**Technická dokumentácia projektu**

(Textový editor obohatený o grafické prvky)

*Tímový projekt*

|  |  |
| --- | --- |
| **Autor:** | tím č.10 – Innovators |
| **Téma projektu:** | textový editor obohatený o grafické prvky (TrollEdit) |
| **Vytvorený:** | 02.10. 2011 |
| **Stav:** | predbežný |
| **Vedúci projektu:** | Ing. Peter Drahoš |
| **Vedúci tímu:** | Bc. Lukáš Turský |
| **Členovia tímu:** | Bc. Marek Brath  Bc. Adrián Feješ  Bc. Maroš Jendrej  Bc. Jozef Krajčovič  Bc. Ľuboš Staráček |
| **Kontakt:** | tp-team-10@googlegroups.com |

Obsah

[1 Úvod 4](#_Toc308465428)

[2 Analýza 5](#_Toc308465429)

[2.1 Existujúce riešenia 5](#_Toc308465430)

[2.2 Analýza predchádzajúceho riešenia nástroja TrollEdit 5](#_Toc308465431)

[2.2.1 Inicializácia editora, otvorenie súborov 5](#_Toc308465432)

[2.2.2 Práca s editorom 5](#_Toc308465433)

[2.2.3 Programovanie v editore 6](#_Toc308465434)

[2.2.4 Komentáre 7](#_Toc308465435)

[2.2.5 Bloky 7](#_Toc308465436)

[2.2.6 Práca so súbormi, prílohami 7](#_Toc308465437)

[2.2.7 Syntaktický analyzátor 8](#_Toc308465438)

[2.2.8 Gramatika 8](#_Toc308465439)

[2.2.9 Literate programming 8](#_Toc308465440)

[2.3 Analýza použitých technológií 9](#_Toc308465441)

[2.3.1 Qt 9](#_Toc308465442)

[2.3.2 Qt Quick a jazyk QML 9](#_Toc308465443)

[2.3.3 Jazyk Lua 11](#_Toc308465444)

[2.3.4 Knižnica LPeg 11](#_Toc308465445)

[2.3.5 RTF 12](#_Toc308465446)

[3 Špecifikácia požiadaviek 13](#_Toc308465447)

[3.1 Funkcionálne požiadavky 13](#_Toc308465448)

[3.2 Nefunkcionálne požiadavky 14](#_Toc308465449)

[4 Návrh riešenia 15](#_Toc308465450)

[4.1 Diagram prípadov použitia 15](#_Toc308465451)

[4.2 Architektúra programu 18](#_Toc308465452)

[4.3 Návrh GUI 19](#_Toc308465453)

[5 Implementácia prototypu 22](#_Toc308465454)

[5.1 Popis prototypu 22](#_Toc308465455)

[6 Testovanie 23](#_Toc308465456)

[6.1 Akceptačné testy pre overenie funkcionality 23](#_Toc308465457)

# Úvod

Súčasné textové editory zdrojových kódov len minimálne využívajú možnosti grafickej reprezentácie, čo je veľká škoda vzhľadom na to, že práve obohatenie editorov o grafické prvky by mohlo v mnohých veciach uľahčiť prácu s takýmto editorom. Sprehľadnil by sa zdrojový kód, zjednodušila a zefektívnila nie len jeho tvorba, ale aj údržba a prezentácia, a vnieslo by to možnosť nového pohľadu na integráciu dokumentácie s programom.

Práve to by malo byť výsledkom tohto projektu, ktorého cieľom bude pokračovať vo vývoji multiplatformového editora „TrollEdit“ (ktorý bol riešení v roku 2009/10 tímom s názvom UFOPAK) pre editovanie najmä zdrojových kódov, ktorý bude využívať grafické prvky na zjednodušenie a zefektívnenie práce programátora. Našim zameraním bude rozšírenie stávajúcej funkcionality do podoby vhodnej pre reálne nasadenie editora do praxe.

Tento dokument obsahuje zhrnutie všetkých riešení nášho tímu na tomto projekte od analýzy až po implementáciu.

# Analýza

## Existujúce riešenia

//Sem by mohlo isť nejaké porovnania súčasných editorov (Scite, JuffEed etc.) tak stručne //o editoroch ich klady a zápory a zhodnotenie

//Nakoniec nejaká tabuľka kde budú porovnania funkcionality

Vytvorí Marek

## Analýza predchádzajúceho riešenia nástroja TrollEdit

Vzhľadom na to, že pokračujeme na projekte, ktorý bol vyvíjaný v rámci minuloročného tímového projektu bolo nutné vykonať podrobnú analýzu predchádzajúceho riešenia. Výsledkom je porovnanie medzi reálnym stavom editora a technickou dokumentáciou minulého tímu. Správa o stave bola rozdelená podľa jednotlivých funkčných častí.

### Inicializácia editora, otvorenie súborov

#### Implementované:

* pri načítaní súboru určenie správnej gramatiky a jej kontrola
* pri otvorení súboru automatická analýza a zobrazenie do blokov
  + komentáre sú prepojené v blokoch avšak umiestnené sú mimo riadku, na ktorý sa odvolávajú, bolo by vhodné ich umiestniť vedľa textu
* história naposledy otvorených súborov
* obsahuje modul pre syntaktickú analýzu
* novšia LuaJit verzia – rýchle spracovanie menších súborov

#### Chýba:

* veľké súbory stále dosť pomalé na prácu
* pri otvorení napr. Analyzer.cpp nevyrobí správne bloky častí súboru, všetko brané ako samostatný riadok

### Práca s editorom

#### Implementované:

* zvýrazňovanie syntaxe až na úrovni blokov, teda je možné určiť grafické vlastnosti pre všeobecné prvky naprieč viacerým jazykom a gramatikám
* popis zvýrazňovania jednotlivých blokov, ktoré majú byť zvýraznené, je obsiahnutý v konfiguračnom súbore
* všetko v rámci editačného okna editora je možné presúvať
* existuje možnosť *„Edit plain text“* pre úpravu textu, vtedy je zobrazené v samostatnom okne všetko ako čistý text (tu prepínanie na dva módy, zabudovať priamo do editore)
* v súbore text\_item.cpp implementovaný pohyb medzi blokmi po stlačení kláves
* samostatné zoomovanie každého otvoreného súboru nehľadiac na tie ostatné
* presúvanie blokov v editore
* vyhľadávanie v texte zobrazí bloky, v ktorých sa text nachádza

#### Chýba:

* klávesové skratky veľmi chýbajú
  + nejaké už sú implementované (v Menu -> File sa dajú vidieť)
  + priamo na začiatku v súbore main\_window.cpp sa priraďujú skratky k akciám
* chýba možnosť Undo, Redo, Copy, Paste
  + len cez kontextovú ponuku cez pravé tlačítko je možná
* selekcia textu aj v rámci viacerých blokov
* malé možnosti vyhľadávania
  + chýba možnosť pri veľkých súboroch krokovania nájdených výskytov vyhľadávania
  + vyhľadáva len v aktuálnom súbore
  + lupa nie je klikateľné tlačítko
* naraz otvorená len jedna pracovná plocha (workspace)
  + triedu BlockGroup je v budúcnosti možné využiť na paralelné zobrazenie viacerých hierarchií (BlockGroup) na jednej scéne (DocScene) – viacero pracovných plôch
* konfiguračný súbor sa načíta len raz, pri spustení editora (sprevádza ho)
* základná štruktúra menu pre prácu s textom a options je zakomentovaná a neimplementovaná

### Programovanie v editore

* analýza zdrojového kódu je časovo náročnejšia a preto sa spúšťa iba v čase prechodu písania na ďalší riadok.
* funguje automatické odsadzovanie pri analýze
  + medzery a tabulátory sa nezobrazujú a realizujú sa len ako prázdny priestor pred príslušným blokom
  + prebytočné zbavenie sa medzier
* znak konca riadku je nahradený nastavením príznaku v bloku

### Komentáre

#### Implementované:

* posúvanie komentárov – plávajúce komentáre
* funguje zobrazenie jednoriadkových aj blokových komentárov ako samostatný blok
* šípka ku blokovému komentáru začína vždy na začiatku riadku
* šípka ku jednoriadkovému komentáru začína od konca daného riadku
* funguje CTRL + Ľavé tlačítko myší = vytvorí na danom mieste nový ale len bežný blok (podobný ako ten pre komentár), pričom ho prepojí šípkou z miestom kde sa nachádzal kurzor

#### Chýba:

* textové komentáre ako samostatné bloky, nie je možné napísať tvrdú medzeru
* šípka by mohla byť aj zmysluplnejšie ukazujúca na daný blokový komentár
* textové komentáre len ako bežné bloky, nie je možnosť rovno písať dokumentáciu ako bolo spomínané cez dokumentačné bloky
* chýba možnosť vytvárania dokumentačných blokov, ale funkčne je implementovaná

### Bloky

#### Implementované:

* možnosť presúvať bloky, alebo časti blokov
* pomocná čiara pri presune

#### Chýba:

* plávajúce bloky sa nedajú zmazať priamo, jedine postupným vymazaním ich obsahu
* chýba možnosť samostatne vytvárať bloky a prepájacie šípky
* šípka odkazuje len na jeden blok
* vždy možné presúvať len jeden blok naraz, chýba výber viacerých blokov
* chýba skrývanie blokov, nezobrazuje možnosť na skrytie blokov
* pri inicializácii sa nedeteguje prekrývanie viacerých blokov na jednom mieste
* pri vkladaní bloku rozostupovanie ostatných blokov

### Práca so súbormi, prílohami

#### Implementované:

* pridanie súboru ako prílohy v podobe bloku,
  + treba mať označený nejaký blok, aby bolo možné určiť časť súboru od ktorej sa priraďuje
* prílohy vkladá ako odkaz
* obrázky vie rovno zobraziť
* možnosť úpravy rozmerov obrázka
* ukladanie ako pôvodný súbor s komentármi, súbor bez komentárov, alebo ako PDF tlačiť (len printscreen v rámci ohraničenia pri tlači)
  + žiadne pokročilé prvky popisujúce obsah dokumentačných blokov ako bolo spomínané

### Syntaktický analyzátor

#### Implementované:

* analýza realizovaná v jazyku LUA za pomoci knižnice LPeg
  + možnosť rozširovania o ďalšie gramatiky (načítavajú sa z priečinka grammars)
  + výstupom je LUA tabuľka obsahujúca ďalšie tabuľky a tento systém tabuliek zodpovedá syntaktickému stromu
* vytváranie AST na strane jazyka LUA a jeho prenos do C++

### Gramatika

#### Implementované:

* základná gramatika default\_grammar.lua
  + rozloženie ľubovoľného textu na slová a riadky
  + popísané povinné konštanty, ktoré musia gramatiky obsahovať
  + funkcie na testovanie gramatík
* gramatika pre C, LUA a XML

### Literate programming

#### Implementované:

* možnosť vkladať ku kódu okrem klasických komentárov aj obrázky

#### Chýba:

* neukladajú sa pridané obrázky a iné formátovacie zmeny v dokumente
* ukladanie dokumentácie do RTF formátu

## Analýza použitých technológií

Pri implementácií budeme používať nástroje a technológie, ktoré používal predchádzajúci tím UFOPAK počas vývoja editora. Nosnými technológiami sú Qt SDK, Lua a využitie RTF, ktoré v skratke predstavíme ako aj dôvod, prečo sme sa rozhodli pokračovať v ich používaní.

### Qt

Qt je implementačný nástroj založený na jazyku C++. Je to technológia, pomocou ktorej je možné vyvíjať aplikácie pre rôzne platformy. Qt umožňuje vytvárať a jednoducho nasadzovať aplikácie pre počítače, mobilné telefóny, ale aj vnorené systémy (MP3prehrávače), bežiace pod operačnými systémami Windows, Linux, MAC OS, Symbian. Multiplatformovosť je práve jedna z rozhodujúcich výhod, kvôli ktorým je editor implementovaný pomocou tohto nástroja. Qt je v súčasnosti dostupné pod komerčnou ale aj GNU GPL v3.0 licenciou.

Nástroj Qt ponúka okrem množstva tried a knižníc pre tvorbu GUI aplikácií aj vlastné vývojové prostredie Qt Creator. Uvažované možnosti práce s nástrojom Qt boli nasledovné:

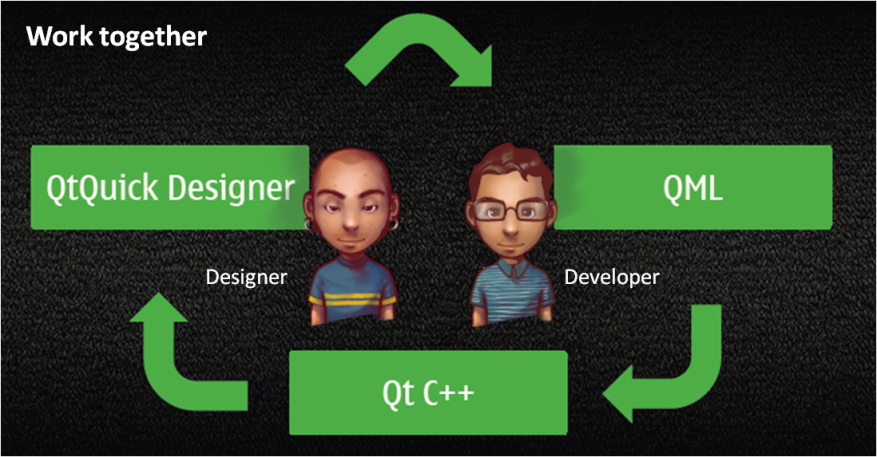
* Qt modul pre vývojové prostredie Eclipse
* Qt modul pre vývojové prostredie Visual Studio
* Integrované vývojové prostredie Qt Creator

Rozhodli sme sa pre použitie prostredia Qt Creator, keďže poskytuje samostatné vývojové prostredie, čiže pre naše potreby by malo byť ideálne, a taktiež integruje v sebe viacero novších technológií a prístupov, ktoré nám pomôžu pri vývoji. Napr. obsahuje novú technológiu Qt Quick.

### Qt Quick a jazyk QML

Qt Quick je nová technológia určená pre rýchle vytváranie jednoduchých a bohatých používateľských rozhraní aplikácií pre rôzne platformy. Qt Quick obsahuje jazyk QML, ktorý je navrhnutý vychádzajúc z jazykov HTML, CSS, JavaScript, pričom spája ich výhody.

S použitím technológie Qt Quick je možne aby dizajnér navrhol UI podľa vlastnej fantázie a vývojár len doplnil logiku aplikácie, čo prináša obrovskú výhodu keďže vývojár a dizajnér majú každý iný pohľad na svet a nie vždy bolo možne nájsť konsenzus pri vytváraní aplikácie.



Obrázok Pracovný cyklus s použitým Qt Quick

Qt Quick umožňuje vytvárať rôzne animácie, ktoré využívajú knižnicu OpenGL. Takisto umožňuje navrhnúť dizajn jednotlivých používateľských prvkov aplikácie ako napr. tlačítka v grafických editoroch Adobe Photoshop, Autodesk Maya, Gimp.

Jednoduchosť technológie Quick možno vidieť v rozdiele medzi definovaním jednoduchého tlačítka klasickým spôsobom cez actionscript a novým pomocou jazyka QML.

Actionscript: **MenuButton**.as

public class MenuButton extends MovieClip

public function MenuButton() {

this.x = 60;

this.addEventListener(MouseEvent.MOUSE\_DOWN, ClickBt);

}

function ClickBt(e:MouseEvent) {

trace(“clicked”);

}

}

QtQuick: **MenuButton**.qml

Item {

x:60;

MouseArea: {

anchors.fill: parent;

onClicked: print("clicked");

}

}

Použitie technológie Qt Quick by mal pre nás veľký význam keďže nám umožňuje navrhnúť UI pre TrollEdit podľa našej potreby, ktorý by bol zaujímavejší ako súčasné UI riešenia editorov. To nám dáva možnosť v tomto smere vytvoriť kvalitnejší produkt.

Príklad dizajnu navrhnutého s použitím technológie Qt Quick je možno vidieť v takých aplikáciách ako Skype, VLC Media Player atď.

### Jazyk Lua

Luaje rýchly procedurálny skriptovací jazyk, určený hlavne na vnorené používanie. Programátorské rozhranie (API) je navrhnuté tak, aby umožňovalo integráciu s programami napísanými v iných jazykoch (C, C++, Java, C#, . . . ) vrátane skriptovacích (Perl, Ruby).

Filozofiou jazyka Lua je jednoduchosť a rozšíriteľnosť. Obsahuje základnú funkcionalitu a mechanizmy ako definovať čokoľvek, čo považujeme za potrebné. Týmto spôsobom je možné získať aj schopnosti objektovo orientovaných (rozhrania, dedenie) alebo funkcionálnych jazykov. Lua je dynamicky typovaná a obsahuje niekoľko atomických dátových typov doplnených o jednu dátovú štruktúru – tabuľku. Tabuľka funguje ako asociatívne pole a jej pomocou je možné simulovať iné štruktúry (pole, množina, hash tabuľka, strom, atď.) a tiež objekty v zmysle OO paradigmy.

Lua patrí medzi najrýchlejšie skriptovacie jazyky. Je implementovaná v štandardnom ANSI (ISO) C, čo sa prejavuje na jej vysokej prenositeľnosti. Funguje pod všetkými známymi platformami. Výhodou Lua je jej veľkosť (aktuálna verzia Lua 5.1.4 má 860KB aj s dokumentáciou), vďaka ktorej nie je problém pripojiť ju celú k aplikácii, ktorá ju používa.

Lua je vyvíjaná pod voľnou licenciou (MIT) a môže byť používaná zdarma na akékoľvek (aj komerčné) účely. Lua sa dnes často používa pri skriptovaní počítačových hier, ale využívajú ju aj iné programy ako napríklad Skype, Wireshark, VLC media player atď.

### Knižnica LPeg

LPeg je knižnica jazyka Lua určená na hľadanie vzoriek v texte (pattern matching). Snaží sa odstrániť problémy spojené s používaním regulárnych výrazov, ktoré môžu byť pri komplikovanejších úlohách neprehľadné. Je postavená na gramatikách typu PEG (Parsing Expression Grammar) a formalizme podobnom bezkontextovým gramatikám. Na rozdiel od bežných gramatík, PEG nedeﬁnuje jazyk, ale algoritmus na jeho rozpoznanie. LPeg poskytuje dva moduly s rozličným spôsobom práce. V prvom module re (skratka z regex) sú vzory popisované reťazcami so syntaxou odvodenou z regulárnych výrazov. Druhý modul lpeg pracuje so vzormi ako s premennými vlastného dátového typu a obsahuje viac spôsobov na ich vytváranie a spájanie. Obidva moduly podporujú vyhľadávanie (vyjadrené priamo vzorom) rovnako ako zachytávanie reťazcov na pokročilej úrovni. Vybraný text je možné ukladať do tabuliek, ľubovoľne zamieňať a inak transformovať. LPeg používa tzv. limitovaný backtracking, vďaka ktorému je veľmi rýchly a efektívny.

### RTF

Rich Text Format(RTF) je metóda slúžiaca na zakódovanie formátovaného textu a obrázkov v textovom dokumente. RTF bolo vyvinuté pre prenášanie dokumentov medzi rôznymi platformami bez straty formátovania.

Každý RTF súbor obsahuje neformátovaný text, riadiace slová, riadiace symboly a grupy. Pre zjednodušenie prenositeľnosti štandardný RTF dokument obsahuje 7-bitové znaky. Riadiace slovo je špeciálne formátovaný príkaz, ktorý sa používa na označenie riadiaceho kódu a informácií používaných pri manažovaní zobrazenia dokumentov. Riadiace slovo má maximálnu dĺžku 32 znakov a jeho forma je:

\LetterSequence<Delimiter>

Každé riadiace slovo začína spätným lomítkom (backslash). Nasleduje postupnosť písmen (LetterSequence) tvorených malými písmenami v rozsahu „a“ až „z“ vrátane. RTF je citlivý na veľkosť písmen a každé riadiace slovo musí byť tvorené malými písmenami. Nakoniec nasleduje oddeľovač (Delimiter), ktorý označuje koniec riadiaceho slova.

# Špecifikácia požiadaviek

Keďže vychádzame z už existujúceho multiplatformového textového editora „TrollEdit“ obohateného o grafické prvky, popisujeme iba tie požiadavky na systém, ktoré chceme implementovať pripadne modifikovať v súčasnej verzií.

Hlavné ciele tohto projektu sú:

* Rozšíriť súčasnú - implementovanú funkcionalitu
* Modifikovať používateľské rozhranie - GUI
* Vytvoriť kvalitný produkt, ktorý bude úspešný a mohol by sa presadiť aj v praxi

## Funkcionálne požiadavky

Pre TrollEdit boli identifikovane funkcionálne požiadavky na základe dôkladnej analýzy predchádzajúceho riešenia a taktiež na základe podnetov od nášho vedúceho tímu, ktoré sú spísané v Tab.1.

Tab. Funkcionálne požiadavky

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Požiadavka** | **Charakteristika** | **Priorita** |
| F01 | Možnosť Undo/Redo | Možnosť vrátiť zmeny naspäť a opačne | vysoká |
| F02 | Podpora skratiek v editore (Shortcuts) | Možnosť spustiť funkcie programu pomocou klávesových skratiek | vysoká |
| F03 | Dopytovanie sa do Lua |  | vysoká |
| F04 | Podpora paralelizmu | Syntaktický strom by mal bežať na pozadí pod vlastným vláknom a program pod vlastným | vysoká |
| F05 | 2 módy písania | Prvý by bol klasický editor na úpravu kódu a po prepnutí by editor prešiel do druhého grafického módu. | vysoká |
| F06 | Nastavenie programu | Možnosť rozšírených nastavení priamo v editore | stredná |
| F07 | Podpora intellisense | Rozpoznávania bežných kľúčových slov programovacích jazykov, ale aj najčastejšie používané bloky kódu (napr. funkcie, cykly, podmienky) | stredná |
| F08 | Rozšírenie funkcionality | Možnosť rozširovať funkcionalitu pomocou zásuvných modulov | stredná |
| F09 | Vyhľadávanie | Určitý druh fulltextového vyhľadávania s prípadnou optimalizáciou pre najčastejšie vyhľadávané výrazy | stredná |
| F10 | Export súborov | Možnosť exportovania súboru do iných formátov (.csv, .doc) | nízka |
| F11 | Podpora sw metrík | Schopnosť detegovať určité ukazovatele v zdrojovom kóde ako index udržateľnosti, cyklomatická zložitosť, hodnoty Fan in a Fan out, ktoré by boli zobrazené v tabuľke, prípadne vizuálne v podobe grafov | nízka |
| **Legenda:**  Vysoká priorita – nevyhnutia funkcia systému, je základom funkcionality systému.  Stredná priorita – funkcia, ktorú možno implementovať neskôr netvorí základ funkcionality systému.  Nízka priorita – funkcionalita bude implementovaná v ďalších verziách | | | |

## Nefunkcionálne požiadavky

Pre TrollEdit boli identifikovane nasledujúce nefunkcionálne požiadavky pre správne zabezpečenie fungovania programu.

Tab. Nefunkcionálne požiadavky

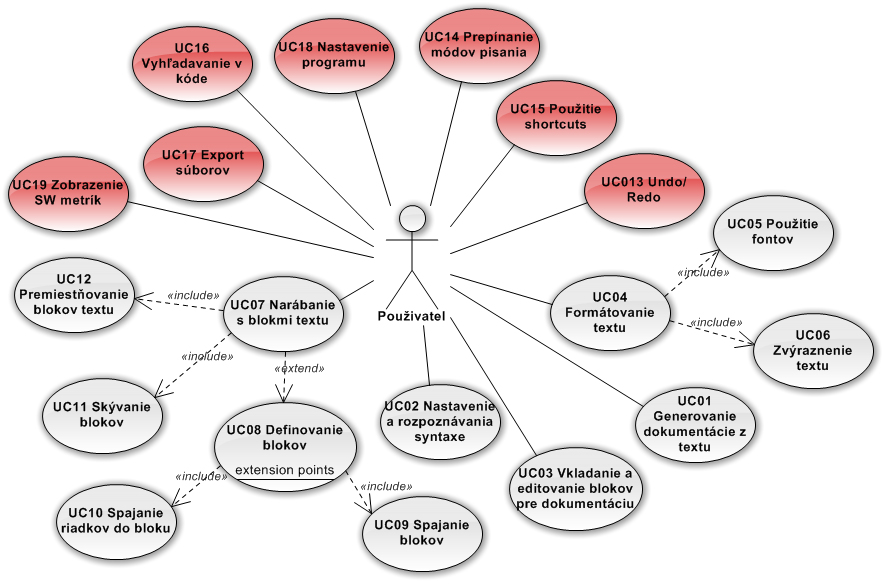
|  |  |  |
| --- | --- | --- |
| **ID** | **Požiadavka** | **Charakteristika** |
| N01 | Rýchlosť a spoľahlivosť | Zrýchlenie programu hlavne čo sa týka parsovania. Program by mal byť schopný pracovať aj na menej výkonnom hardvéri |
| N02 | Modulárnosť | Možnosť rozširovania jeho funkcií pomocou dodatočnej implementácie nových modulov. Tým pádom nie je v zásade nutné zasahovať do samotnej implementácie systému pri rozširovaní jeho funkcionality |
| N03 | Redesign používateľského rozhrania GUI | Musí byť jednoduché a prehľadné, pričom najčastejšie funkcie  systému by mali byť prístupné používateľovi bez náročného hľadania |

# Návrh riešenia

V tejto kapitole je popísaný návrh programu TrollEdit podľa požiadaviek definovaných v predchádzajúcej kapitole. Funkcionálne požiadavky sa premietnu do diagramu prípadov použitia a nefunkcionálne do architektúry systému.

## Diagram prípadov použitia

Na diagrame sú znázornené prípady použitia popisujúce funkcionalitu, ktorá je už implementovaná v programe TrollEdit a taktiež novu funkcionalitu, ktorú sme identifikovali na základe analýzy. Nové prípady použitia sú odlíšené od tých existujúcich červenou farbou.



Obr. Diagram prípadov použitia

Tab. Prípad použitia UC13 Undo/redo

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Undo/ redo** | | | | |
| **ID** | UC13 | | | | |
| **Opis** | Možnosť voľby undo/ redo nad vykonanými zmenami v zdrojovom kóde | | | **Priorita** | vysoká |
| **Vstupne podmienky** | História vykonaných zmien | | | | |
| **Výstupne podmienky** | - | | | | |
| **Participant** | Používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | vyberie možnosť undo/ redo v pop menu na zdrojovým kódom | | |
| 2. | Systém | urobí zmeny v kóde podľa histórie výkonných akcií | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
| **-** |  |  |  | | |
| **Poznámky** | - | | | | |

Tab. Prípad použitia UC14 Prepínanie módov písania

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Prepínanie módov písania** | | | | |
| **ID** | UC14 | | | | |
| **Opis** | Prvý mód pre klasický editor na úpravu kódu a po prepnutí by editor prešiel do druhého grafického módu | | | **Priorita** | vysoká |
| **Vstupne podmienky** | - | | | | |
| **Výstupne podmienky** | - | | | | |
| **Participant** | používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | V pop menu si zvolí možnosť prepnutia do druhého módu písania kódu | | |
| 2. | Systém | prepne úpravu kódu do grafického módu | | |
| 3. | Systém | rozšíri možnosti funkcionality pre graficky mód úpravy kódu | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
| **-** |  |  |  | | |
|  |  |  |  | | |
| **Poznámky** | - | | | | |

Tab. Prípad použitia UC14 Použitie shortcuts

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Použitie shortcuts** | | | | |
| **ID** | UC15 | | | | |
| **Opis** |  | | | **Priorita** | vysoká |
| **Vstupne podmienky** | - | | | | |
| **Výstupne podmienky** | - | | | | |
| **Participant** | používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. |  |  | | |
| 2. |  |  | | |
| 3. |  |  | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  |  |  |  | | |
|  |  |  |  | | |
| **Poznámky** | - | | | | |

Tab. Prípad použitia UC16 Vyhľadávanie v kóde

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Vyhľadávanie v kóde** | | | | |
| **ID** | UC16 | | | | |
| **Opis** | Vyhľadanie zvoleného výrazu v zdrojovom kóde | | | **Priorita** | stredná |
| **Vstupne podmienky** | - | | | | |
| **Výstupne podmienky** | Zobrazenie výsledku hľadaného výrazu | | | | |
| **Participant** | Používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | zadá hladný výraz do textboxu pre vyhľadávanie a potvrdí tlačidlom hľadať | | |
| 2. | Systém | vyhľadá zvolený výraz v aktuálnom zdrojovom kóde | | |
| 3. | Systém | zobrazí výsledky hľadaného výrazu | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 3.a | Systém | v prípade nenájdenia hľadaného výrazu zobrazí modálne okno s upozornením | | |
| **Poznámky** | - | | | | |

Tab. Prípad použitia UC17 Export súborov

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Export súborov** | | | | |
| **ID** | UC17 | | | | |
| **Opis** | Export súborov zdrojového kódu do iných formátov | | | **Priorita** | nízka |
| **Vstupne podmienky** | Parsovaný zdrojový kód | | | | |
| **Výstupne podmienky** | Vyexportovaný súbor | | | | |
| **Participant** | Používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | vyberie možnosť exportu súborov zo zdrojového kódu | | |
| 2. | Systém | ponúkne možností do akých formátov ma exportovať súbory | | |
| 3. | Použ. | vyberie formát súboru pre uloženie | | |
|  | 4. | Systém | uloží súbory vo zvolenom formáte | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
| **-** |  |  |  | | |
| **Poznámky** | formát pdf, doc. | | | | |

Tab. Prípad použitia UC18 Nastavenie programu

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Nastavenie programu** | | | | |
| **ID** | UC18 | | | | |
| **Opis** | Podrobne nastavenie možností programu | | | **Priorita** | stredná |
| **Vstupne podmienky** | - | | | | |
| **Výstupne podmienky** | Zmena nastavenia programu | | | | |
| **Participant** | Používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | si zvolí možnosť nastavenia programu z hlavného menu | | |
| 2. | Systém | zobrazí modálne okno s možnosťami nastavenia programu | | |
| 3. | Použ. | vykoná zmeny v nastaveniach a uloží zmeny | | |
|  | 4. | Systém | uloží vykonane zmeny a reštartuje program | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 4.a | Systém | v prípade nekorektného nastavenia oznámi používateľa varovaním oknom s popisom chyby | | |
| **Poznámky** | - | | | | |

Tab. Prípad použitia UC19 Zobrazenie sw metrík

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Názov** | **Zobrazenie sw metrík** | | | | |
| **ID** | UC19 | | | | |
| **Opis** | Zobrazenie sw metrík zdrojového kódu | | | **Priorita** | nízka |
| **Vstupne podmienky** | Zdrojový kód pre vygenerovanie metrík | | | | |
| **Výstupne podmienky** | Zobrazenie výsledkov metrík vo forme grafov | | | | |
| **Participant** | Používateľ (použ.) | | | | |
| **Základná postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
|  | 1. | Použ. | si zvolí možnosť zobraziť sw metriky z menu | | |
| 2. | Systém | vygeneruje metriky zo zdrojového kódu a zobrazí výsledky vo forme grafov | | |
| **Alternatívna postupnosť** | **Krok** | **Rola** | **Činnosť** | | |
| **-** |  |  |  | | |
| **Poznámky** | - | | | | |

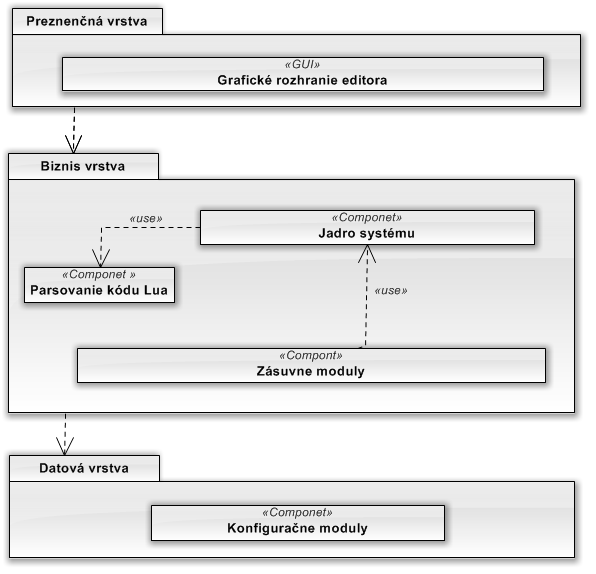
## Architektúra programu

Architektúra programu bude postavená na klasickom trojvrstvom princípe, t.j. rozdelená na prezentačnú, biznis a dátovú vrstvu vid. .

V prezentačnej vrstve budú implementované triedy pre grafické rozhranie editora od hlavného menu až po nápovedu. Prezentačná vrstva bude komunikovať s biznis vrstvou, v ktorej bude spracovávaná aplikačná logika programu.

Biznis vrstva sa bude skladať z troch komponentov. Jeden pre jadro systému kde bude implementovaná základná funkcionalita programu. Druhý pre parsovanie zdrojového kódu, kde sa bude vytvárať AST strom v skriptovacom jazyku Lua. A tretí pre pridávanie novej funkcionality, ktorá nenaruší základnú funkcionalitu jadra programu.

V dátovej vrstve budú dáta ktoré si bude program ukladať ako nastavenie programu dočasnú históriu zmien nad zdrojovým kódom a údaje o projekte.



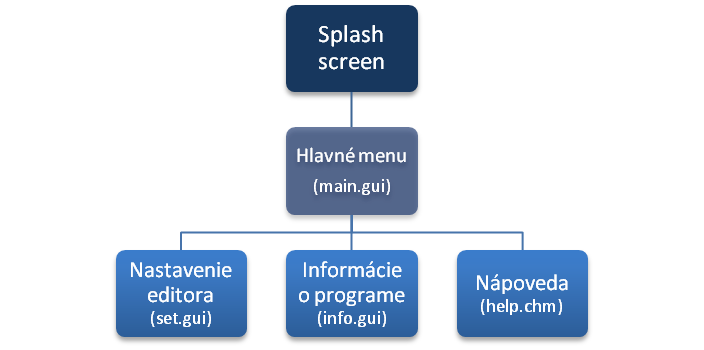
Obr. Architektúra programu

//Refactoring použiť nejake vzory ktore by sa nam tam hodily

//Nejaky diagram tried

## Návrh GUI

Na základe analýzy sme identifikovali 5 okien, ktoré budú v programe implementované. Tieto okná budú mať hierarchicky význam, t.j. okno na vyššej úrovni môže volať iba okná na nižšej úrovni, nie však opačne.



Obr. Hierarchické rozdelenie okien programu

#### Hierarchia:

* Splash screen
  + úvodné modálne okno, ktoré sa zobrazí vždy pri spustení TrollEditu
  + odstraňuje problémy studeného štartu
* Hlavne menu
  + nemodálne okno, ktoré slúži ako hlavné menu programu
  + z tohto okna je možné volať iné okna
* Nastavenie editora
  + modálne okno, ktoré slúži pre detailné nastavenie editora
* Informácie o  programe
  + modálne okno, ktoré zobrazuje informácie o programe ako popis programu, dátum vytvorenia, verzia programu.
* Nápoveda
  + nemodálne okno pre zobrazenie nápovedy programu



Obr. Splash screen programu



Obr. Predbežný návrh hlavného menu programu

# Implementácia prototypu

Táto kapitola popisuje implementáciu systému t.j. prevedenie návrhu do výsledného funkčného kódu.

## Popis prototypu

# Testovanie

V rámci tejto časti budú popísať spôsoby testovania a jednotlivé navrhnuté akceptačné testy.

## Akceptačné testy pre overenie funkcionality

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Názov** | | **Použitie Undo/ redo** | | | | |
| **Rozhranie** | | hlavne menu | | | **ID testu** | 01-13 |
| **Účel testu** | |  | | | **ID UC** | 13 |
| **Vstupne podmienky** | |  | | | | |
| **Výstupné podmienky** | |  | | | | |
| **Krok** | **Akcia** | | **Očakávaná reakcia** | **Skutočná reakcia** | | |
| 1. |  | |  |  | | |
| **Úroveň splnenia testu** | | splnil očakávanie | | | | |
| **Poznámka** | | - | | | | |

**Inkrementálny a iteratívny vývoj**

* Analýza, špecifikácia požiadaviek a hrubý návrh (zimný semester)
  + *Úvod - o čom je tento dokument, ciele, ohraničenia.*
  + *Analýza problému a špecifikácia riešenia*(pre tvorbu softvérového systému typicky zahŕňa tieto časti: Kontext systému, Špecifikácia funkcií systému (určí sa aj priorita pre jednotlivé funkcie), Špecifikácia údajov v systéme, Špecifikácia správania systému)
  + *Hrubý návrh riešenia*
  + *Ďalšie požiadavky a ohraničenia*
* Prototyp (zimný semester)
  + Cieľ prototypovania, dosiahnuté výsledky
  + Podľa dohody s pedagógom, odporúča sa používateľská príručka (pre celý systém)
* Produkt a dokumentácia k produktu (letný semester)
  + Stanoví sa podľa povahy projektu. Štandardne zahŕňa tieto časti:
    - *Používateľská príručka*
    - *Systémová príručka* (spolu s návodom na inštaláciu)
* Návrh, implementácia a overenie riešenia (letný semester)
  + *Zapracovanie nedostatkov špecifikácie a hrubého návrhu*
  + *Návrh systému*(pre tvorbu softvérového systému typicky zahŕňa tieto časti: Architektúra systému, Fyzický model údajov systému, Návrh algoritmov spracovania)
  + *Ohraničenia, zmeny špecifikácie, priority riešenia*
  + *Výber implementačného jazyka a prostredia*
  + *Opis realizácie (implementácie jednotlivých modulov, napr. zaujímavé veci, optimalizácia, doplnenia oproti návrhu,...)*
  + *Overenie výsledku (určenie spôsobu overenia výsledku, postup, testovacie údaje, ak sa zmenili oproti návrhu)*
  + *Záznam o používaní systému*
  + *Čo sme nestihli*
  + *Čo sme sa naučili*