodu.

Elektrické obvody jsou složeny z jednotlivých větví, které jsou spojovány pomocí terminálů. Každá větev má dva terminály – vstupní uzel a výstupní uzel. Proud větví teče od vstupního uzlu k výstupnímu. Každý uzel obvodu spojuje několik větví. To, ke který uzlům je každá větev připojena, pak určuje strukturu obvodu a definuje jeho topologii. Obvod je možné jednoznačně popsat definicí vlastností jednotlivých větví a určením jeho topologie

Pokud budeme chtít topologii obvodu snadno definovat, je možné to udělat tak, že jednoznačně identifikujeme jednotlivé uzly a následně přiřadíme každé větvi vstupní a výstupní uzel. Tato myšlenka je implementována v datové struktuře net-list, kterou využívá SPICE ve svých stupních souborech.[6]

Na obrázku 13 pak můžeme vidět schéma jednoduchého obvodu (a), jeho topologii (b) a jeho reprezentaci pomocí net-listu (c).




Obrázek 13: a) jednoduchý obvod b) jeho topologie c) a jeho reprezentace pomocí netlistu

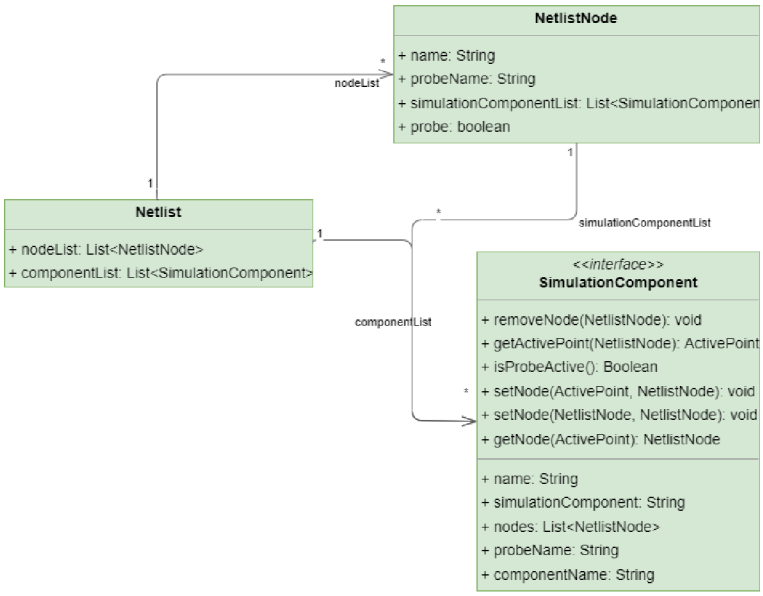
## 6.6 Vytvoření net-listu z dokumentu

Objektem, zodpovědným za reprezentaci net-listu, je třída Netlist, která obsahuje seznam větví/komponentů (SimulationComponent) a seznam uzlů (NetlistNode). Každému uzlu je definován seznam komponent, k němu připojených. Každému komponentu jsou nastaveny uzly. Jednotlivé třídy popisuje diagram na obrázku 14.

Pro sestavení net-listu z dokumentu je podstatné, že DocumentObject má definované aktivní body, které odpovídají svorkám komponent. V místech, kde se dva aktivní body překrývají, pak dochází ke spojení příslušných součástek. Dva a více překrývajících se bodů odpovídá jednomu uzlu ve výsledné topologii.

Samotné vytvoření net-listu probíhá v těchto krocích:

- pro každý shluk aktivních bodů je vytvořen unikátní uzel (NetlistNode),
- je vytvořena mapa aktivních bodů a jim odpovídajícím uzlům,
- každému komponentu (SimulationComponent) jsou pomocí funkce setNode (ActviePoint, NetlisNode) přiřazeny příslušné uzly,
- a současně s předchozím krokem jsou uzlům přiřazeny seznamy připojených komponent,
- odstranění logických spojení.




Obrázek 14: Diagram tříd simulačních souborů

Poslední krok je nezbytný proto, že není předem možné odhadnout, jakým způsobem uživatel komponenty propojí. Jednotlivé komponenty tak, v reprezentaci dokumentu, nejsou napojeny přímo, ale pomocí logické spojnice – objektu linky. Výsledný net-list by logické spoje obsahovat neměl a je proto nutné je odstranit a upravit net-list tak, aby v něm byly reprezentovány pouze komponenty a uzly.

## 6.7 Vstupní soubor NGSpice

Vstupní soubory pro NGSPice se obecně skládají z části definující topologii obvodu a z části, která definuje parametry simulace. Jediná závazná pravidla jsou tato:

- první řádek souboru je vždy určený pro název,
- poslední řádek souboru musí obsahovat „.end“ a znak nového řádku.

Části topologie a parametrů nemusí být striktně odděleny a nezávisí ani na jejich pořadí. Jen započaté bloky musí být vždy ukončeny. Blok „.subckt“ musí být ukončen tagem „.ends“, „.if“ musí být ukončeno „.endif“, atd.

## 6.8 Elementy obvodu ve vstupním souboru

Každý element obvodu je specifikován jedním „řádkem instance“, který obsahuje následující prvky v uvedeném pořadí:

- název elementu instance,

Tabulka 1: Seznam znaků a jim odpovídajícím elementům

| První písmeno | Popis elementu |
|---|---|
| A | XSPICE model kódu |
| B | Zdroj chování (libovolný) |
| C | Kondenzátor |
| D | Dioda |
| E | Zdroj napětí řízený napětím |
| F | Zdroj proudu řízený proudem |
| G | Zdroj napětí řízený proudem |
| H | Zdroj proudu řízený napětím |
| I | Zdroj napětí |
| J | Tranzistor s přechodovým polem |
| K | Spřažené (vzájemné) induktory |
| L | Induktor |
| M | Tranzistor na bázi oxidu kovu |
| N | Číslicové zařízení pro GSS |
| O | Ztrátové přenosové vedení |
| P | Spřažené vícevodičové vedení (CPL) |
| Q | Bipolární tranzistor (BJT) |
| R | Rezistor |
| S | Spínač (řízený napětím) |
| T | Bezeztrátové přenosové vedení |
| U | Rovnoměrně rozložená linka RC |
| V | Zdroj napětí |
| W | Spínač (řízený proudem) |
| X | Dílčí okruh |
| Y | Jednoduché ztrátové přenosové vedení (TXL) |
| Z | Kovový polovodičový tranzistor (MESFET) |

- seznam uzlů, ke kterým je komponenta připojena,
- hodnoty parametrů, které specifikují elektrické vlastnosti daného elementu.

Název elementu instance může obsahovat až 7 znaků. První písmeno názvu vždy specifikuje typ elementu, zbytek je libovolný. Seznam znaků a jim odpovídajícím typům elementů je umístěn v tabulce 1.

### 6.8.1 Řádky příkazů

Příkazy se řadí mezi část vstupního souboru, který řídí parametry a průběh simulace. Příkazy zpravidla začínají tečkou a NGSpice jich nabízí celou řadu. Seznam použitých příkazů znázorňuje tabulka 2.

Tabulka 2: Vybrané příkazy NGSpice

| Příkaz | Funkce příkazu |
|---|---|
| .CONTROL | začátek kontrolní sekce |
| .ENDC | ukončení kontrolní sekce |
| .INCLUDE | používá se na vložení části net-listu na dané místo |
| .LIB | vložení knihovny |
| .MODEL | definice parametrů modelu součástky |
| .OP | zahájení simulace operačních bodů |
| .SUBCKT | zahájení definice dílčího obvodu |
| .ENDS | ukončení definice dílčího obvodu |
| .TRAN | zahájení simulace přechodových jevů |

## 6.9 Elementy simulace pro konkrétní použité součástky

V aplikaci je dostupná knihovna komponent, ze kterých je možné sestavit obvod a následně jej simulovat. Jsou to: zdroj napětí, zdroj proudu, rezistor, cívka (a spřažená cívka), kondenzátor, ideální dioda, NPN-tranzistor, PNP-tranzistor a časovač 555.

### 6.9.1 Zdroj napětí a zdroj proudu

Obě možnosti nabízejí tři různé druhy funkce pro analýzu přechodových jevů: Lineární, Pulz a Sinusoida. Podle typu zdroje se poté do simulačního souboru vkládá element s odpovídajícími parametry.

**Lineární** charakteristika má po celý průběh simulace neměnný proud/napětí a nevyžaduje žádnou jinou charakteristiku. Element net-listu pak vypadá například takto:

V0 n2 n1 5.0

V0 je název komponenty. N2 a n1 jsou vstupní a výstupní uzel a hodnota 5.0 označuje velikost napětí. Zdroj proudu by vypadal identicky, jen na místo V0 by bylo označení například I2 a hodnota 5.0 by označovalo velikost proudu.

**Pulz** je již charakteristika zdroje, která je závislá na čase a vyžaduje bližší specifikaci. Obecná forma funkce pulzu je (konkrétní proměné příkazu popisuje tabulka 3):

PULSE ( V1 V2 TD TR TF PW PER PHASE )

Konkrétní příklad elementu by pak vypadal takto:

VIN 3 0 PULSE(-1 1 2 NS 2 NS 2 NS 50 NS 100 NS)

Tabulka 3: Proměnné příkazu PULSE

| Název proměnné | Význam | Jednotka |
|---|---|---|
| V1 | Výchozí hodnota | V,A |
| V2 | Hodnota při impulzu | V,A |
| TD | Časová odchylka | s |
| TR | Čas vzestupu | s |
| TF | Čas pádu | s |
| PW | Šířka impulzu | s |
| PER | Perioda | s |
| PHASE | Fáze | stupně |

Tabulka 4: Proměnné příkazu SIN

| Název proměnné | Význam | Jednotka |
|---|---|---|
| V0 | Odchylka | V,A |
| VA | Amplituda | V,A |
| FREQ | Frekvence | Hz |
| TD | Zpoždění | s |
| THETA | Tlumicí faktor | 1/s |
| PHASE | fáze | stupně |

**Sinusoida** je charakteristika zdroje se sinusovým průběhem, obecný popis funkce pak vypadá následovně (konkrétní proměné příkazu, tak popisuje tabulka 4):

VIN 3 0 PULSE(-1 1 2 NS 2 NS 2 NS 50 NS 100 NS)

Konkrétní příklad elementu:

VIN 3 0 SIN(0 1 100 MEG 1 NS 1 E10)

### 6.9.2 Diody a Tranzistory

Diody a tranzistory jsou elementy, které jsou specifikovány pomocí modelu. Pro oba typy komponent poskytuje NGSpice výchozí model, který by měl odpovídat obecné charakteristice komponenty. Pro oba typy součástek byly použity právě tyto výchozí modely z důvodu jednoduchosti použití pro uživatele.

### 6.9.3 Časovač 555

Element časovače již nebylo možné řešit pomocí předdefinovaných komponent, proto byla využita standardizovaná knihovna SPICE, která tento komponent obsahuje. Po importu knihovny pak výsledný element vypadá takto:

X0 n8 n4 n5 n2 n1 n3 n6 ua555

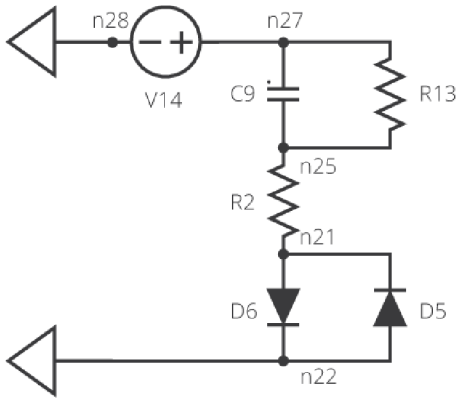
X0 je označení pro komponent dílčího obvodu a jeho jméno. N8 je uzel odpovídající svorce spouštěče (trigget), n4 výstupu, n5 resetu, n2 řídícímu napětí, n1 prahové hodnoty, n3 vybití (discharge) a n6 odpovídá uzlu na svorce +Vcc. Ua555 je potom odkaz na knihovnu dílčího obvodu.

## 6.10 Generace vstupního souboru

Za generaci je zodpovědná třída SimulationFile, která současně nese i veškeré informace spojené se simulací dokumentu. Třída obsahuje funkci createSimulationList, ta vrací list textových řetězců, kde každý odpovídá jednomu řádku vstupního souboru. Za tvorbu jednotlivých textových řetězců elementů net-listu jsou zodpovědné jednotlivé třídy implementující SimulationComponent. Třída SiulationFile tyto řetězce formátuje a doplní o blok „.CONTROL“ obsahující instrukce k simulaci.

## 6.11 Konkrétní příklad obvodu a výsledného vstupního souboru

Pro obvod uvedený na obrázku 15 by pak generovaný soubor vypadal následovně (znakem # jsou označeny komentáře popisující soubor):




Obrázek 15: Jednoduchý obvod s označenými uzly a komponentami

```
* Title: Test circuit

#net-list komponent popisující obvod
* Netlist
R1 n22 0 0
R2 n25 n21 470000.0
```

```
D5 n22 n21 DMOD
.model DMOD D
D6 n21 n22 DMOD
.model DMOD D
C9 n25 n27 0.22 ic=0.0
R13 n27 n25 160.0
V14 n27 n28 SIN(0.0 230.0 50.0 0.001 0.0 0.0)
R15 n28 0 0

#blok příkazů
.control
op
tran 0.1 5.0 0 uic
echo output test > result.txt $ start new file

#nastavení režimu výpisu pro formátování výstupního souboru
set appendwrite
set wr_vecnames
set wr_singlescale

#vypsání výsledků simulace pro všechny vektory
wrdata data.txt all
.endc
.end
```

### 6.11.1 Výsledky simulace a jejich zpracování

Výsledkem simulace je textový soubor, který vždy obsahuje název simulované proměnné a výsledné hodnoty. Výsledky jsou nahrány a parsovány do datových struktur vhodných pro použití v grafu panelu simulace a jeho ostatních grafických komponentech.

# 7 Programátorská příručka

Systém Maven umožňuje specifikaci veškerých závislostí v souboru pom, který je součástí zdrojového kódu. IDE jako NetBeans nebo Intelij Idea registrují zdrojové sobory Maven jako již spustitelné projekty. Pro sestavení zdrojových kódů pak stačí jen otevřít příslušnou složku jako nový projekt. Případně zvolit možnost „projekt s existujícím zdrojovým kódem“ a zvolit soubor pom.

## 7.1 Testování aplikace

Aplikace byla z velké části vyvíjena metodikou vývoje řízenou testy. Stěžejní funkcionality a algoritmy jsou tedy pokryty automatickými JUnit testy. Zby-

tek aplikace byl testován manuálně v prostředí MS Windows 10. Byli testovány použité verze Java 11.0.15 – 17.0.2.

# 8 Uživatelská příručka

Aplikace umožňuje sestavování obvodů z knihovny komponent a jejich následnou simulaci. V programu jsou také dostupná některá ukázková zapojení a informace o komponentách. Aplikace je určena pro začínající zájemce o elektroniku.

## 8.1 Systémové požadavky

Minimální konfigurace systému:

- MS Windows 10,
- 300MB volné RAM,
- 40MB volného místa na disku (velikost spustitelného souboru),
- naistalována Java 11 a vyšší.

## 8.2 Panely aplikace a jejich prvky

Aplikace obsahuje dva samostatné panely, panel návrhu a panel simulace, mezi kterými je možné libovolně přepínat. Dále je v aplikaci k dispozici panel informací o komponentách, který lze vyvolat z knihovny komponent.

## 8.3 Panel návrhu

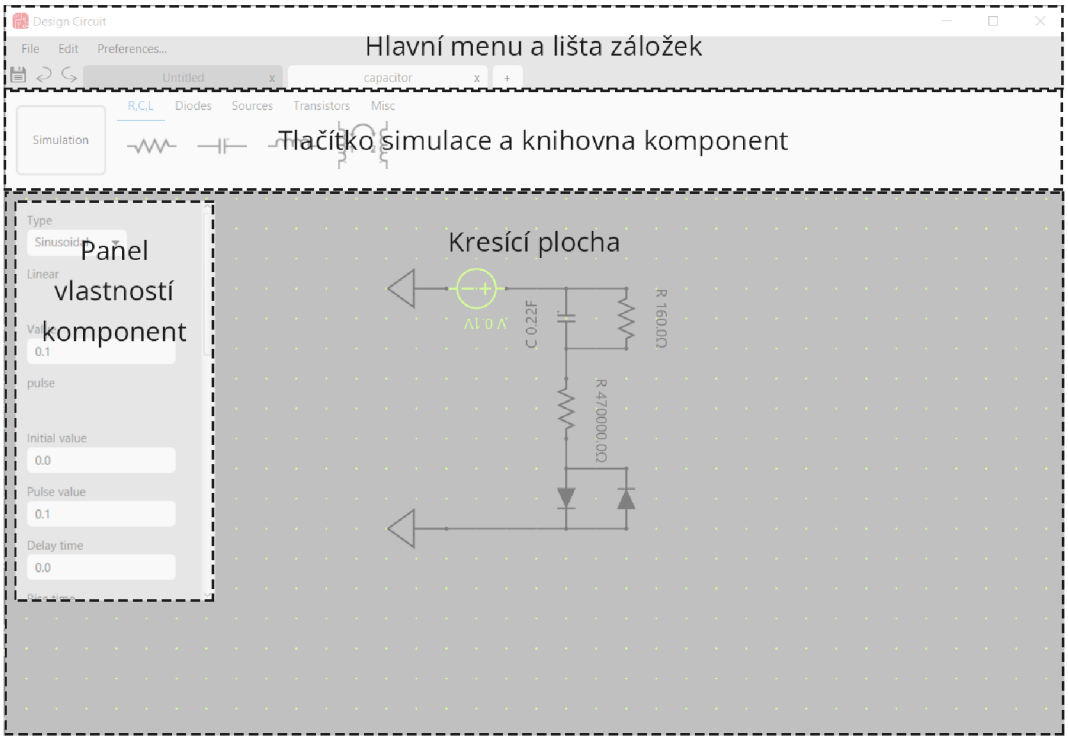
Rozložené jednitlivých elemntů panelu návrhu je uvedeno na obrázku 16.

**Hlavní menu a lišta záložek** – je hlavním kontrolním prvkem aplikace. Umožňuje práci se soubory a dokumenty a také poskytuje uživateli přístup k editačním funkcím kreslící plochy.

**Tlačítko simulace a knihovna komponent** – je ovládací panel dostupný pouze z obrazovky návrhu. Umožňuje přechod do panelu simulace, vkládání komponent do kreslící plochy a vyvolání panelu informace o komponentách.

**Panel vlastností komponent** – je specifický pro danou komponentu, není viditelný vždy. Zobrazuje se při dvoj-kliku na komponentu, která má uživatelsky nastavitelné hodnoty.

**Kreslící plocha** – je centrální částí celého panelu, umožňuje tvorbu a editaci elektronických obvodů.




Obrázek 16: Panel návrhu a jeho části

### 8.3.1 Hlavní menu a lišta záložek

V pravé horní části jsou v panelu dostupné možnosti soubor (file), úpravy (edit) a možnosti (preferences). Vpravo je dostupné menu rychlých akcí. Spodní centrální část obsahuje panel záložek (viz. obrázek 17).


Obrázek 17: Hlavní menu a lišta záložek

**Tlačítko uložit** – uloží aktivní dokument, pokud dokument nebyl doposud uložen, vyvolá možnost uložit jako.

**Tlačítko zpět** – vrátí poslední provedenou změnu.

**Tlačítko znovu** – opět provede vrácenou změnu.

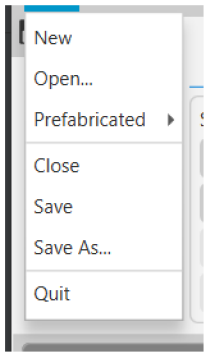
**Lišta záložek** – obsah odpovídá seznamu otevřených dokumentů. Kliknutím na konkrétní záložku se nastaví odpovídající dokument jako aktivní. Kliknu-

tím na křížek v pravé části tlačítka dojde k zavření odpovídajícího dokumentu.

**Tlačítko nového dokumentu** – se nachází v pravé části panelu záložek a po kliknutí otevře nový dokument.

### 8.3.2 Menu souboru

Menu souboru obsahuje možnosti manipulace se soubory a dokumenty a obsahuje možnost ukončení aplikace (viz. obrázek 18).




Obrázek 18: Menu souboru

**Nový (New)** – otevře nový dokument a nastaví ho jako aktivní.

**Otevřít (Open...)** – vyvolá dialog výběru souboru.

**Ukázkové obvody (Prefabricated)** – otevře podmenu ukázkových dokumentů.

**Zavřít (Close)** – uzavře aktuální záložku, pokud jsou v záložce neuložené změny, vyžádá nejprve potvrzení.

**Uložit (Save)** – pokud byl dokument již uložen, přepíše stávající soubor aktuálním stavem dokumentu. Pokud nebyl, vyvolá dialog uložit jako.

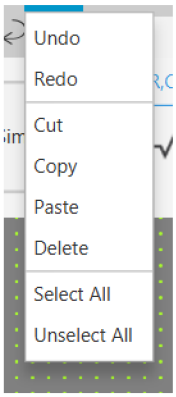
**Uložit jako... (Save as...)** – vyvolá dialog uložit jako.

**Ukončit (Quit)** – vyvolá potvrzovací dialog. Po potvrzení uzavře všechny záložky a ukončí program.

### 8.3.3 Menu úprav

Menu editace obsahuje editační funkce pro manipulaci elementů kreslící plochy (viz. obrázek 19).

**Zpět (Undo)** – vrátí poslední provedenou změnu.




Obrázek 19: Menu úprav

**Znovu (Redo)** – znovu provede vrácenou změnu.

**Vyjmout (Cut)** – vyjme označené objekty z kreslící plochy a přesune je do schránky.

**Kopírovat (Copy)** – přesune označené objekty kreslící plochy do schránky.

**Vložit (Paste)** – vloží objekty ze schránky do kreslící plochy.

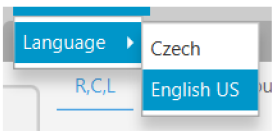
**Smazat (Detele)** – odstraní označené objekty z kreslící plochy.

**Označit vše (Select All)** – označí všechny objekty kreslící plochy.

**Zrušit všechna označení (Unselect All)** – zruší všechna označení objektů kreslící plochy.

### 8.3.4 Menu možností

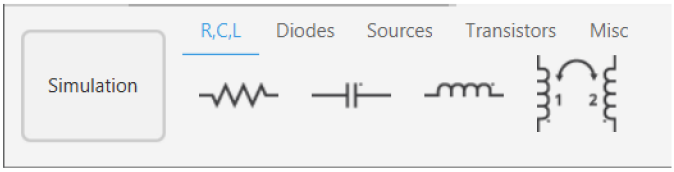
Menu možností obsahuje možnost volby jazyka, který nejprve vyvolá dialog potvrzení a poté nastaví požadovaný jazyk (viz. obrázek 20). Změna jazyka se projeví po restartu aplikace.




Obrázek 20: Menu možností

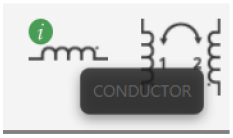
### 8.3.5 Tlačítko simulace a knihovna komponent

Knihovna komponent (viz. obrázek 21) umožňuje zobrazení komponent v logických kategoriích. Kliknutím na tlačítko kategorie se příslušně opraví obsah knihovny. Přetažením komponentu z knihovny do kreslící plochy dojde ke vložení zvoleného komponentu na požadované místo. Po kliknutí na tlačítko simulace program přejde na panel simulace.




Obrázek 21: Tlačítko simulace a knihovna komponent

Při najetí na libovolné tlačítko komponenty z knihovny se zobrazí ikona informace (viz. obrázek 22) a tooltip s názvem. Po kliknutí na tlačítko informace se zobrazí panel informací o komponentě. Pokud má daný komponent svou vlastní stránku „informace o komponentě“, otevře se panel právě na ní. Pokud komponent nemá informace dostupné, zobrazí se panel informací s poslední viditelnou stránkou informací (nebo se stránkou výchozí).




Obrázek 22: Tlačítko informace o komponentě a tooltip

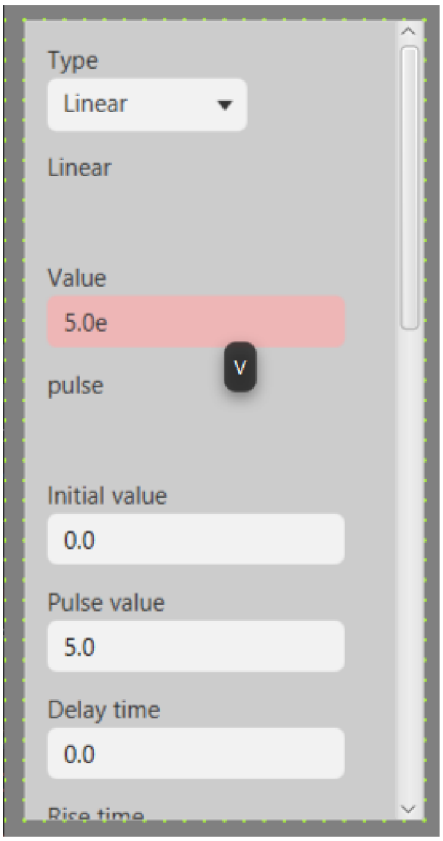
### 8.3.6 Panel vlastností komponent

Panel vlastností umožňuje nastavovat jednotlivé charakteristiky komponent. Je specifický pro každou komponentu a zobrazuje se dvoj-klikem na ikonu konkrétní součástky v kreslící ploše. Při najetí do okénka vlastností se také zobrazí tooltip s jednotkou dané charakteristiky (pokud charakteristika jednotku má). Náhled panelu je zobrazen na obrázku 23.

Textové pole panelu nabízí také validaci hodnot a její vizuální indikaci. Pokud uživatel vyplní do pole neplatnou hodnotu, toto pole se zabarví červeně a zůstane tak, dokud není hodnota opravena nebo automaticky nahrazena hodnotou před editací.

### 8.3.7 Kreslící plocha

Kreslící plocha umožňuje tvorbu a editaci elektronických obvodů. Editaci v kreslící ploše lze provádět třemi různými způsoby: pomocí menu editace, klávesovými




Obrázek 23: Panel vlastností, tooltip jednotky a indikace neplatné hodnoty

zkratkami nebo pomocí kontextového menu kreslící plochy (viz. obrázek 24).

### 8.3.8 Kontextové menu kreslící plochy

Kontextové menu je vyvoláno pravým klikem do kreslící plochy a obsahuje následující možnosti:

**Kopírovat (Copy selection)** – zkopíruje označené objekty do schránky.

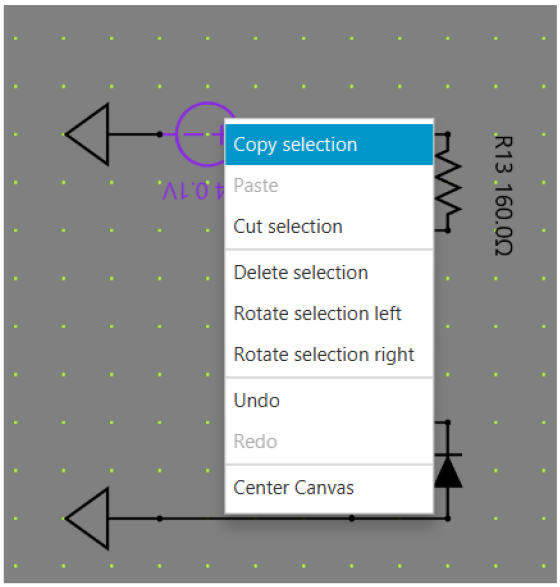
**Vložit (Paste)** – vloží objekty ze schránky do kreslící plochy.

**Vyjmout (Cut selection)** – smaže vybrané objekty z kreslící plochy a umístí je do schránky.

**Smazat (Delete selection)** – smaže vybrané objekty.

**Rotovat vlevo (Rotate selection left)** – otočí vybrané objekty po směru hodinových ručiček.

**Rotovat vpravo (Rotate selection right)** – otočí vybrané objekty proti směru hodinových ručiček.




Obrázek 24: Kreslící plocha a její kontextové menu

Tabulka 5: Seznam klávesových zkratek

| Zkratka | Název funkce | Popis |
|---|---|---|
| R | Rotace vlevo | Rotuje vybrané objety po směru hodinových ručiček |
| Alt + R | Rotace vpravo | Rotuje dané objekty proti směru hodinových ručiček |
| Delete | Smazat | Smaže vybrané objekty |
| Ctrl + A | Označit vše | Označí všechny objekty kreslícího plátna |
| Ctrl + C | Kopírovat | Zkopíruje vybrané objekty do schránky |
| Ctrl + V | Vložit | Vloží objekty ze schránky do pracovní plochy |

**Zpět (Undo)** – vrátí poslední provedenou změnu.

**Znovu (Redo)** – znovu provede vrácenou změnu.

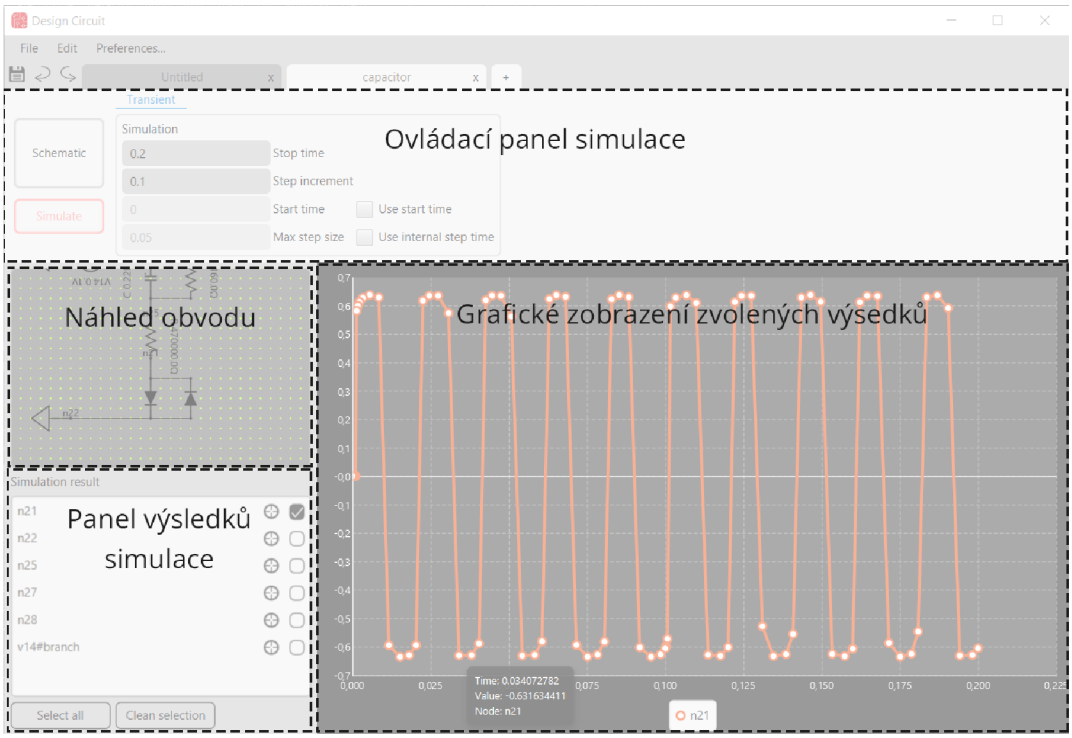
**Vycentrovat (Center canvas)** – nastaví pozici kreslící plochy tak, aby byl počátek ve středu okna.

### 8.3.9 Seznam klávesových zkratek kreslící plochy

Pokud je kreslící plocha aktivní, je možné pro editaci používat také klávesové zkratky. V aplikaci dostupné zkratky jsou popsány v tabulce 5.

## 8.4 Panel Simulace

Panel simulace je možné vyvolat tlačítkem simulace v panelu návrhu. Umožňuje provést simulaci s nastavenými parametry a interpretovat její výsledky (viz. obrázek 25).




Obrázek 25: Panel simulace a jeho elementy

**Ovládací panel simulace** – umožňuje přechod no panelu návrhu, nastavení parametrů simulace a spuštění simulace.

**Náhled obvodu** – zobrazuje obvod s popisky jmen uzlů. Je možné s ním manipulovat stejně jako s kreslící plochou.

**Panel výsledků simulace** – zobrazuje seznam dostupných výsledků a umožňuje nastavit, které z nich jsou viditelné v grafu. Uživatel zde také může centrovat náhled obvodu na příslušný uzel/komponentu.

**Grafické zobrazení zvolených výsledků** – umožňuje zobrazení průběhu dané veličiny v čase. Po najetí do grafu se zobrazuje tooltip s konkrétními hodnotami. Obsah grafu odpovídá zvoleným výsledkům simulace z panelu výsledků simulace.

### 8.4.1 Nastavení parametru simulace

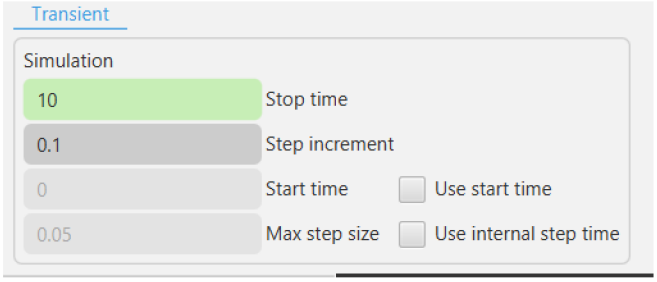
Panel nastavení parametru simulace (viz. obrázek 26) nabízí podobný způsob validace jako panel nastavení vlastností komponent. Panel poskytuje následující možnosti nastavení:

**Čas konce simulace (Stop time)** – čas konce simulace v sekundách.

**Čas kroku (Step Increment)** – velikost simulačního kroku v sekundách

**Čas začátku (Start time)** – čas začátku simulace (použit pouze pokud je zaškrtnuta možnost „použít začátek simulace“). Při volbě probíhá simulace od zvoleného času na místo od výchozí nuly.

**Maximální délka kroku (Max step size)** – maximální délka kroku u automaticky generované délky kroku (je použito pouze pokud je zvolena možnost „použít vnitřní délku kroku“).



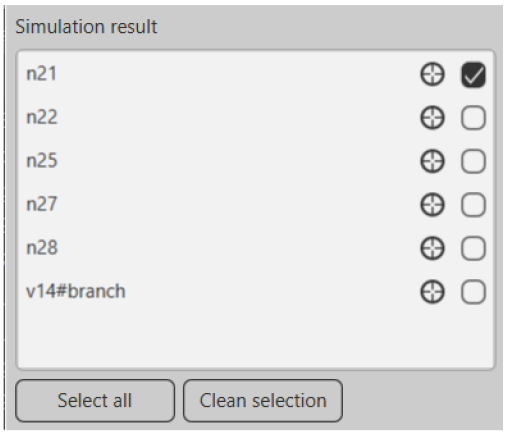

Obrázek 26: Nastavení parametrů simulace

### 8.4.2 Panel výsledků simulace

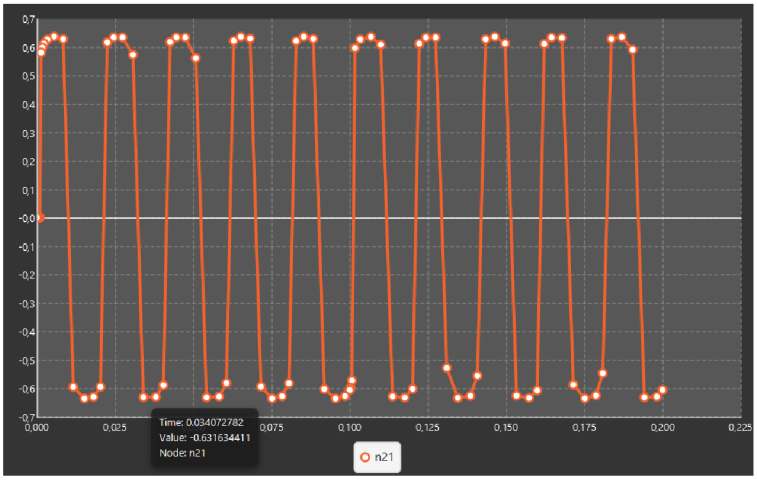
Panel výsledků (viz. obrázek 27) obsahuje seznam uzlů/komponent, ke kterým byly sestaveny výsledky simulace. Zaškrtnutím volby v pravé části každého řádku se volí, zda je výsledek viditelný v grafu. Tlačítko vlevo od checkboxu vycentruje náhled obvodu nad zvolený uzel/komponent. Volby ve spodní části grafu slouží k označení všech výsledků, anebo naopak ke zrušení všech označení.

### 8.4.3 Graf výsledků simulace

Výsledky simulace jsou zobrazovány v klasickém spojnicovém grafu (viz. obrázek 28). Rozsah grafu je nastavován automaticky podle zobrazovaných hodnot. Osa x zobrazuje čas simulace, osa y hodnotu v daném čase (napětí nebo proud v závislosti na typu výsledku). Ve spodní části grafu je dostupná legenda. Při najetí myši na konkrétní bod grafu se zobrazí tooltip s konkrétním časem, hodnotou a názvem uzlu/komponenty.




Obrázek 27: Panel výsledků simulace



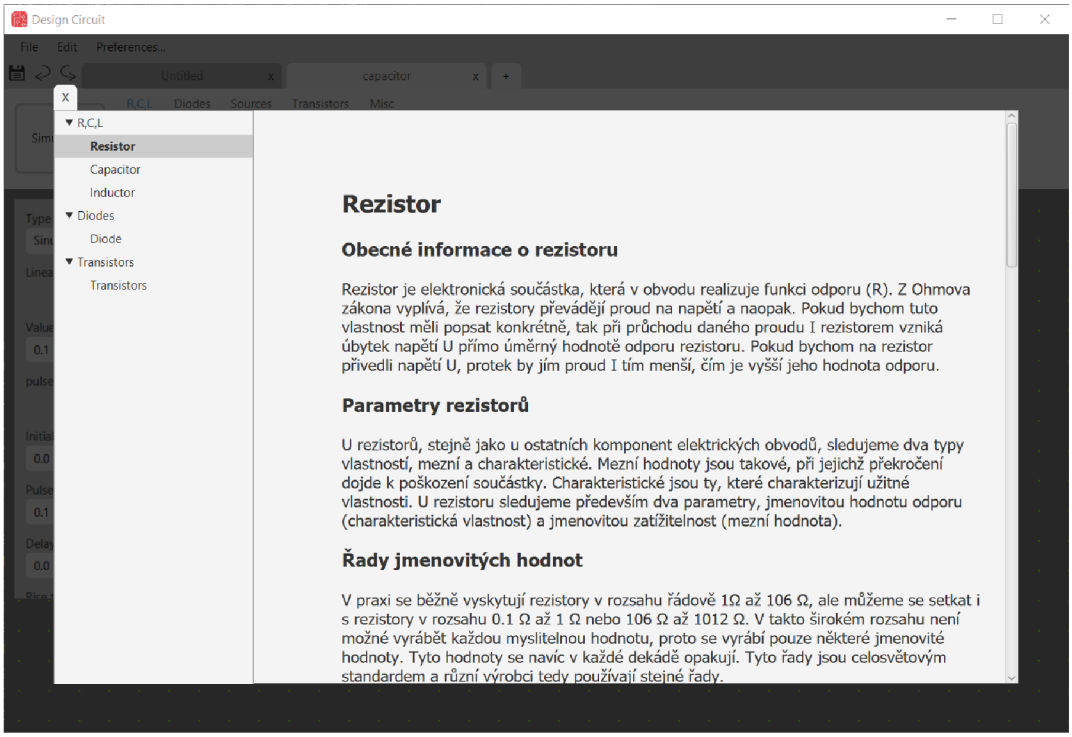

Obrázek 28: Graf výsledků simulace

## 8.5 Panel informací o komponentách

Panel informací (viz. obrázek 29) o komponentách je vyvolán pomocí knihovny komponent. V levé části je stromová struktura dostupných stránek informací a jejich kategorií. Při volbě se konkrétní stránka nastaví do centrální části panelu a je dostupná k prohlížení. Panel je možné ukončit kliknutím mimo plochu panelu anebo křížkem v levé horní části.

## 8.6 Metodika práce se soubory a dokumenty

Program nabízí tři možnosti vytvoření dokumentu: vytvoření prázdného dokumentu, nahrání dokumentu ze souboru a nahrání dokumentu z ukázkových zapojení.

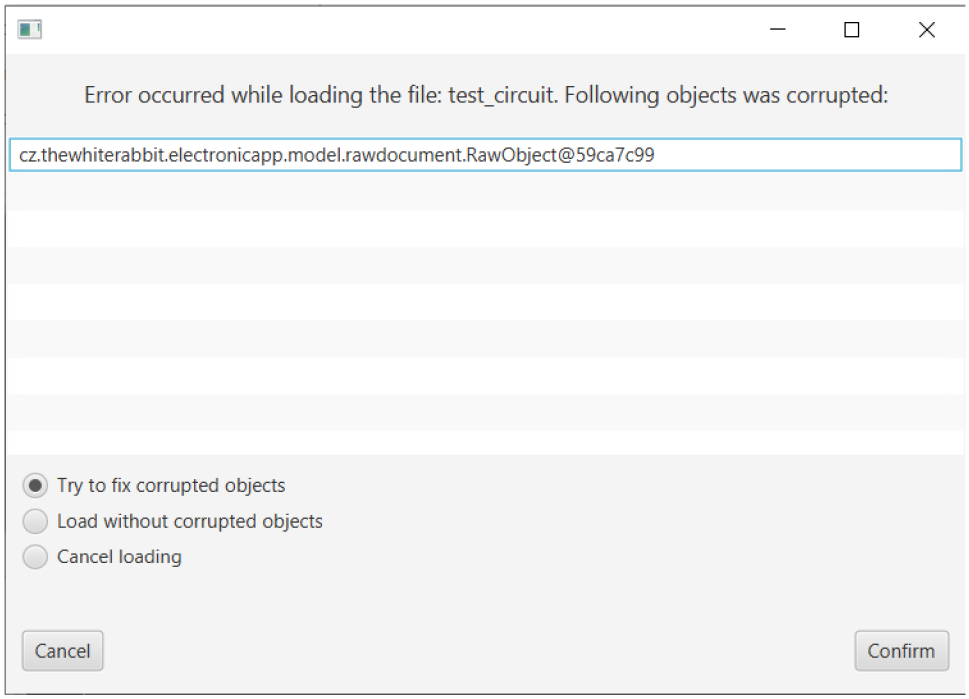



Obrázek 29: Obrazovka informací o komponentách

**Vytvoření nového dokumentu** – může být provedeno z menu nebo lišty záložek. Nový dokument je také automaticky vytvořen při startu programu a při uzavření poslední záložky z lišty záložek. Nemůže tedy nikdy nastat stav, že by v programu nebyl otevřen žádný dokument.

**Otevření dokumentu ze souboru** – otevřít dokument ze souboru je možné volbou z menu souboru. Tato volba vyvolá dialog otevření souboru, kde je možné standardně zvolit soubor z disku. Oficiální soubory programu jsou označeny koncovkou „.aeon“, ale program je schopný načíst jakékoliv textové soubory ve správném datovém formátu. Při načítání souboru může dojít k detekci poškozených nebo jen nekompletních dat. V takovém případě je zobrazen dialog opravy souboru (viz. obrázek 30). Uživatel pak má volbu pokusit se poškozené objekty opravit, vynechat poškozené objekty, nebo úplně zrušit nahrávání. Dokumenty vytvořené ze souboru si pamatují soubor, ze kterého byly vytvořeny a při ukládání ho automaticky přepisují

**Otevření dokumentu ukázkového zapojení** – je možné provést z menu souboru. Chování této funkcionality je totožné s načítáním ze souboru. Výjimkou je ukládání souboru. Ukázkový obvod lze uložit jako nový soubor, ale nikdy nelze původní soubor přepsat.




Obrázek 30: Dialog poškozených souborů

## 8.7 Metodika tvorby obvodu

Postup tvorby obvodu se skládá z několika úkonů:

- vložení komponent z knihovny na požadovaná místa v obvodu,
- nastavení parametrů komponent,
- vytvoření spojnic mezi komponenty.

### 8.7.1 Vložení komponenty do kreslící plochy

Komponent je možné vkládat do pracovní plochy přetažením ikony dané součástky z knihovny. Jakmile je kurzor přetažen do kreslící plochy, je graficky indikováno místo, kam bude komponent vložen. Po uvolnění myši pak dojde k samotnému vložení.

### 8.7.2 Označování komponent

Je možné označovat samostatné komponenty i jejich skupiny. Označení konkrétního komponentu probíhá kliknutím na jeho ikony v kreslící ploše. Označení více komponent je pak možné skupinovou selekcí. Kliknutí do kreslící plochy mimo komponent a následné tažení vyvolá funkci skupinového označení. Oblast, která má být označena, je indikována vykresleným obdélníkem.

Pokud je při kterémkoli z výše zmíněných druhů selekce držena klávesa Ctrl, budou vybrané objekty přidány k současnému výběru.

Označené komponenty jsou barevně odlišeny od neoznačených.

### 8.7.3 Manipulace vybraných komponent

Vybrané komponenty lze libovolně posouvat tažením po kreslící ploše. Je možné je také mazat pomocí kontextového menu nebo klávesové zkratky Delete. Poslední možnou manipulací objektů je jejich rotace. To je opět možné provést z kontextového menu nebo klávesovou zkratkou R/Alt+R. Objekty vždy rotují podle své vlastní osy. Pokud je tedy zvoleno více objektů, nebude otočen celý výběr, ale každý objekt samostatně.

## 8.8 Kopírování a vkládání komponent

Označené komponenty lze kopírovat z kontextového menu nebo pomocí klávesové zkratky Ctrl+C. Zkopírované objekty je možné vložit opět pomocí menu nebo klávesové zkratky Ctrl+V. Objekty jsou vkládány na stejné pozice, odkud byly zkopírovány. Při vkládání jsou zrušena všechna označení. Jediné označené objekty jsou ty vložené.

### 8.8.1 Nastavení parametrů komponent

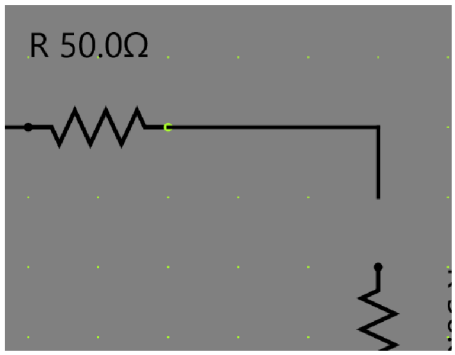
Pokud má komponent nastavitelné vlastnosti, je možné zobrazit dialog nastavení těchto vlastností dvojklikem na ikonu daného komponentu. Obsah dialogu je vždy specifický pro daný komponent. Při změně hodnoty v dialogu je barevně indikována správnost vložené hodnoty. Pokud je textové pole zelené, je hodnota změněna a ve správném tvaru. Pokud je textové pole červené je hodnota změněna a nesprávná. Šedá pole jsou aktuální, nezměněné hodnoty.

### 8.8.2 Vytvoření spojnice mezi komponenty

Jednotlivé ikony komponent v kreslící ploše mají definována aktivní místa tam, kde by se nacházely svorky komponenty. Tažením tohoto aktivního místa je do kreslící plochy vložena čára(viz. obrázek 31). Současně se mohou vkládat až dvě spojené čáry, jedna pro osu x a druhá pro osu y. Po vložení čar dojde k jejich optimalizaci. Při ní jsou vložené čáry rozděleny v místech průniku s aktivními body nebo jinými čarami. Naopak tam, kde jsou čáry děleny a nemusí, jsou spojeny.

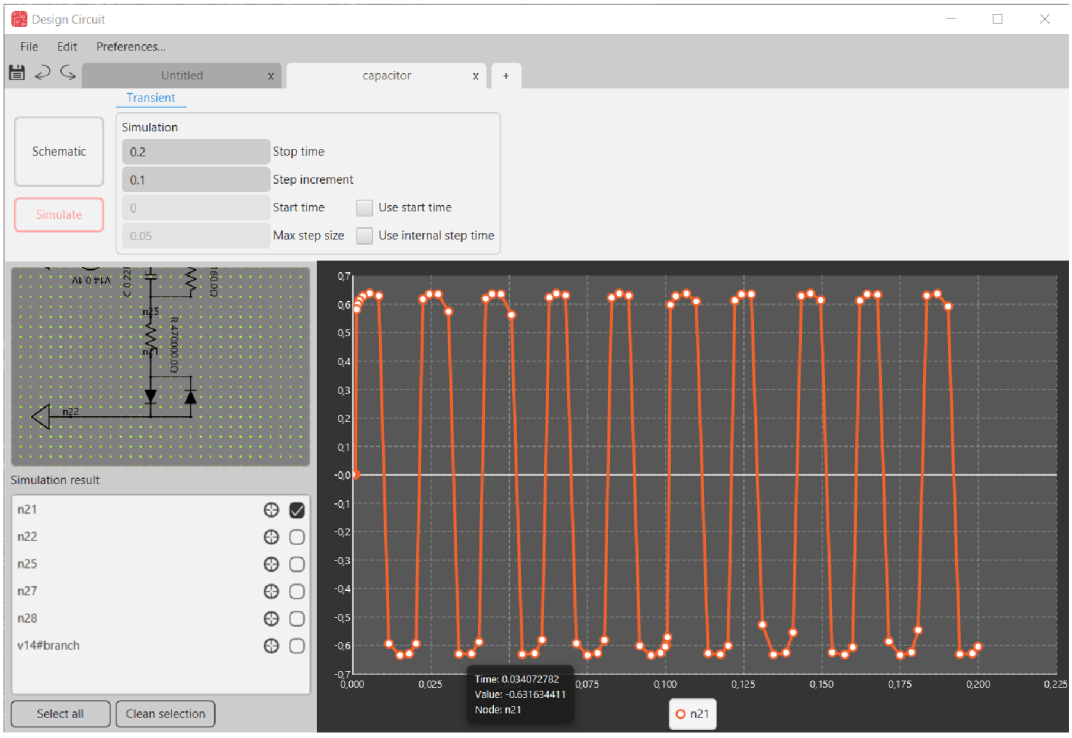
## 8.9 Metodika simulace

Po vytvoření obvodu je možná jeho simulace, jež se provádí v panelu simulace. Dokud nebyla simulace provedena, je panel simulace prázdný (graf ani seznam výsledů neobsahují žádná data). Výsledky se získají stisknutím tlačítka „Simuluj“




Obrázek 31: Aktivní bod a z něho tažená spojnice

z řídícího panelu simulace. Po stisku je vyvolán dialog simulace, který indikuje probíhající výpočet. Po dokončení jsou do panelu automaticky nahrány výsledky (viz. obrázek 32).




Obrázek 32: Obrazovka simulace zobrazující výsledky

Výsledky simulace jsou uloženy až do spuštění další simulace. Jsou také ukládány napříč jednotlivými dokumenty. Graf simulace má nastaven automatický rozsah, může tedy dojít k situaci, kdy budou výsledky v grafu nečitelné, z důvodu velké hustoty dat. V takovém případě je potřeba zkrátit časový interval simulace tak, aby byly výsledky lépe čitelné.

# Závěr

Byla vytvořena aplikace, která umožňuje sestavování elektronických obvodů a jejich následnou analýzu pomocí simulace. Aplikace splňuje všechny předem definované požadavky a navržené funkcionality, avšak mohla by být dále rozšířena o další komponenty a možnosti simulace.

Návrh architektury grafického editoru je dostatečný v daném rozsahu aplikace, ale v případě dalšího rozvoje by bylo pravděpodobně nutné ho upravit. Při návrhu se nepočítalo s nestandardním chováním některých grafických objektů, a proto bylo zapotřebí při práci s těmito objekty zvolit jiný způsob práce než s ostatními. Do programu tím byla vnesena zbytečná složitost, kterou by bylo pravděpodobně nutné před dalším rozvojem odstranit.

Návrh struktury souboru je také dostatečný a zcela plní svoji funkci. Avšak mohl by být, ale rozšířen o další možnosti validace dat, jako je například kontrolní součet.

Využití eventů a agregátorů událostí v aplikaci se ukázalo jako správný přístup, jenž snížil vzájemnou závislost jednotlivých částí grafického rozhraní a umožnil jeho snadnou implementaci. Díky tomuto přístupu může být aplikace snadno dále upravována a rozšiřována.

Momentálně je aplikace vhodná a dobře použitelná pro zájemce o elektroniku, a je možné s ní pohodlně uživatelsky pracovat. Po menších opravách některých chyb v návrhu bude aplikace dobře připravena na další rozšiřování o funkcionality editoru i simulace a mohla by mít potenciál například ve výuce elektronických obvodů na školách, anebo jako open source projekt.

# Conclusions

As a result of the thesis was created an application that allows users to create circuits and analyze them using simulation. Application meet all of the initial requirements and suggested functionalities, but it can be further expanded with more component models and simulation options.

The design of the drawing area is sufficient considering the current scope of the application, but in case of further development minor changes would be necessary. The initial design did not take into count the irregular behavior of some objects, so the basic manipulation of these objects is slightly different than with regular objects. This brought unnecessary complexity to the application, which should be removed before further development.

The design of the file structure is sufficient but could be further expanded by more validation options for example control count.

Usage of the events and events aggregators was a good choice, which led to the independence of the components and allow quick and easy graphical user interface development. Thanks to that application can be easily further developed.

In conclusion, application is well suited for users interested in electronics and for possible usage in education. After minor changes in the design will application be prepared for further development.

# A Obsah přiloženého datového média

**`bin/`**
: Adresář „bin“ obsahuje soustitelný soubor PROGRAMU PRO ELEKTRONIKU a instalátor příslušné verze Java.

**`doc/`**
: Adresář „doc“ kompletní text práce.

**`src/`**
: Adresář „src“ obsahuje zdojové kódy programu.

**`readme.txt`**
: Soubor „readme.txt“ obsahuje instrukce ke spuštění aplikace.

# Literatura


[1] *idealCircuit Simulator*. Dostupný z: ⟨https://www.sidelinesoft.com/ic/f⟩.

[2] *Circuit Lab. : Circuit simulation and schematics*. Dostupný z: ⟨https://www.sidelinesoft.com/ic/f⟩.

[3] Fowler, Martin (ed.). *GUI Architectures*. Dostupný z: ⟨https://www.martinfowler.com/eaaDev/uiArchs.html⟩.

[4] Fowler, Martin (ed.). *Event Aggregator*. Dostupný z: ⟨https://www.martinfowler.com/eaaDev/EventAggregator.html⟩.

[5] Vogt, Holt (ed.). *Ngspice User's Manual Version 36 plus*. 703 s. Dostupný z: ⟨http://ngspice.sourceforge.net/docs/ngspice-manual.pdf⟩.

[6] Ogrodzki, Jan. *CIRCUIT SIMULATION METHODS and ALOGORITHMS*. First. 2000 Corporate Blvd., N.W., Boca Raton, Florida 33431: CRC Press, Inc., 2000. 465 s. ISBN 0-8493-7894-X.