Fakulta matematiky, fyziky a informatiky Univerzity Komenského

Návrh

na SVOSA

Obsah

1 Úvod

Návrh je dokument opisujúci ako bude presnejšie fungovať náš software SVOSA. Tento dokument obsahuje predošlé dokumenty: Diagramy, Konceptuálna analýza, Analýza technológií, návrh rozhrania a popis dátového modelu spolu s podrobnou špecifikáciou komponentov. Tento dokument je určený pre skupinu programátorov, tak by došla z hľadiska funkcionality a vlastností softvéru k rovnakému riešeniu.

1.1 Legenda

SA - Spektrálny analyzátor.

SVOSA - Softvér pre vzdialené ovládanie spektrálneho analyzátora.

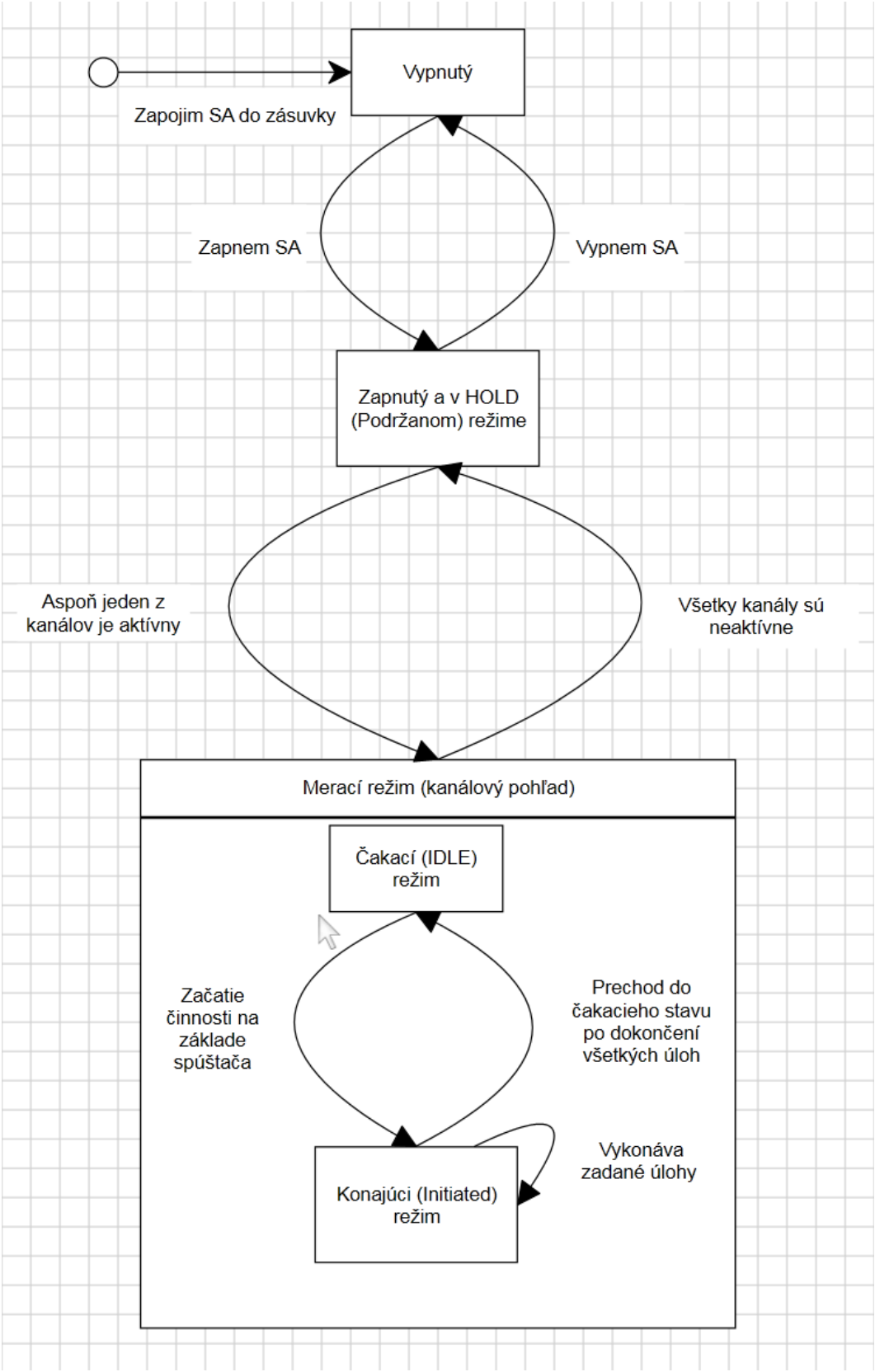
2 Konceptuálna analýza

Tento dokument slúži na logičke odvodenie požiadaviek z platného katalógu požiadaviek, súvislosti a prezentáciu základných dát. Obsahuje diagramy to entitno-relačný diagram, stavové diagramy, a use-case diagram. Na konci tohto dokumentu je legenda s vysvetlivkami pojmov a popis užívateľského rozhrania.

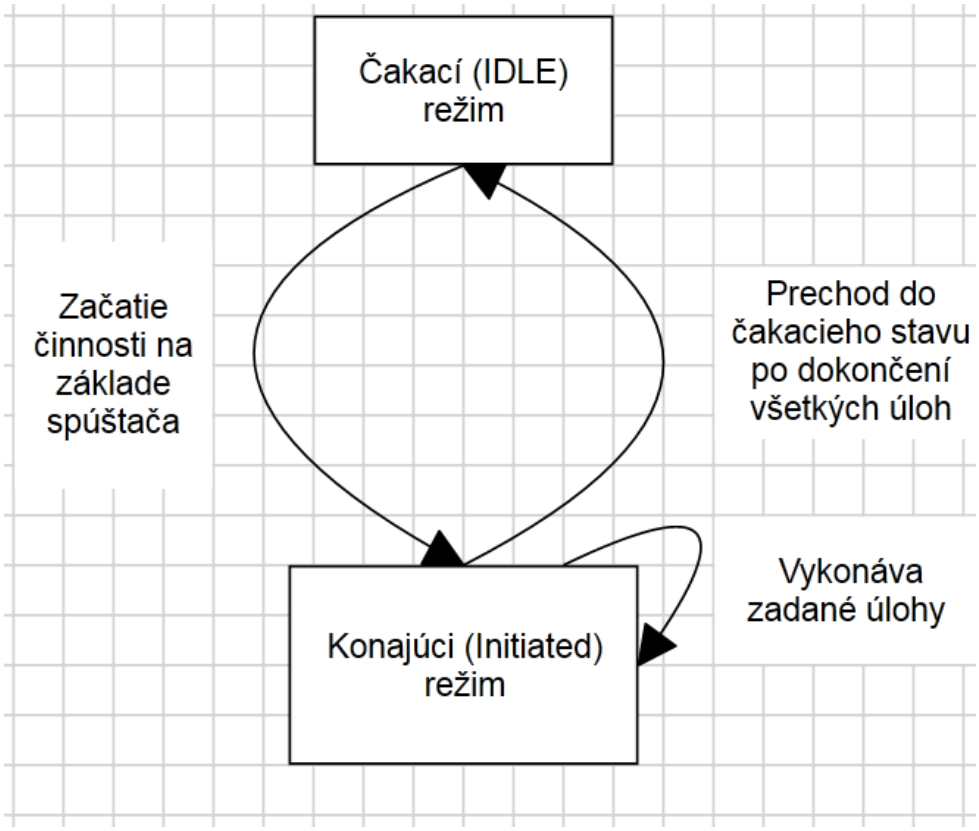
2.1 Stavové diagramy

Diagramy vysvetľujú ako z pohľadu SA funguje proces experimentu, teda jeho hlavnej činnosti.

Bližšie detaily sú vysvetlené v dokumentácii [2] na 81 strane.



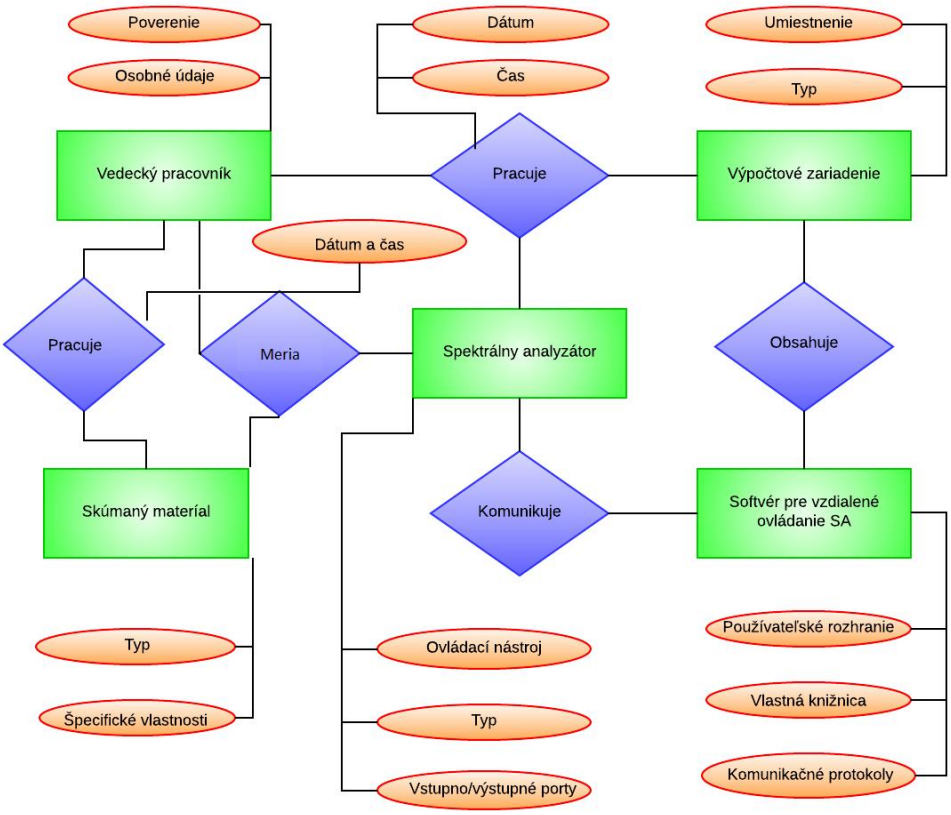
Obrázok 2.1 : Vysvetľuje systémový pohľad na proces experimentu z pohľadu SA.



Obrázok 2.2: Vysvetľuje kanálový pohľad na proces experimentu z pohľadu SA. Samotný SA vie vykonávať experiment vo viacerých kanáloch, čo zvyšuje rýchlosť experimentu, lebo vie robiť viac veci naraz.

2.2 Entitno-relačný diagram

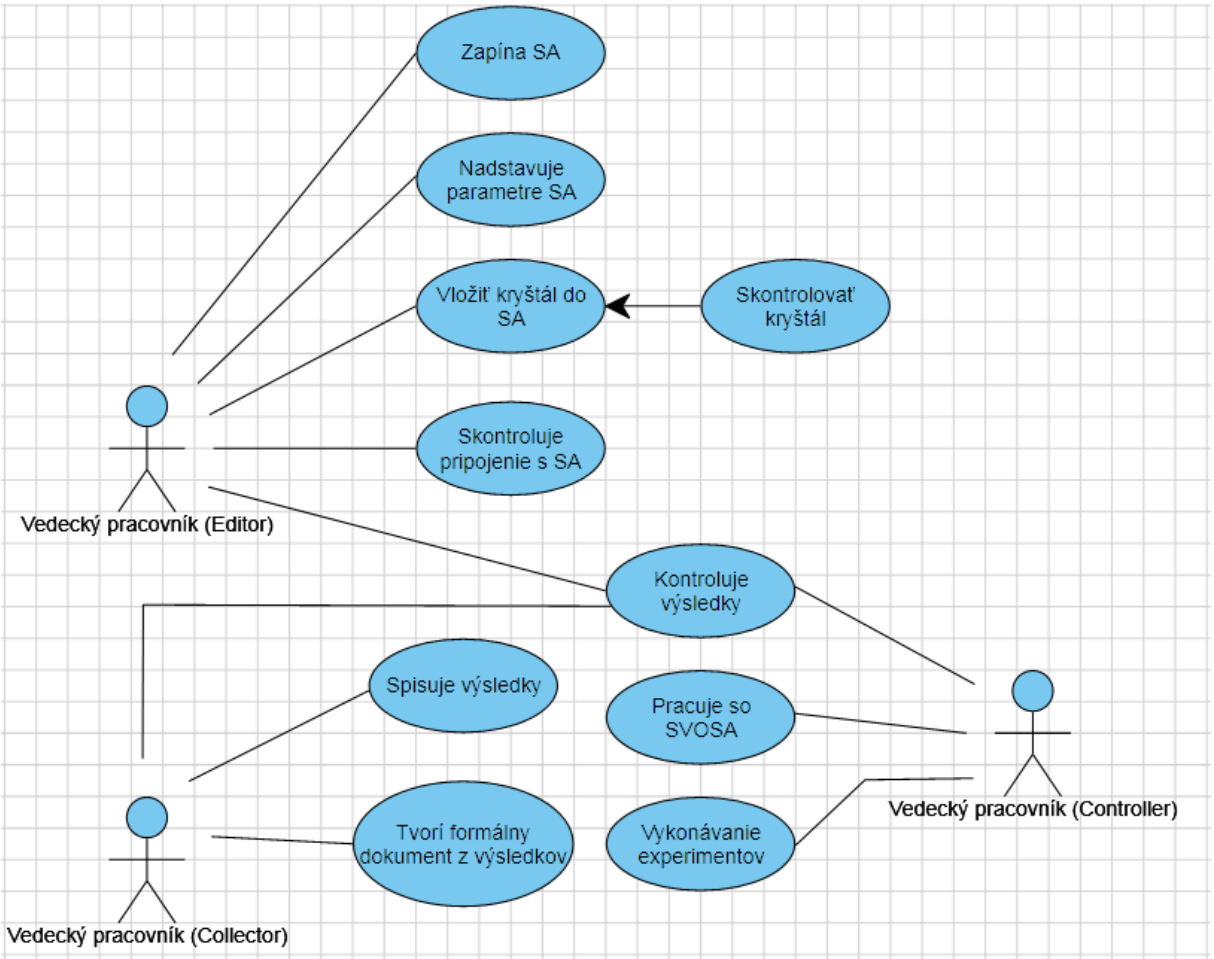
Diagram znázorňuje jednotlivé entity vstupujúce do experimentu a ich vzťahy (relácie) medzi sebou v priebehu tohto experimentu.



Obrázok 2.3: Znázorňuje jednotlivé vzťahy medzi entitami a ich atribúty v procese experimentu.

2.3 Use-case diagram

Use-case diagram je najjednoduchší spôsob reprezentácie vzťahov medzi užívateľom systému popísaný stavmi použitia. Tento diagram popisuje, ako užívateľ reaguje počas experimentu so SA a so softvérom SVOSA.



Obrázok 2.4: Popisuje interakcie užívateľov so systémom. Jednotlivé role môže vykonávať aj jeden človek, ale aj skupina.

2.4 Užívateľské rozhranie

Užívateľské rozhranie bude spočívať vo forme príkazového riadku, do ktorého používateľ bude volať procedúry a metódy SVOSA. Užívateľ potom môže dodefinovať ďalšie argumenty k programu. Ak uvedenie 1 argument, tak ten je definovaný ako dvojica frekvencie a impedancie. Ak uvedie ešte jeden parameter, bude to meno súboru, do ktorého sa uložia dáta do prehľadnej tabuľky (do relatívneho priečinka odkiaľ bol súbor spustený). Ostatné argumenty budú ignorované.

3 Analýza technológií

3.1 Úvod

Tento dokument má slúžiť na popis technológii, ktoré sú potrebné na plnú funkcionalitu ISVOS-y. Taktiež opisuje komponenty, ktoré budú v našom systéme a taktiež zobrazuje dátový model a triedny diagram.

Ako bolo stanovené v dokumente "Katalóg požiadaviek", v sekcii "2.1.1 Používateľské rozhrania" projekt bude písaný v programovacom jazyku Python.

3.2 Používané technológie

3.2.1 Možnosti technológií pripojenia SVOSA k SA:

1. LAN - (Local Area Network) systém vzdialeného ovládania poskytuje dve metódy :

    - SICL-LAN – v ovládacom systéme používajúcom SICL-LAN server, komunikácia medzi vonkajším ovládačom (klientom) a SA (serverom) je uskutočnená použitím SICL-LAN protokolu. Samotná komunikácia je uskutočnená pomocou SICL (Standard Instrument Control Library). Užívateľ môže ovládať SA programovaním pomocou SICL alebo VISA v jazyku C pod operačným systémom Linux, alebo Visual C++, Visual Basic a VEE pod operačným systémom Windows.

    - Telnet – v ovládacom systéme cez telnet server, komunikácia je uskutočňovaná cez pripojenie medzi zásuvnými modulmi poskytovanými procesmi vonkajšieho ovládača a SA na nadviazanie sieťového spojenia medzi nimi. Ovládací systém cez telnet môže komunikovať dvoma portami a to:

        - port 23 – používa sa na dialógové ovládanie používajúce telnet (užívateľské rozhranie pre TELNET protokol)

        - port 5025 – používa sa na ovládanie z programu

2. GBIP – (General Purpose Interface Bus) je štandardné rozhranie pre pripojenie počítačov a periférnych zariadení, ktoré podporuje nasledujúce medzinárodné štandardy : IEEE 488.1,IEC-625, IEEE 488.2, a JIS-C1901 . GPIB rozhranie umožňuje ovládať Agilent SA z externého počítača. Počítač odosiela príkazy a pokyny na SA a prijíma dáta odoslané z SA cez GPIB .

Výber technológie pripojenia SVOSA k SA

Na základe špecifikácie požiadaviek sme sa rozhodli uprednostniť LAN technológiu, keďže má byť SA ovládaný cez lokálnu sieť, s použitím metódy SICL-LAN pretože ovládací počítač má bežať pod operačným systémom Linux.

3.2.2 Hardwarové technológie

1. Náš systém bude používať technológiu TCP/IP na spojenie sa s koncovým zariadením. Očakáva sa že to bude väčšinou notebook alebo PC. Prípadné rozšírenie je zapojenie zariadenia do sieťového prvku (napríklad rozbočovač).

3.2.3 Softwarové technológie

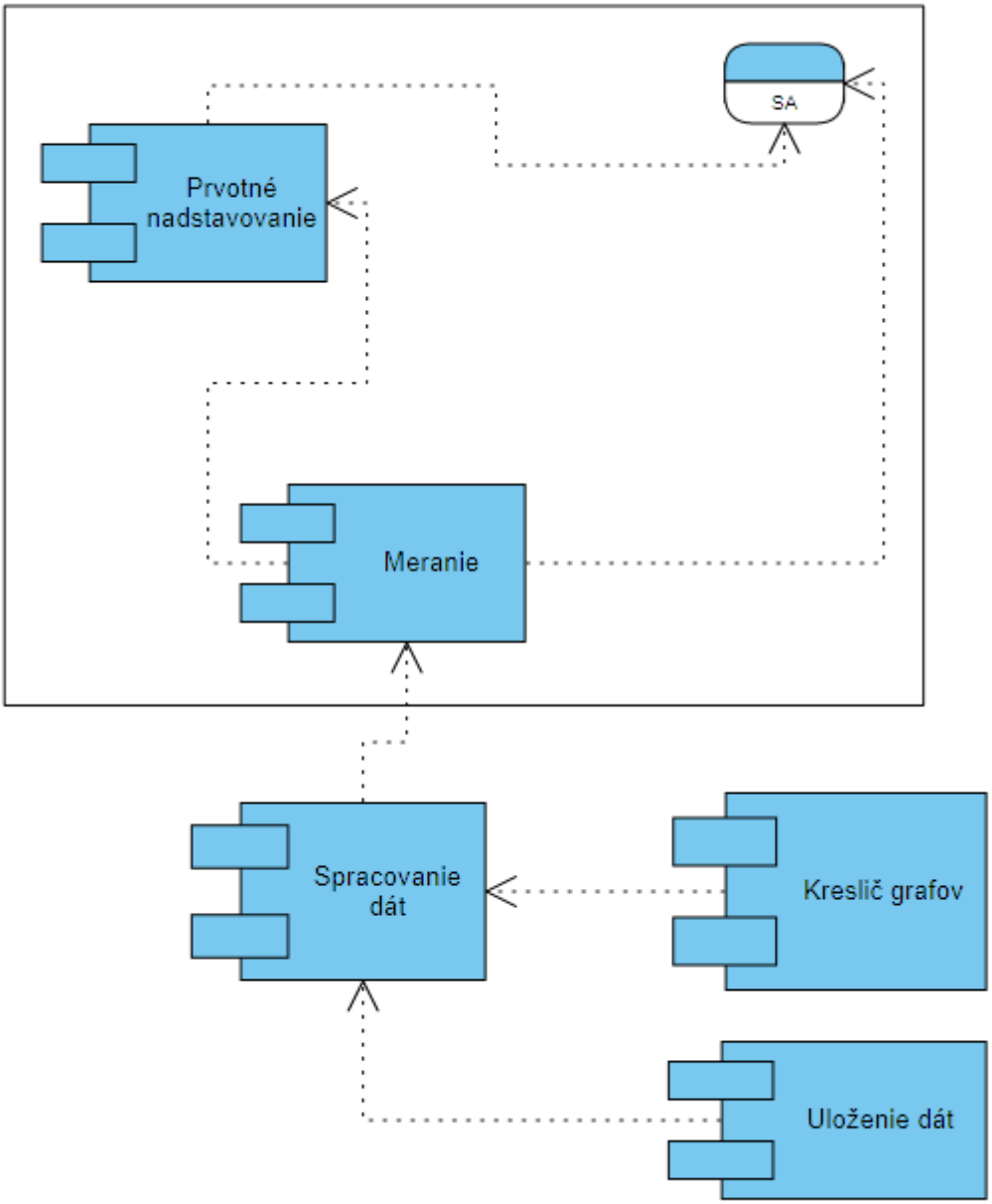
Náš systém bude používať rôzne knižnice pre dosiahnutie konkrétnych požiadaviek

1. Na vykresľovanie grafov budeme používať knižnicu "Gnuplot", ktorá dokáže vykresľovať rôzne typy grafov. Všeobecnejšie sa dá používa aj ako nástoj v Linuxe na kreslenie grafov cez príkazový riadok.

2. Taktiež na nízko úrovňovú sieťovú komunikáciu s SA budeme potrebovať aj knižnicu menom "socket". Budeme vedieť pomocou nej sa pripojiť na konkrétnu IP adresu na konkrétny port nášho SA.

3. Knižnica "math" budeme používať na rôzne výpočty matematických vzorcov a volanie matematických funkcií.

4 Komponentný diagram



Obrázok 4.1

4.1 Komponent prvotné nastavovanie

Komponent slúži na nastavenie všetkých potrebných parametrov potrebných na napojenia sa pomocou siete na SA. Okrem iného sa v ňom nastavujú dôležité konštanty ako napríklad adresa zariadenia na siete či príslušný port, ktoré užívateľ ďalej nemôže meniť. Tu sa budú dať aj nastaviť ostatné parametre ako bolo už definované v katalógu požiadaviek.

4.2 Komponent meranie

Komponent slúži na posielanie pokynov na meranie a ich spätné prijímanie v nastavenom formáte. Dáta sa ukladajú do internej štruktúry. Spracovanie dát zo štruktúry je už spracúvaný iným systémom, ktorý dátam rozumie.

4.3 Komponent spracovanie dát

Komponent prečíta dáta z internej štruktúry a preloží do ďalších potrebných štruktúr a počítajú sa potrebné veličiny z dodaných dát pre ich ďalšie využitie.

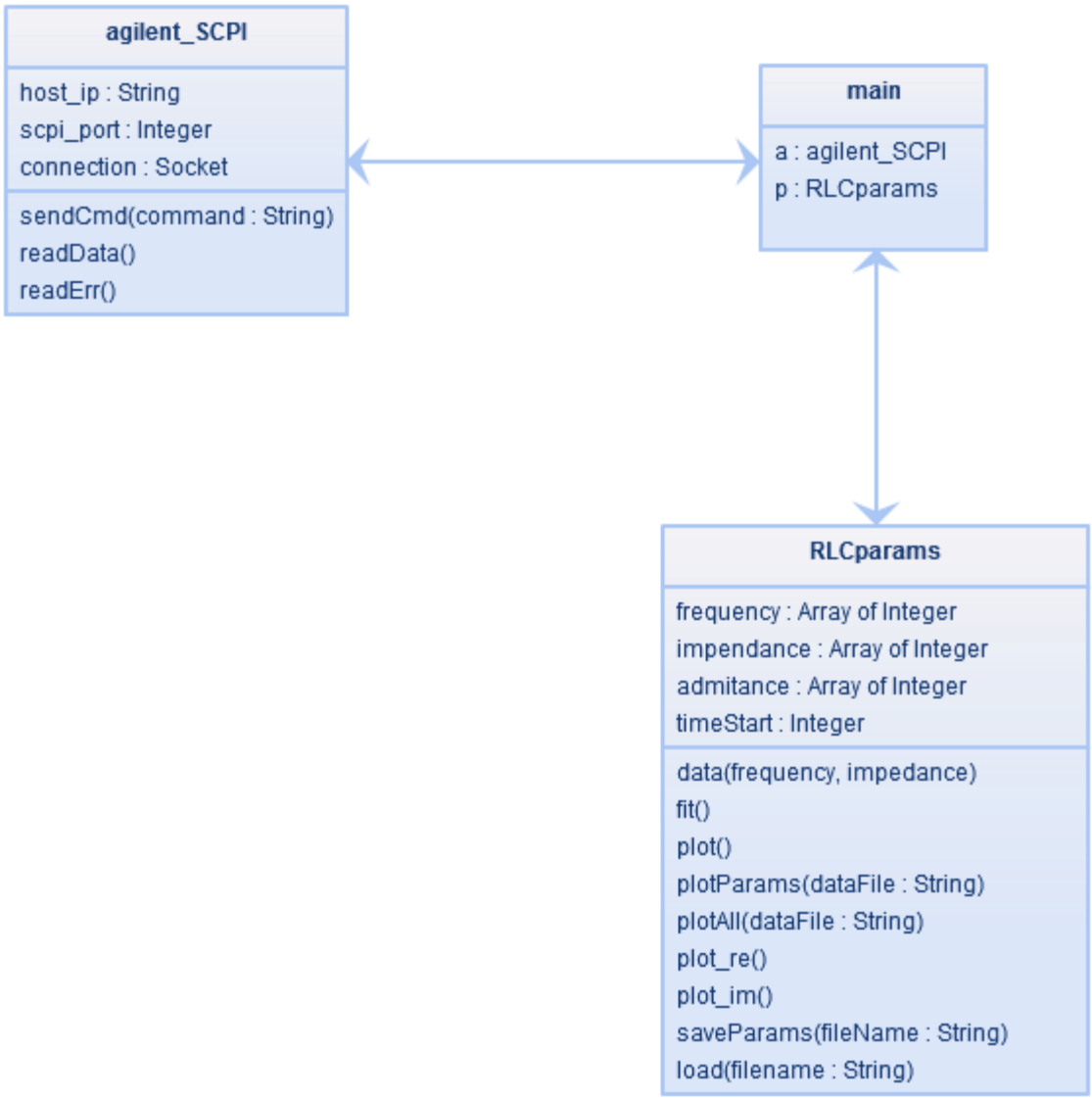
4.4 Komponent uloženie dát

Dáta sa ukladajú do prehľadnej tabuľky, ak sa užívateľ rozhodne inak sa dáta nebudú ukladať.

4.5 Komponent kreslič grafov

Z poskytnutých spracovaných a vyrátaných dát a veličín dokáže tento komponent zobraziť potrebné grafy.

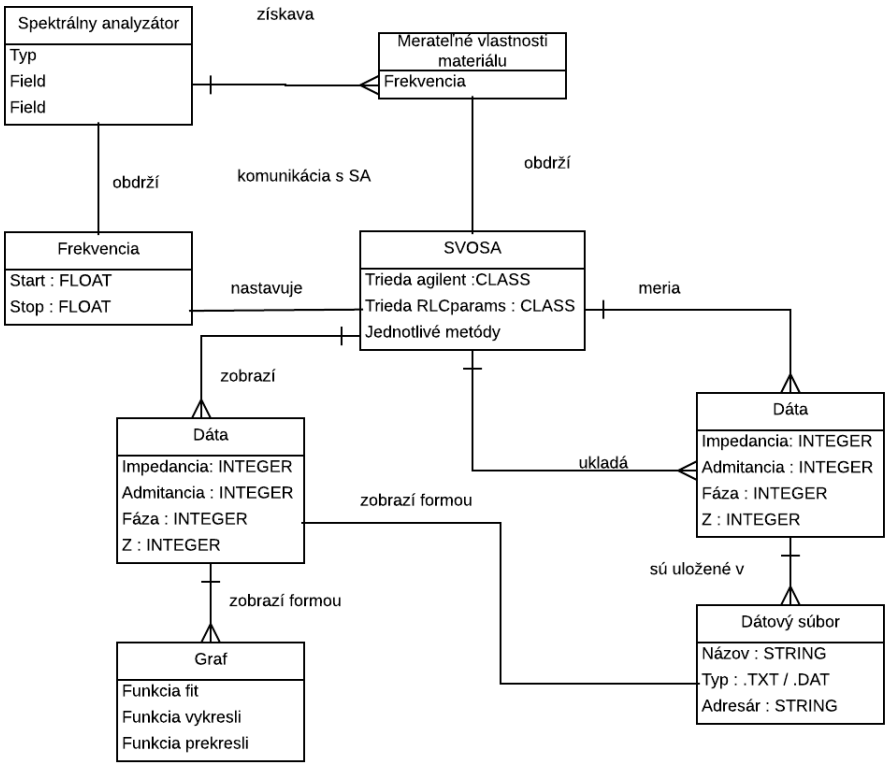
5 Triedny diagram



Obrázok 5.1

Triedny diagram znázorňuje štruktúru projektu, popisuje jeho triedy a ich vzájomné prepojenie. Hlavný program main spravuje a inicializuje triedy agilent_SCPI a RLCparams. SVOSA pomocou triedy agilent_SCPI nadviaže spojenie zo SA, a následne pomocou tejto triedy aj príjme dáta, ktoré po úspešnom prijatí zanalyzuje a graficky znázorní pomocou triedy RLCparams.

6 Dátový model zobrazený pomocou entitno-relačného diagramu



Obrázok 6.1

Podrobnejší entitno-relačný diagram znázorňujúci dátový model Informačného systému.

SVOSA pracuje a a nalyzuje získané dáta od SA a tie následne dokáže graficky znázorňovať, alebo ich dokáže ukladať (exportovať) do textového, alebo dátového súboru s ktorým môžu pracovníci ďalej pracovať.

7 Testovacie scenáre

Pri používaní spektrálneho analyzátora môže dôjsť k rôznym situáciám. Tomuto sa venuje kapitola testovacie scenáre, ktorá popisuje rôzne prípady, aké môžu nastať počas merania na spektrálnom analyzátore pri použití SVOSA. Pri spustení programu sa najskôr inicializujú triedy (vytvoria sa inštancie objektov) Agilent a RLCparams aby sa s nimi dalo ďalej pracovať, nachádzajúce sa v samostatných jednotkách (agilent.py, respektíve rlcparams.py). Nasleduje popis tried a priebeh testovania v rámci behu kódu:

7.1 rlcparams

Triede RLCparams sa veľmi podrobne venovať nebudem. Súvisí s hlbšou matematikou a fyzikou. Nachádza sa tu konštruktor triedy, kde si inicializujeme atribúty triedy na nulové(prázdne hodnoty). Atribúty frequency, impedance, admitance, vyrez, vyref, vyrezimag, vyref_fit, yre_fit, yim_fit sú štruktúrované ako polia (lists) ďalej tu sú atribúty: R1, C0, C1, L1, fResonance, fResonanceFit, gnuplot, gnuplot1 , gnuplot2 (objekty Gnuplot.Gnuplot()) a timeStart. Trieda má metódy : data(self, frequency, impedance) - nastavenie dát; fit(self) - výpočet rezonancie pomocou fitovacej funkcie; metóda plot(self) - metóda na grafické zobrazenie celej krivky a výrezu a metódy plotParams(self, dataFile); plotAll(self, dataFile); plot_re(self) ; plot_im(self); saveParams(self, fileName = '') a load(filename). Táto trieda je plne funkčná a preto ju nebude treba špeciálne testovať.

7.2 agilent

V triede agilent má 3 základné atribúty = gnuplot = None a 2 atribúty typu slovník: data = {} a frequency = {}. Ako prvú vec si pri inicializácii vytvoríme objekt agilent_scpi, ktorý inicializujeme. Táto trieda sa nachádza v jednotke(unite) agilent_scpi.py a nach=adzaj=u sa v nej metódy určené priamo pre komunikáciu so SA : metóda na odosielanie príkazov(povelov): send_cmd, metóda na čítanie prichádzajúcich dát zo SA - read_data a metóda na zber chybových správa (errorov) - read_err .Trieda má 3 atribúty: _host_ip - tu sa nastavuje a ukladá ip adresa prístroja ('192.168.1.102'), atribút _scpi_port - kde sa nastavuje komunikačný port (5025) a atribút _connection: kde sa pomocou knižnice socket nastaví vhodný protokol, kde sa zadá ip adresa a port prístroja. Pri inicializovaní triedy si zadefinujeme v slovníku data pojmy: ['frequency'], ['Z'], ['impedance'], ['admitance'] a ['phase']. Ďalej ideme nastavovať SA prvým príkazom pomocou send_cmd a tým je príkaz ':syst:pres' (preset), ktorý nastaví SA na pôvodné továrenské nastavenie, kde je nastavený aj východiskový kanál, ako kanál 1. Ďalej nastavujeme formát dát z viacerých možných budeme v testoch používať formát typu Smythovho diagramu $(R+jX)$ pomocou commandu ':calc1:form smit'. Potom nastavujeme východiskový režim na nepretržitý pomocou volania príkazu ':init1:cont on' a následne sa zavolá

abort pomocou príkazu ':abor' , ktorý zastaví merania a zmení spúšťacie poradie pre všetky kanály do u pokojného stavu. Ďalej nastavíme počet meracích bodov (hustota zobrazovania nameraných údajov - sweeping point) v ukážkovom teste na maximálnu možnú hodnotu pre náš typ SA, pre väčšiu presnosť merania a tou je 1601 bodov (minimum sú 2) príkazom ':sens1:swe:poin 1601'. V ďalšom kroku nastavíme formát pre prenos dát na ASCii formát vďaka príkazu ':form:data asc' a povieme analyzátoru, aby počkal na vykonanie všetkých príkazov, ktoré boli doposiaľ analyzátoru zaslané. Nasleduje opýtanie sa na počiatočnú hodnotu sweep rangu - krivky(grafu)zobrazujúcej výsledky merania (pýtame sa na frekvenciu, čiže výslednú hodnotu dostávame v Hertzoch) pomocou commandu ':sens1:freq:star?' a načítame túto odpoveď do slovníka frequency ako jeho zložku ['start'] ako reálne číslo typu float pomocou funkcie read_data(). Takisto sa opýtame na koncovú hodnotu frekvencie príkazom ':sens1:freq:stop?' a toto si zapíšeme do slovníka frequency a jeho zložky ['stop'] podobným spôsobom ako pri start. Ďalej nasleduje samotná inicializácia Gnuplotu (triedy pre vizuálne zobrazenie pomocou grafov - Tejto triede sa budem venovať neskôr).

metóda set_frequency(self, start='', stop='', center='', span=''):

Najprv nastavíme SA vstupné parametre vďaka funkcii send_cmd a to tak, že pre start použijeme príkaz ':sens1:freq:star ' + start - (teda k nemu rovno "prilepíme" samotnú číselnú hodnotu pre štartovaciu rekvenciu zo vstupnej premennej start). Nastavenie končiacej frekvencie: ':sens1:freq:stop ' + stop , nastavenie stredovej hodnoty pre sweep range (meranie) pomocou príkazu ':sens1:freq:cent ' + center , a nakoniec nastavenia rozpätia pre sweep range vďaka príkazu ':sens1:freq:span ' + span . Nasleduje príkaz '*wai' , teda povieme analyzátoru, aby počkal až, kým sa nevykonajú všetky zadané príkazy a vypýtame si od SA nové štartovacie a koncové hodnoty frekvencií a uložíme si ich do slovníka data :

self.send_cmd(':sens1:freq:start?') , self.frequency['start'] = float(self.read_data())

self.send_cmd(':sens1:freq:stop?'), self.frequency['stop'] = float(self.read_data())

metóda measure(self):

Metóda pre samotné meranie. V prvom kroku si vyprázdnime celý slovník data (vyprázdnime jeho položky ['frequency'],['Z'],['impedance'],['admitance'] a ['phase']). V ďalšom kroku nastavíme režim SA na hold, teda vypnutie kontinuálneho režimu príkazom: ':init1:cont off' . Počkáme kým sa tak udeje (príkaz *wai) a pre aktívnu stopu kanála 1 prečítame formátované dátové polia vo formáte, aký bol predtým nastavený, teda v našom prípade to je ASCI, pomocou príkazu ':calc1:data:fdat?' a tieto dáta ukladáme do slovníka data a budeme ich ďalej spracovávať. Opäť sa spýtame na začiatočné a koncové frekvencie a hodnoty si aktualizujeme v slovníku frequency, počkáme a nastavíme režim SA naspäť do kontinuálneho meracieho režimu ':init1:cont on' .

Nasleduje spracovávanie dát. Najprv ich rozdelíme pomocou metódy split, oddeľovacou čiarkou na pole stringov. vypočítavame hodnotu gamma pre každú rozdelenú hodnotu z dát = complex(float(data[i]), float(data[i+1])) ďalej pridávame jednotlivé frekvencie, získavame hodnoty čísla Z , impedanciu, admitanciu a fázu a všetko si to ukladáme do jednotlivých zložiek slovníka data.

metóda plot_data(self):

Metóda triedy agilent zobrazujúca dáta pomocou Gnuplot u do formy grafu.

metóda save_data(self, fileName=''):

Pokiaľ nie je zadaný názov súboru, do ktorého chceme dáta ukladať, metóda neurobí nič, inak nám uloží dáta do nového súboru, ktorý pomenuje podľa vstupného parametra fileName a to nasledovne : vynechá dva riadky, zapíše # a 140krát pomlčku (oddeľovač), do ďalšieho riadku zapíše - '# frequency\t\t| Z.real\t\t| Z.imag\t\t| |Z|\t\t\t| phase\t\t\t| admitance\n' , nasleduje ďalší oddeľujúci riadok a potom cyklicky zapisujeme ako do tabuľky všetky hodnoty slovníka data v poradí frekvencia, Z - imaginárna, Z - reálna časť, impedancia, fáza a admitancia pekne po riadkoch až do konca a zavrieme súbor - datafile.close().

7.3 Popis validného testovacieho scenára

Meranie sa spúšťa v jednotke(unite) main.py. Vytvoríme objekt a = agilent a p = rlcparams. Nastavíme frekvenciu pomocou metódy objektu a (funkcie) set_frequency s hodnotami pre center = '8000000' a pre span = '500000' . Voláme metódu measure() = meraj. A po vykonaní celého merania zavoláme metódu plot_data() na zobrazenie výsledku merania. Do premennej m si uložíme maximálnu hodnotu admitancie - vytiahneme ju príkazom max(a.data['admitance']) zo slovníka data. Zapamätáme si index maximálnej admitancie do premennej maxIndex a nastavíme novú stredovú (center) frekvenciu na hodnotu frekvencie zo zložky frequency v slovníku data pod indexom maxIndex - maximálnej admitancie : a.set_frequency(center = '%f' % a.data['frequency'][maxIndex]) . Nastavíme aj nové rozpätie merania (span) na hodnotu 2000000 : a.set_frequency(span = '2000000') . Opäť voláme metódu measure() = meraj a zobrazíme výsledné dáta po jej ukončení.

Nasleduje podobný proces, teda nájdeme si znova najväčšiu admitanciu z nového merania a zapamätáme si jej index a prednastavíme stredovú hodnotu merania na hodnotu frekvencie pod týmto indexom. Nastavíme rozpätie na 40 000 Hertzov a v cykle spúšťame opäť meranie pomocou metódy a.measure() . Ak chceme uložíme dáta, inak pokračujeme a voláme metódu data inštancie objektu p (rlcparams) a následne voláme fitovaciu funkciu - taktiež metódu objektu p, v ktorej pomocou zložitých matematických funkcií a pravidiel dostávame optimálny (maximálny) výrez impedancie. Výsledok vypíšeme ako string pomocou príkazu: print str(p.fResonanceFit) + '\t' + str(p.R1) + '\t' + str(p.C0) + '\t' + str(p.C1) + '\t' + str(p.L1).

7.4 Ďalšie možné testovacie scenáre

Chceli by sme taktiež otestovať viaceré možné chybové hlášky a to napríklad chybnou inicializáciou triedy agilent : nastavením iného typu diagramu, ako smith chart, napríklad PHAS - ktorý nastaví formát na fázový formát (':calc1:form phas), alebo REAL - reálny formát (':calc1:form real'). ďalej nastaviť sweeping point cez povolenú hranicu 1601 bodov a aj pod najnižšiu povolenú hranicu 2 body - ':sens1:swe:poin 1800' / ':sens1:swe:poin 1'. Vyskúšať nastaviť iný formát dát ako ASCii pomocou príkazu ':form:data *' na real a real32 (sú iba 3 možné formáty). Ďalej by sme sa chceli zamerať v testoch na nastavenie frekvencií a to presiahnutím rozmedzia pre center aj span (stred a

rozpätie) center má rozmedzie od 3E5 do 3E9 (podľa mojich odhadov to znamená 3 na 5 až 3 na 9) TO DO zistiť spresniť, a pre span je to rozmedzie od 0 do 2.9997E9.

Potom by sme sa radi zamerali na série meraní s správnymi hodnotami - v rámci rozmedzia, ale prednastavenými podľa náhodného výberu - vďaka random generatoru a sledovať výsledky a priebeh meraní.