TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

Studijní program: B2612 – Elektrotechnika a informatika
Studijní obor: 1802R007 – Informační technologie

Webové rozhraní programu Ferrodo

Web interface of programme Ferrodo

Bakalářská práce

Autor: **Vladimír Hrdina**
Vedoucí práce: Ing. Pavel Márton, Ph.D.

V Liberci 17. 5. 2013

Originál zadání

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu TUL.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Bakalářskou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce a konzultantem.

Datum

Podpis

Poděkování

Na tomto místě bych chtěl poděkovat Ing. Pavlu Mártonovi, Ph.D. za vedení práce a také mé rodině, přítelkyni a kamarádům, kteří mě po dobu studia podporovali.

Věnování

Tuto bakalářskou práci bych chtěl věnovat mému tátovi, který se vinou náhlých zdravotních potíží v březnu roku 2013 nedočkal jejího dokončení.

Abstrakt

Bakalářská práce seznamuje čtenáře s programem Ferrodo a základy testování softwaru. Dále popisuje vytvořenou webovou aplikaci pro automatizované testování výstupních souborů programu Ferrodo porovnávacími skripty a vizualizaci výsledků, která slouží pro včasné a adresné odhalování chyb. Poté je aplikace srovnána s jinými programy.

Druhá část práce se věnuje vytvoření grafického webového rozhraní pro ovládání pomocných konzolových programů pro postprocesing výsledků programu Ferrodo s možností rozšíření o další programy pomocí vytvořeného generátoru rozhraní na základě konfiguračního souboru ve formátu XML.

Klíčová slova: Testování softwaru, vizualizace dat, uživatelské rozhraní, webová aplikace

Abstract

Bachelor thesis introduces program Ferrodo and gives basic information about software testing. It describes development of an application for automated testing of Ferrodo. It is based on comparsions and visualization of differences between actual and reference data. The application is designed to serve a purpose of early detection of errors during development of Ferrodo. Comparsion with other programs is provided.

Second part describes web interface for control of auxiliary console programs for post-processing of the results from the Ferrodo. Interface is generated based on XML configuration file. This allows i) simple extendability for additional (not yet implemented) features of auxiliary programs, ii) use for other programs designed to interpret Ferrodo data, and even those which are not related to Ferrodo.

Keywords: Software testing, data visualization, user interface, web application

Obsah

Seznam ilustrací

Seznam zkratek

- PHP – PHP: Hypertext Preprocessor je skriptovací programovací jazyk. Je určený především pro programování dynamických internetových stránek a webových aplikací
- HTML – HyperText Markup Language je značkovací jazyk používaný zejména k tvorbě webových stránek
- XML – eXtensible Markup Language je značkovací jazyk požívaný pro přenos dat
- AJAX – Asynchronous Javascript and XML je obecné označení pro technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek bez nutnosti jejich znovunačítání
- JSON – JavaScript Object Notation je způsob zápisu dat nezávislý na počítačové platformě, určený pro přenos dat

1 Úvod

Feroelektrické materiály vykazují elektrické, mechanické a elektromechanické vlastnosti, které jsou intenzivně studovány a využívány v množství důležitých aplikacích, zejména v oblasti senzorů, ale také ve vstřikovacích jednotkách motorů, akcelerometrech mobilních telefonů, atd. Vlastnosti materiálu mohou být do značné míry ovlivněny komplikovanou mikrostrukturou materiálu. Formování a vývoj této mikrostruktury často není možné postihnout analytickými přístupy. Zde využíváme numerických simulací.

V oddělení dielektrik FzÚ AV ČR je vyvíjen program `Ferrodo` pro numerické simulace mikrostruktury. Vstupem programu je soubor s polem polarizace, z něhož se bude doménová struktura vyvíjet a řada parametrů, které umožňují nastavit rozmanité podmínky. Hlavním výstupem programu jsou textové a grafické soubory vektorového pole polarizace. [1]

Při vývoji tohoto programu je potřeba testovat konzistenci výstupů mezi jednotlivými verzemi programu. První část práce se věnuje způsobům testování softwaru. Dále je zde popsán způsob testování výstupů programu `Ferrodo` a představeny používané programy a technologie. Poté je popsána webová aplikace vytvořená pro automatizaci testování výstupů a porovnána s jinými programy.

Druhá část práce se věnuje návrhu a realizaci rozhraní programů pro postprocesing výstupů z programu `Ferrodo`. Popisuje rozhraní pro program `Manipulate` s možností výběru bodů z vizualizovaného vektorového pole a dále popisuje vytvořený generátor rozhraní pro další programy pomocí konfiguračního souboru.

Část 1:

Automatizované testování programu Ferrodo

2 Testování softwaru

Testování softwaru je nedílnou součástí vývoje softwaru. Včasným odhalením chyby můžeme ušetřit spoustu času a peněz, ale i zabránit katastrofám. Náklady na opravu chyby nalezené už při specifikaci softwaru, nebo při jeho návrhu jsou zanedbatelné, zatímco chyba nalezená po uvedení softwaru na trh nám může způsobit nemalé potíže. Chyba v programu například způsobila nefunkčnost obranného raketového systému Patriot [2]. V případě programu `Ferrodo` se nejedná o tak dramatickou věc, ale pro důvěryhodnost vědeckého programu je konzistence dat a včasné a rychlé odhalování chyb nutné.

I při dodržení všech pravidel a použití správných postupů, se téměř vždy v softwaru najdou chyby. S testováním souvisí několik axiomů.

- Žádný program není možné otestovat kompletně

  Po kompletní ověření bychom museli vyzkoušet všechny možné kombinace vstupů, což není ve většině případů možné v reálném čase. Proto se v praxi používá pouze podmnožina všech možných kombinací. Tím ale není software otestován kompletně.
- Testování nikdy nemůže prokázat, že chyby neexistují

  Můžeme prokázat existenci chyb, ale nemůžeme dokázat jejich nepřítomnost.
- Čím více chyb najdeme, tím více jich v softwaru je
- Paradox pesticidů

  Opakování stále stejných testů odhalí a opraví chyby, žádné další ale nenalezne, proto je nutné testy měnit. Analogie k postřikům proti hmyzu, kdy si hmyz na aplikování stále stejného postřiku vytvoří imunitu.

Statické a dynamické testování

Testování můžeme dělit na statické a dynamické. Statické testování nevyžaduje spuštěný software, testujeme statické části, tedy specifikaci, či dokumentaci.

Dynamické testování provádíme na běžícím softwaru.

Testování černé a bílé skříňky

Při testování černé skříňky nevidíme do kódu programu (proto černá –

neprůhledná). Software testujeme z pohledu uživatele, testujeme vstupy a výstupy.

Při testování bílé skříňky (někdy také skleněné, průhledné) máme k dispozici zdrojový kód a můžeme tak odhadnout kde spíše hledat chyby a kde ne.

Střední variantou je testování šedé skříňky, kdy nemáme k dispozici celý kód, ale jsme informování o vnitřních algoritmech

Statické a dynamické testování bíle skříňky

Testování bílé skříňky je méně běžné, než testování skříňky černé. Při statickém testování provádíme inspekce kódu a snažíme se odhalit chyby v deklaraci proměnných, správnost odkazů na data, či správnost porovnání. Také kontrolujme zda jsou funkce volány se správnými argumenty a jestli je ošetřen nenadálý výpadek externího zařízení, se kterým program pracuje. V dnešní době už většinu této práce obstarají kompilátory, které ale nemusí všechny chyby odhalit.

Dynamické testování je důležité nezaměňovat s klasickým laděním (debugging), ač se obě techniky zdají na první pohled velmi podobné. Laděním ale chyby opravujeme, testováním je pouze vyhledáváme.

Při dynamickém testování bílé skříňky například nalezneme rovnici, v níž se dělí nulou a hned pro tento případ připravíme test. Při testování černé skříňky nemusíme tuto skutečnost znát a případ dostatečně otestujeme. Také vidíme jak se program větví a můžeme tak otestovat všechny větve programu.

Testování bílé skříňky je stěžejní částí programovací techniky zvané programování řízené testy, kde se nejprve napíší testy pro danou funkcionalitu, zkontroluje se jejich nesplnění a až poté je napsán kód. Pokud kód splní testy, tak také splňuje všechny specifikované požadavky. Používají se zde takzvané jednotkové testy, které testují pouze konkrétní část kódu, například funkci a jednotlivé testy by měli být na sobě nezávislé.

Statické testování černé skříňky

Dokument specifikace požadavků je souhrn požadavků, které musí software splnit. Určuje co, jak a za jakých podmínek má software dělat, definuje omezení (počet uživatelů, přístup k souborů, …), vstupy a výstupu programu, apod. [4]

Testováním a odhalením chyb tohoto dokumentu můžeme odhalit některé problémy ještě před samotným vývojem softwaru. Ne vždy je ale specifikace požadavků k dispozici

Dynamické testování černé skříňky

Jak jsme si už řekli, není možné otestovat všechny možné vstupy a výstupy, proto je nutné zvolit vhodnou podmnožinu testových případů. Nejprve je vhodné aplikovat tzv. *testování splněním.* Zvolíme několik běžných případů, které očekáváme, že budou při používání softwaru nastávat. Analogii hledejme u právě vyrobeného automobilu. Také ihned nevyjedeme automobil testovat na závodní okruh a testovat jeho hraniční možnosti. Nejprve zkontrolujeme pneumatiky, upevnění zrcátek, apod. [3]

Až po úspěšném *testování splněním* přikročíme k *testování selháním.* Jednou z metod testování selháním je tzv. procházka po okraji. Zde zkoušíme programu podsunout hodnoty, které jsou na konci, či začátku intervalu a také hodnoty překračující dané meze, zápis do zaplněného zásobníku, či změnu data na neexistující den (např. 30. února). Zároveň zde také testujeme reakci na prázdné, nulové, nedefinované, či prázdné vstupy. Kromě procházky po okraji se při testování selháním snažíme navodit stavy, při kterých by mohl program selhat. Pokud máme zadat číslo, zkusíme napsat znaky. Do pole pro kladné číslo zkusíme zapsat číslo záporné a do pole určeného pouze pro celá čísla číslo reálné. U textových polí testujeme reakci na různé národní znaky a jejich korektní zobrazování, nebo reakci na speciální znaky, jako podtržítko, lomítko, apod. Pokud software pracuje se soubory, zkusíme načíst soubor, který není určený pro program, nebo zápis do oblasti mimo naše oprávnění.

Jedním z druhů testování černé skříňky jsou regresní testy. Takové testy jsou spouštěny po každé změně softwaru, abychom se ujistili, že změny neměli vliv na výstup softwaru. Používají se modelové případy a testy musí být stále stejné. Regresní testy se často používají spolu s jednotkovými testy.

Vyhodnocování testových sad programu `Ferrodo` popsané v této práci můžeme zařadit právě mezi regresní testování. Výsledky programu v závislosti na vstupních parametrech porovnáváme s očekávanými výsledky.

Další postupy testování

Zkoumání programového kódu a kontrola vstupů a výstupů programů není jedinou oblastí testování softwaru. Je také nutné otestovat, zda program běží na různých kombinacích hardwaru a na různých operačních systémech. Také bychom měli zkontrolovat dokumentaci, jestli je dostatečně podrobná a přehledná. Pokud je software vícejazyčný je nutné zkontrolovat, jestli například různá délka textů nezmění rozvržení grafického prostředí, apod.

Automatizované testování

Automatizované testování se provádí pomocí softwarových nástrojů. Spouštěné testy jsou při vhodném počátečním nastavení a volbě testů schopny běžet bez zásahu uživatele (kromě nastavení) a tím eliminovat lidskou chybu.

Může šetřit čas a náklady. Vhodné je zejména pro opakované spouštění testu, například u regresního testování, což je případ aplikace popsané v první části této práce. Díky rychlosti automatického testování můžeme testovat mnohem více vstupních možností.

Automatizované testování je vhodné použít u větších projektů. U malých projektů převýší náklady na pořízení software a konfiguraci testových úloh náklady na manuální testování. Navíc nelze automatizovat všechny druhy testování a manuální testování je přesnější.

Vybrané nástroje pro automatizované testování

Testcomplete

TestComplete od firmy Smartbear Software je profesionální a komplexní nástroj pro testování softwaru. Umožňuje regresní testování, jednotkové testy, testování grafického rozhraní, zátěžové testy, testy databází a mnoho dalšího pro většinu používaných jazyků, jako je mj. Java, .NET nebo Flash. Podporuje také testování webových aplikací včetně standardu HTML5. Testovací případy je možné buď psát v jednom z podporovaných jazyků, nebo zaznamenat posloupnost interakcí myši a klávesnice. TestComplete není volně dostupný a jeho cena se pohybuje okolo dvou tisíc dolarů. [5]

HP Quality Center

Quality Center od firmy Hawlett – Packard je nástroj pro správu, plánování a spouštění testů. Je k němu možné přistupovat přes prohlížeč a vzdáleně úlohy spouštět. Pro samotné testování může využívat například nástroj HP QuickTest Professional, který je součástí sady nástrojů. [6]

Další nástroje

Na trhu je mnoho dalších nástrojů, například IBM Rational Functional Tester a Testing Anywhere. Mnoho nástrojů slouží pouze pro testování webových aplikací, například volně dostupný Selenium, nebo Oracle Application Testing Suite, který umožňuje i testování databází.

3 Aplikace pro testování souborů

Pro automatizované porovnávání výstupních souborů z programu `Ferrodo` pomocí porovnávacích skriptů jsem vytvořil webovou aplikaci, která spouští skripty a vizualizuje získaná data.

3.1 Porovnávací skripty

U programu `Ferrodo` kontrolujeme konzistenci výstupů mezi jednotlivými verzemi porovnáváním některých výstupních souborů s referenčními soubory. K porovnání využíváme tří programů. Prvním nástrojem je `Diff`, který je standardně součástí operačního systému. `Diff` slouží pro podrobné porovnávání dvou souborů, případně i složek. Hledá společné a rozdílné části souborů a vypisuje rozdílné řádky [7]. Má několik možností zobrazení výsledků, ale pro naše účely je využíván pouze výchozí formát. Porovnání je vyhovující, pokud není nalezen žádný rozdíl.

Příklad vstupních souborů a výstupu programu diff:

Soubor a.tx

```
first line - dog
second line - bird
fourth line – snake
```

soubor b.txt

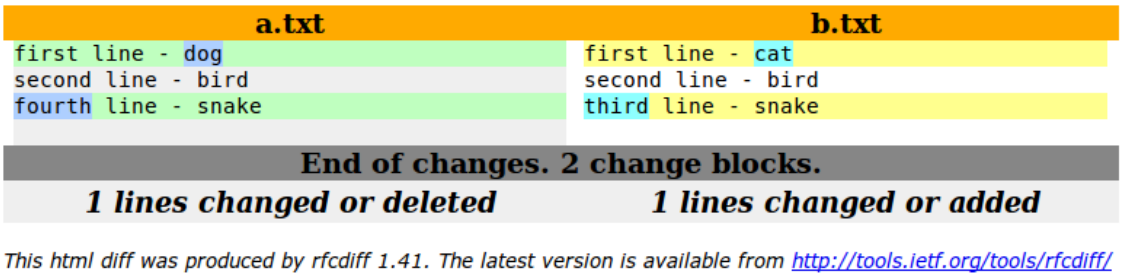
```
first line - cat
second line - bird
third line – snake
```

Výstup programu diff:

```
1c1
< first line - dog
---
> first line - cat
3c3
< fourth line - snake
---
> third line - snake
```

Druhým nástrojem je skript `Ndiff` [8]. Slouží pro numerické porovnání souborů. Umožňuje nastavit maximální absolutní a relativní přípustnou odchylku. Absolutní odchylka je rozdíl porovnávaných hodnot a relativní odchylka je podíl absolutní odchylky a větší z porovnávaných hodnot. Výstupem je počet případů přesahujících alespoň jednu maximální přípustnou odchylku a hodnoty nejvyšší nalezené absolutní a relativní odchylky v souborech. Porovnání je vyhovující, pokud je počet hodnot přesahujících maximální přípustné odchylky roven nule.

Třetí nástroj je skript `Rfcdiff [9]`. `Rfcdiff` přebírá dva soubory a zobrazuje rozdíly v několika formátech. V aplikaci je využívám barevný html výstup (viz Obrázek 1). Skript není používán pro vyhodnocení souborů. Ale pouze pro vytvoření přehledného logu s barevným zvýrazněním rozdílů.



Obrázek 1: Výstup skriptu Rfcdiff

Může nastat případ, kdy uživatel rozhodne, že je porovnávaný soubor i přes nevyhovující porovnání správný. V takovém případě bude původní referenční soubor zálohován a za jeho název připojeno aktuální datum. Pro každé další porovnávání bude použit nový referenční soubor.

3.2 Požadavky a nastavení serveru

Pro automatizaci testování výstupních souborů jsem navrhl a vytvořil webovou aplikaci. K běhu aplikace je nutný webový server na linuxovém operačním systému.

Pro běh aplikace je nezbytná přítomnost webového serveru. Nejpoužívanějším webovým serverem je Apache HTTP server [10], ale aplikace by měla fungovat i na jiných webových serverech. Další podmínkou je přítomnost PHP, což je volně šiřitelný, objektově orientovaný serverový skriptovací jazyk navržený pro potřeby webových stránek. PHP je interpretováno na straně serveru a generuje výstup, většinou ve formátu HTML. PHP je platformě nezávislé.

Vznik PHP se datuje do roku 1994, kdy Rasmus Lerdorf vytvořil systém evidence přístupů ke svému webu, nejdříve v jazyce Perl, poté v C. Systém se stal populárním i mezi ostatními vývojáři, postupně vylepšoval a vznikl tak systém Personal Home Page Tools, později Personal Home Page Construction Kit [11]. Současná verze jazyka PHP je verze 5.

PHP můžeme nainstalovat z repozitáře (balíček php5). Pokud chcete využívat možnosti vzdáleného nahrávání testových sad (popsáno v kapitole 3.6) musíte z důvodu nedostatečné výchozí velikosti pro nahrávané soubory provést dodatečná nastavení v souboru `php.ini`. Úprava se týká těchto řádků: [12]

- `upload_max_filesize = 50M`

  Maximální velikost souboru, v tomto případě 50 Mb. Výchozí hodnota je 2 Mb. Nutno zohlednit předpokládanou velikost celé sady.

- `max_execution_time = 30`

  Maximální počet sekund pro vykonání skriptu. Zde je nutné uvážit předpokládané rychlosti internetového připojení obou stran komunikace

- `max_input_time = 60`

  Čas po který může skript zpracovávat vstupní data. Změna bude nutná pouze u velkých souborů přesahujících stovky Mb

- `memory_limit = 128M`

  Maximální množství paměti pro skript. Je nutné nastavit více, než je velikost nahrávaného souboru

- `post_max_size = 60M`

  Maximální velikost dat předávaných metodou POST. Je nutné nastavit hodnotu větší než u `upload_max_filesize`

3.3 Adresářová struktura

Výstupní soubory programu `Ferrodo` jsou generovány při použití různých parametrů. Soubory odpovídající jedné sadě parametrů jsou sdružovány do adresáře. Dále budeme takový adresář nazývat test. Jednotlivé testy jsou dále sdruženy do adresáře, jehož jméno odpovídá datu vygenerování souborů, tomuto adresáři budeme dále říkat sada testů.

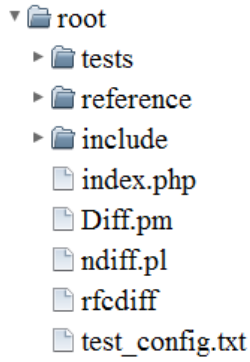
Referenční soubory strukturou odpovídají sadě testů. Soubory pro porovnání v jednotlivých testech ale nejsou přímo ve složce testu, jako je tomu u souborů výstupních, ale nachází se v podadresáři `RESULT`. V adresáři s referenčním testem se také mohou nacházet soubory `INIT`, `OUT_final.gif` a `OUT_all_img.gif`. Soubor `INIT` je inicializační soubor, podle kterého byly soubory vygenerovány, `OUT_final.gif` je vizualizované vektorové pole a `OUT_all_img.gif` je obrázek všech vygenerovaných obrázků.

Sady testů jsou uloženy v adresáři `tests`. Jednotlivé sady jsou dále členěny na adresáře testů. Adresáře testů obsahují soubory určené pro testování (viz Obrázek 3).

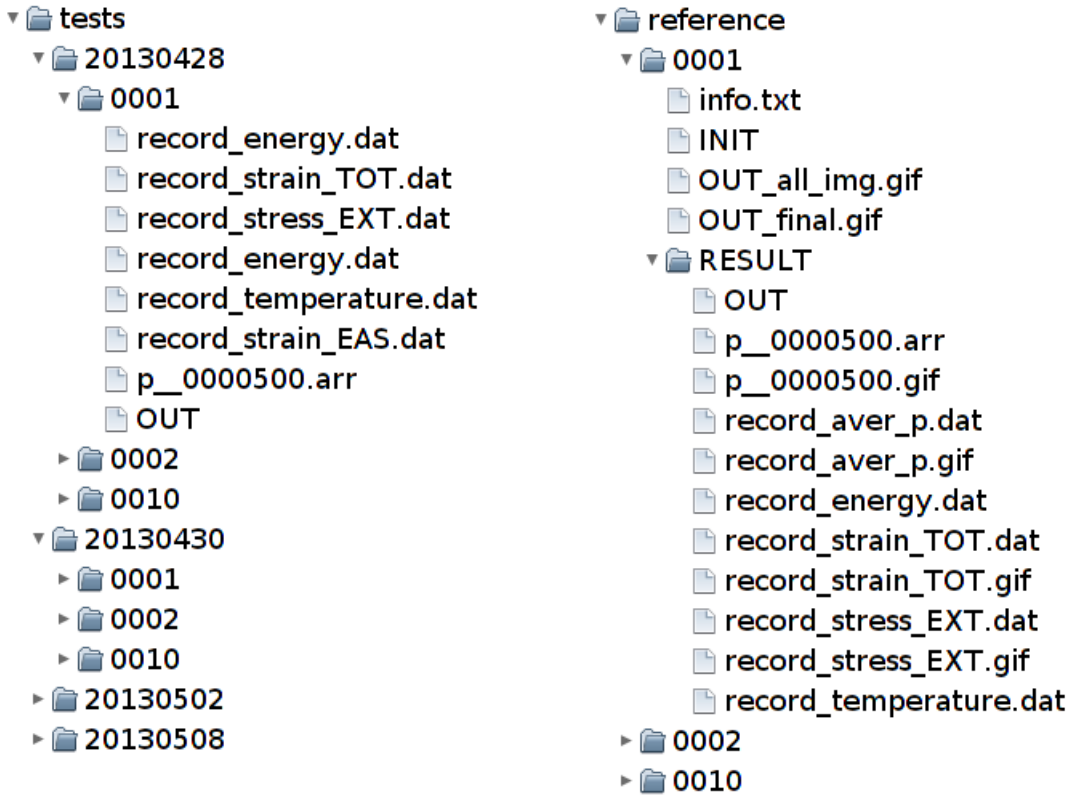
Sada s referenčními daty se nachází v adresáři `reference`. Struktura odpovídá běžné sadě testů s tím rozdílem, že dílčí soubory jsou zanořeny v podadresáři `RESULT`. Referenční data jsou společná pro všechny sady testů.

Všechny soubory aplikace, kromě souboru `index.php`, který je v kořeni aplikace, jsou v adresáři `include`.

Pro správný chod aplikace je nutné zajistit dostatečná oprávnění. Aplikace (uživatel `www-data`) musí mít právo číst všechny soubory a mít právo zápisu do adresářů `tests` a `reference`.



Obrázek 2: Adresářová struktura aplikace



Obrázek 3: Adresářová struktura porovnávaných souborů

3.4 Konfigurační soubor

Uživatel souborem `test_config.txt` určuje, které soubory z kterého testu porovnávat a jakým skriptem. Jednotlivé testy mohou být členěny do sekcí. Název sekce je uvozen znakem „-“. Testy jsou odděleny prázdným řádkem (nutná podmínka) a za názvem testu následují na jednotlivých řádcích soubory, které mají být porovnány

s referenčními soubory. Za názvem souboru je příznak určující porovnávací skript („N“ pro `Ndiff`, „D“ pro `Diff`) a v případě použití skriptu `Ndiff` následují hodnoty maximální absolutní a relativní odchylky.

```
-Harmonic polarization profiles

0010
record_energy.dat N 1E-10 1E-10
record_aver_p.dat N 1E-10 1E-10
force_distrib_EAS.arr N 1E-10 1E-10
force_distrib_EAS.gif D

0011
record_energy.dat N 1E-10 1E-10
record_strain_TOT.dat N 1E-10 1E-10
record_aver_p.dat N 1E-10 1E-10
p__0000000.arr N 1E-10 1E-10

-External conditions

0110
record_energy.dat N 1E-10 1E-10
record_strain_TOT.dat N 1E-10 1E-10
record_stress_EXT.dat N 1E-10 1E-10
record_aver_p.dat N 1E-10 1E-10
p__0003000.arr N 1E-10 1E-10
```

Obrázek 4: Ukázka souboru test_config.txt

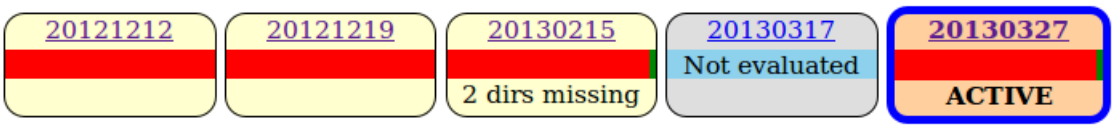
Příklad tohoto souboru zobrazuje Obrázek 4 – Z testu `0010` v sekci „Harmonic polarization profiles“ budou soubory `record_energy.dat`, `record_aver_p.dat` a `force_distrib_EAS.arr` porovnány skriptem `Ndiff` a maximální absolutní i relativní odchylka bude 1e-10. Soubor `force_distrib_EAS.gif` bude porovnán programem `Diff`.

3.5 Vyhodnocování sady testů

Stěžejním skriptem aplikace je `index.php`, který zajišťuje volání funkcí pro vyhodnocování sady, výpisy testů a nové vyhodnocení sady. Tento skript nejprve projde složku `tests` a zaznamená zde přítomné sady testů. Pokud není uživatelem zvolena konkrétní sada (volba bude popsána později) je vybere poslední sadu. Zkontroluje, zda

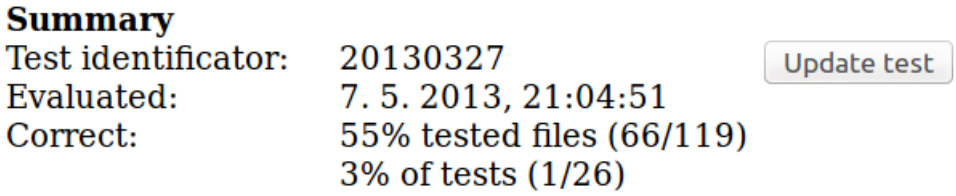
již byla sada v minulosti vyhodnocena. Pokud ano, zavolá funkci `printCards` ze souboru `printFunctions.php` ze složky `include`, která vypíše informační karty jednotlivých sad.

Karta je rozdělena na tři vodorovné části. V horní části je název sady, který zároveň slouží jako odkaz na danou sadu. V prostřední části je červenou a zelenou barvou vyjádřena procentuální úspěšnost sady. Je to poměr testů, kde všechny soubory vyhovují a testů jakkoli nevyhovujících. Pokud test vyhodnocen doposud nebyl, je místo procentuálního vyjádření sady vypsána tato informace. Ve spodní části karty je vypsán počet testů u kterých chybí všechny soubory. Pokud nejsou žádné nekompletní testy nalezeny, zůstává spodní část prázdná. Aktuálně prohlížená sada je barevně zvýrazněna.



Obrázek 5: Karty sad testů

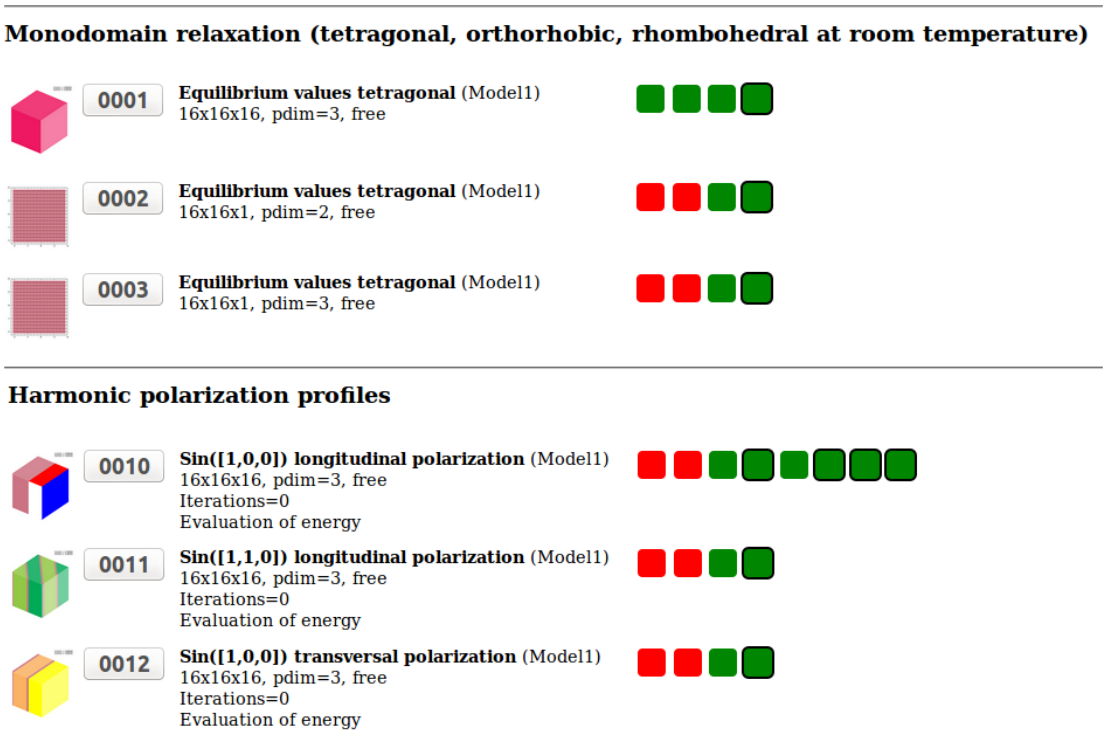
Následně je zavolána funkce `printTest` ze souboru `printFunctions.php`, která vypisuje danou sadu a obstarává kopírování testovaných souborů do referenčních. Nejprve je vypsána hlavička obsahující název sady, tlačítko pro nové vyhodnocení sady, čas vyhodnocení a informace o počtu vyhovujících souborů a zcela vyhovujících testů.



Obrázek 6: Hlavička sady

Pod hlavičkou následuje výpis sekcí a testů. Před názvem testu se pro zpřehlednění testové sady nachází obrázek vizualizovaného vektorového pole za nímž následuje samotný název testu ve formě tlačítka, které je možné použít pro zobrazení

detailních informací o daném testu. Za názvem následuje popis testu ze souboru info.txt z adresáře s referenčním testem. Pro formátování popisu je možné využít HTML syntaxi. Za popisem následují barevné čtverce reprezentující úspěšnost jednotlivých souborů v testu (pro každý soubor jeden). Zeleným čtvercem označený soubor je vyhovující, červeně označený ne. Černý čtverec značí chybu vyhodnocování – testovaný, či referenční soubor neexistuje, nebo selhal vyhodnocovací skript. Čtverec týkající se souboru s koncovkou `arr` je zvýrazněn černým rámečkem. Jako poslední se může objevit chybová hláška o nekonzistenci dat (některý ze souborů chybí), nebo o tom, že chybí všechny soubory testu.



Obrázek 7: Výpis sekcí a testů

Po kliknutí na název testu se rozbalí detailní informace o souborech v něm obsažených. Za názvem testu je tlačítko pro nahrazení všech referenčních souborů porovnávanými a odkazy na soubory `INIT` a `OUT_all_img.gif`, popis ze souboru `info.txt` se přesunul pod název testu. Následuje tabulka s kartami jednotlivých souborů. Názvy sloupců v záhlaví tabulky jsou opatřeny odkazy na jednoduchou nápovědu. Karty obsahují název souboru a pod ním počet případů přesahujících

maximálního hodnoty odchylek určené konfiguračním souborem. Dále následují hodnoty nalezené relativní a absolutní odchylky a pod nimi maximální přípustné hodnoty těchto odchylek. V případě nevyhovujícího souboru jse zde také odkaz na textový log použitého porovnávacího skriptu a odkaz na log skriptu `Rfcdiff`. Následují odkazy na porovnávaný a referenční soubor. V případě že nejsou soubory naprosto totožné (shodnost je určována md5 hašem souborů), je zde tlačítko pro nahrazení referenčního souboru porovnávaným.

Ve spodní části jsou odkazy na výstup testu v textovém formátu, na vzdálené nahrávání sad.

Není-li sada vyhodnocena, je spuštěna funkce `process` ze souboru `compare.php`. Funkce postupně prochází soubor test_config.txt. Narazí-li na prázdný řádek, nastaví příznak změny adresáře. Nachází-li se na dalším řádku název sekce (řetězec uvozený pomlčkou) zaznamená název a pokračuje, pokud to sekce není, uloží obsah řádku jako název dalšího testu. Poté volá pro soubory na dalších řádcích funkci `compare`, které předává název souboru, příznak skriptu, kterým se má porovnávat, cesty k souborů a referencím a v případě použití skriptu `Ndiff` také hodnoty maximálních přípustných odchylek.

Funkce `compare` vrací objekt třídy `Ndiff_result`, nebo `Diff_result` odvozených od třídy `Result`, což jsou třídy, které jsem vytvořil pro uchování informací o porovnávaných souborech. Funkce nejprve zkontroluje přítomnost obou porovnávaných souborů. V případě, že některý ze souborů chybí zaznamená chybu do objektu a objekt vrátí. Také kontroluje přítomnost složky `logs`, do které se vytvářejí logy, která se nachází v adresáři se sadou. Pokud složka neexistuje, tak ji funkce vytvoří. Pokud oba soubory existují, tak je pomocí systémového volání `exec` spuštěn daný porovnávací skript s příslušnými parametry. Poté zpracuje výstup skriptu a hodnoty uloží do objektu. Objekty jsou ukládány do pole.

Po projití celého souboru `test_config.txt` je pomocí funkce `serialize` serializováno a uloženo do souboru. Také je pomocí funkce `arrayToFile` vytvořena textová varianta vyhodnocené sady. Na začátku textové varianty se nachází název sady a čas vyhodnocení. Následují sekce s jednotlivými testy. Soubory v testech mají

v případě použití skriptu `Ndiff` následující strukturu: název souboru, počet případů přesahujících maximální přípustné odchylky, nalezená absolutní odchylka, nalezená relativní odchylka, maximální přípustná absolutní odchylka a maximální přípustná relativní odchylka. V případě použití programu `Diff` se za názvem nachází informace, zda porovnání vyhovuje, či nikoliv.

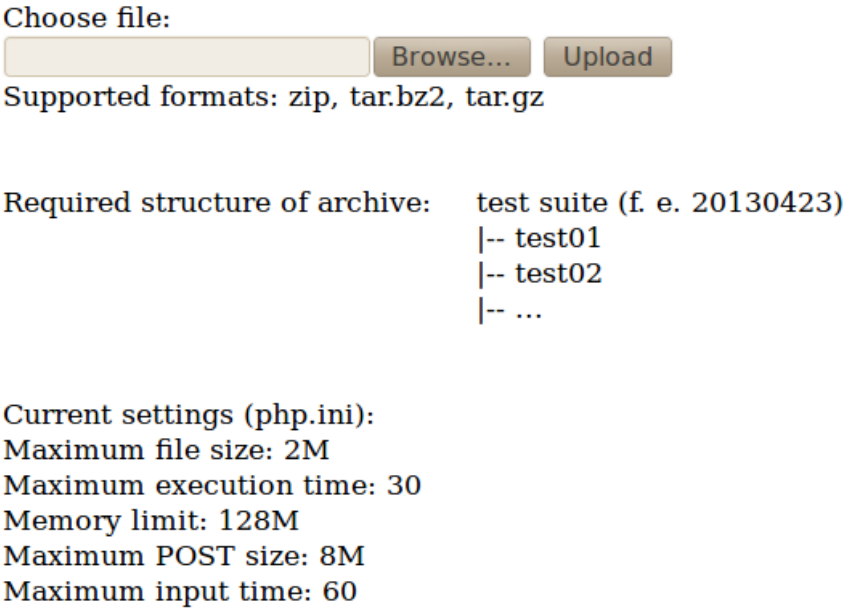
```
20130327
7. 5. 2013, 21:04:51

Monodomain relaxation (tetragonal, rhombohedral at room temperature)
0001
    record_energy.dat        0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    record_strain_TOT.dat    0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    record_aver_p.dat        0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    p__0000500.arr           0  0.00e+0 0.00e+0 1.00e-10 1.00e-10

0002
    record_energy.dat        92 0.00e+0 3.96e-6 1.00e-10 1.00e-10
    record_strain_TOT.dat    32 0.00e+0 3.85e-6 1.00e-10 1.00e-10
    record_aver_p.dat        0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    p__0000500.arr           0  0.00e+0 0.00e+0 1.00e-10 1.00e-10


Harmonic polarization profiles
0010
    record_energy.dat        9  0.00e+0 2.25e-6 1.00e-10 1.00e-10
    force_distrib_EAS.arr    0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    force_distrib_EAS.gif    Evaluated by diff - passed
    energy_density_GRA.arr   0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
    p__0000000.arr           0  0.00e+0 0.00e+0 1.00e-10 1.00e-10
```

Obrázek 8: Ukázka textové varianty vyhodnocované sady.

V případě stisku tlačítka pro nové vyhodnocení je smazán soubor se serializovaným polem a spuštěno nové vyhodnocení.

3.6 Vzdálené nahrávání sad

Aplikace umožňuje vzdálené nahrávání sady testů přes jednoduché webové rozhraní. O nutnosti nastavit PHP server z důvodu nedostatečné výchozí maximální velikosti nahrávaných souborů se zmiňuje kapitola 3.2. Sada musí být komprimovaná v jednom z těchto formátů: `zip`, `tar.bz2`, nebo `tar.gz`. Archiv obsahuje složku se sadou, která obsahuje dílčí testy. Archiv je rozbalen do složky `tests`.



Obrázek 9: Rozhraní pro vzdálené nahrávání sad

3.7 Porovnání s ostatními nástroji

Tato aplikace není univerzální a je na míru uzpůsobena konkrétnímu používání pro porovnávání výstupních souborů z programu `Ferrodo`. I proto také nedosahuje funkčnosti mnohých dostupných nástrojů (některé jsou zmíněné v kapitole 3.7).

Zásadním rozdílem proti klasickému testování je, že se v případě použití skriptu `Ndiff` neporovnávají konkrétní výsledky, ale je dovolena určitá tolerance. Univerzální testovací nástroje sice umožňují spouštění skriptů, ale pro automatizaci by musel být vytvořen další skript, zpracovávající konfigurační soubor a vyhodnocující výsledky. Podobný postup při použití nástroje `Bitten` je popsán v bakalářské práci Michala Nekvasila [14]. V případě použití skriptu by bylo problematické zajistit přehledné a detailní zobrazování výsledků, nehledě na možnost nahrazování referenčních souborů. Vývoj takového skriptu by možný byl, ale zabral by mnoho času. Při použití této aplikace stačí pouze nahrát na server porovnávaná data a sestavit konfigurační soubor.

Část 2:

Prohraní programů pro postprocesing

4 Rozhraní programů pro postprocesing

Obsluha programů z příkazové řádky není, zvláště při větším množství parametrů, příliš přehledná a ne každý uživatel s ní umí dobře a rychle pracovat. Volba parametrů pomocí jednoduchého webového rozhraní je komfortnější a přehlednější variantou. Při opakovaném zpracovávání se nemusí upravovat celý příkaz, ale pouze konkrétní parametr. Také můžeme parametry omezit (např. možnost zadat pouze číslo, či definovat přípustné hodnoty).

Druhá část mé bakalářské práce se zabývá vytvořením webového uživatelského rozhraní programu `Manipulate` s možností rozšíření o další programy. Program `Manipulate` pracuje se soubory ze simulací programem `Ferrodo`. Jedná se o stejné soubory, které testuje aplikace z první části práce.

Rozhraní pro program budu dále nazývat modul. Moduly obsahují formulářové prvky pro nastavování parametrů programu a výsledky předchozích průběhů programu.

Pro chod aplikace je nutný webový server na linuxovém operačním systému a PHP stejně jako v aplikaci z první části.

4.1 Adresářová struktura

Soubory aplikace jsou rozděleny do několika adresářů

- `data` – do tohoto adresáře jsou umísťovány soubory se kterými mohou programy pracovat a jsou zobrazitelné přes prohlížeč serverových souborů v aplikaci
- `history` – adresář pro uchovávání výsledků jednotlivých programů
- `include` – adresář pro php skripty aplikace
- `modules` – adresář pro konfigurační soubory rozhraní
- `scripts` – adresář pro javascripty
- `software` – adresář pro programy pro něž jsou vytvářena rozhraní
- `styles` – adresář s kaskádovými styly
- `tmp` – adresář pro dočasné soubory

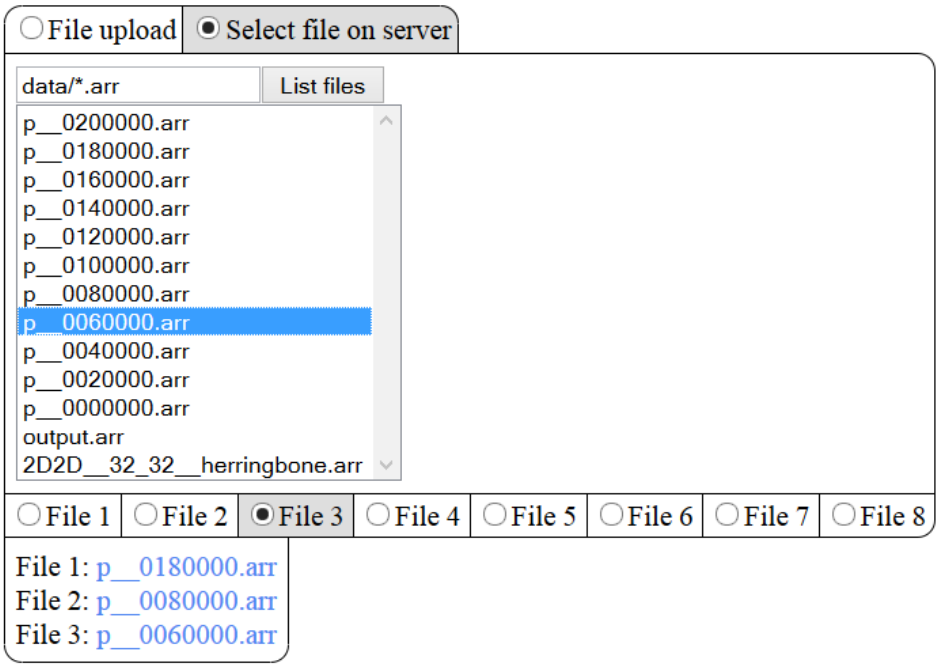
Pro správný chod aplikace je nutné nastavit oprávnění. Webový server (uživatel `www-data`) musí mít právo číst všechny adresáře a do adresářů `data`, `history` a `tmp` musí mít právo zápisu

4.2 Formulář pro výběr souborů

Předpokládá se, že většina programů, pro které bude rozhraní používáno bude pracovat se soubory, a to jak se soubory nahrávanými z disku počítače klienta, tak se soubory umístěnými na serveru. Pro výběr a nahrávání souborů jsem vytvořil formulář, který je společný pro všechna rozhraní.

Program `Manipulate` pracuje s jedním až dvěma soubory, ale další, uživatelem definované programy mohou využívat až osm různých souborů. Volba „File upload“, zobrazí standardní formulářové vstupní pole pro výběr souboru a tlačítko „+“, které přidává další pole pro výběr dalších souborů až do maximálního počtu osmi. Pokud budeme nahrané soubory využívat častěji, můžeme je zkopírovat do složky s daty na serveru zaškrtnutím volby „Copy uploaded files to data folder“.

Druhou možností je výběr souboru z datové složky na serveru. Pro tuto volbu byl vytvořen jednoduchý prohlížeč souborů, jehož vstupem je maska pro výpis souborů v daném adresáři. Například při použití masky „`data/*.arr`“ prohlížeč vypíše všechny soubory s koncovkou `arr` ve složce `data`. Prohlížeč pracuje asynchronně použitím technologie `AJAX`. Soubor se vybere kliknutím na jeho název. Pomocí přepínačů „File 1“ až „File 8“ volíme, který z osmi možných souborů bude nastavován. Zvolené soubory se vypisují pod formulářem.



Obrázek 10: Formulář pro výběr souborů

4.3 Rozhraní pro program Manipulate

Program `Manipulate` pracuje s výstupními soubory programu `Ferrodo` a umožňuje otočení nebo posunutí získaného vektorového pole, vykreslení profilu polarizace podél dané cesty nebo podél kruhu a odečtení dvou polí. Rozhraní pro program `Manipulate` je rozděleno na dva moduly. Modul `Manipulate` obsluhuje funkce pro profilování polarizace a modul `Manipulate (other features)` obsluhuje funkce `shift`, `substract` a `rotate90`. Oba moduly mají svá specifika.

Modul manipulate

Mezi parametry funkce `profile`, která slouží pro vykreslení profilu polarizace, patří mimo jiné také dva body reprezentující přímku ve vektorovém poli hodnot podle níž bude profil vykreslován. Volbu velmi usnadní grafická reprezentace vektorového pole vytvořená programem `Fview`. Existence obrázku je ověřována funkcí `checkImage`, která asynchronně spouští skript `checkImage.php`. Ten vrací získaná data, jako je existence souboru, velikost obrázku a velikost vektorového pole, ve formátu `JSON`. Na základě výsledků skriptu je obrázek buď vykreslen, nebo je

zobrazena nabídka s možností vygenerování obrázku se standardními parametry, nebo přesměrování do modulu programu `Fview` a ruční nastavení parametrů. Generování programem `Fview` se standardními parametry probíhá opět asynchronně bez nutnosti znovu načítat stránku. Existující obrázek není podmínkou úspěšného provedení programu, pouze nebudeme moci zadat body myší.

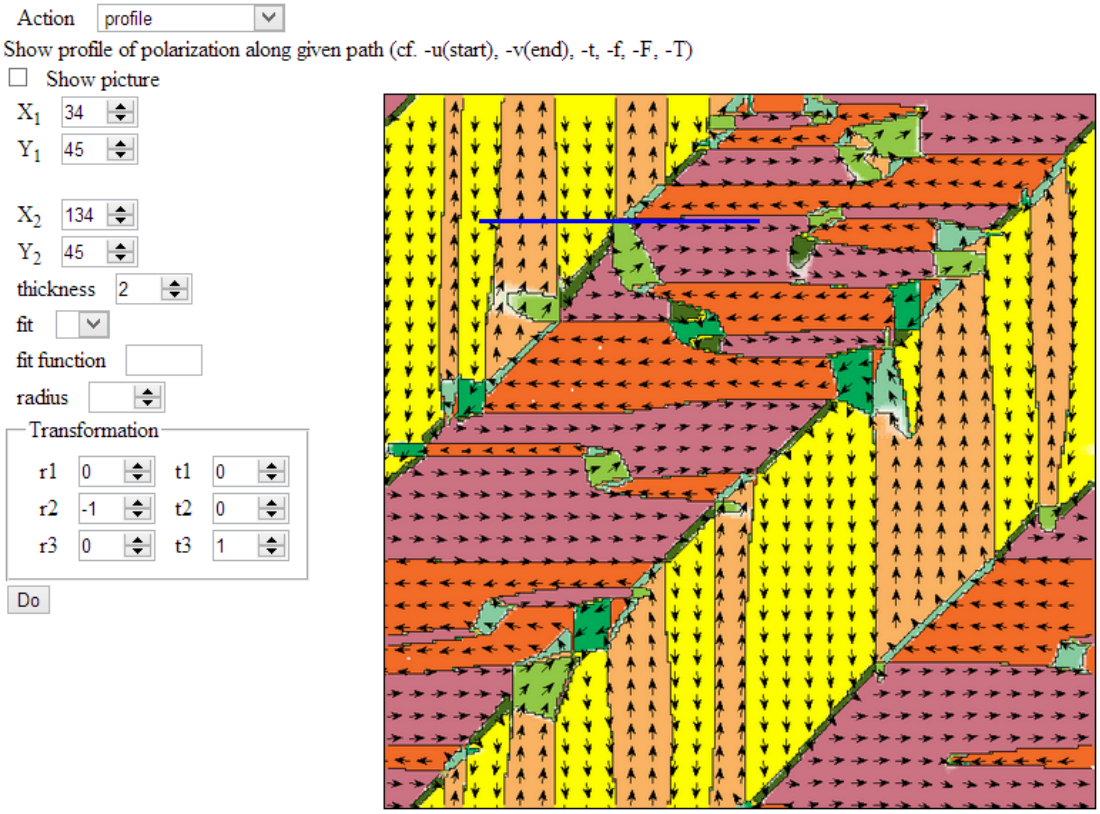
Body je při přítomnosti obrázku možné zvolit pomocí myši. Velikost obrázku ale není stejná jako počet hodnot ze zdrojového souboru. Proto je nutné souřadnice získávané funkcí `getCursorPosition` (viz Obrázek 11) přepočítávat jako poměr velikosti obrázku v pixelech a počtem hodnot v dané ose. Přepočítané hodnoty jsou zobrazovány v příslušných formulářových polích. Funguje zde i opačný postup. Pokud některou hodnotu upravíme, přímka se okamžitě překreslí. Pro jemné úpravy můžeme v prohlížečích podporujících HTML5 využít šipek pro zvýšení, či snížení hodnoty o jedničku.

```
function getCursorPosition(e){

    var curX = e.pageX; // x-coordinate of cursor
    var curY = e.pageY; // y-coordinate of cursor

    var obj = $("#Canvas")[0];

    var objLeft = obj.offsetLeft; // canvas offset
    var objTop = obj.offsetTop;

    return new Array(curX - objLeft , curY - objTop);
}
```

Obrázek 11: Funkce getCursorPosition

Myší zvolený bod je pro přehlednost zvýrazněn terčem tvořeným pomocí dvou soustředných kružnic. Po výběru druhého bodu je vykreslena přímka. Body i přímka jsou zobrazovány pomocí volně dostupné javascriptové knihovny `wz_jsgraphics` [13] a to funkcemi `drawElipse` a `drawLine`. Obrázek 12 zobrazuje formulář s parametry a obrázek s vykreslenou přímkou.

Prezentaci výsledků průběhu programu je popsána v kapitole 4.4.



Obrázek 12: Formulář programu Manipulate s vykreslenou přímkou mezi body

Modul manipulate (other features)

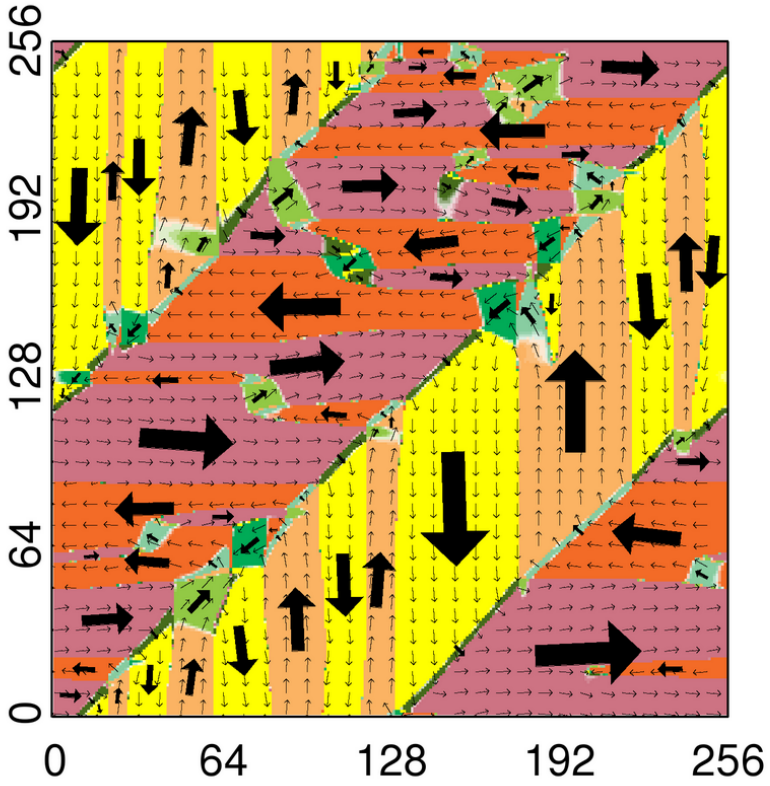
Modul `manipulate (others features)` obsluhuje funkce `shift`, `rotate90` a `substract`. Tyto funkce nepřebírají kromě souborů žádné další parametry. Pro možnost vizuálního porovnání souborů se pod výběrem akce nachází dva boxy, do kterých si můžeme nechat vykreslovat vybrané soubory. Tato funkcionalita má využití zejména u akce `substract` (odečtení dvou souborů). První dva vybrané soubory se vykreslí každý do jednoho boxu. Pro další soubory volíme box přepínačem pod seznamem souborů. Pokud obrázek neexistuje, zobrazí se stejná nabídka generování, která byl popsána u modulu `Manipulate`. Soubor, který má být vykreslen do některého z boxů volíme buď kliknutím na jeho název v seznamu použitých souborů pod formulářem pro výběr souborů, nebo výběrem v prohlížeči souborů.

4.4 Prezentace výsledků

Výsledky získané spuštěním programu jsou zobrazovány pod formulářem jako položky historie. Každá položka obsahuje datum a čas vyhodnocení, použitý příkaz a všechny zvolené parametry. Obsahuje také odkazy na všechny vstupní i výstupní soubory, popřípadě na textový výstup programu. Obrázky nejsou vypsány formou odkazu, ale jsou vykresleny. U každého odkazu na soubor se také nachází tlačítko, kterým lze soubor nastavit pro další použití v aktuálním modulu. V pravém horním rohu je tlačítko pro nevratné smazání položky z historie.

18. 04. 2013, 16:44:44

./software/fview_2.0/fview_2.0_static -i ./tmp/p__0040000.arr --palette 0 -l "color, small_arrows, big_arrows" -s 2
Palette: **0**
Layers: **"color, small_arrows, big_arrows"**
size coefficient: **2**
p__0040000.arr U |



Obrázek 13: Zpracovaný výsledek programu Fview

Položky jsou ukládány po dobu sedmi dní (počet dní může být změněn v souboru `constants.php`), poté jsou automaticky mazány. Ve výchozím stavu se zobrazují pouze položky vyhodnocené v aktuálním sezení. Zobrazení všech uložených položek je možné kliknutím na odkaz „show all history items“ ve spodní části aplikace.

4.5 Rozšiřitelnost rozhraní

Rozhraní by měla být jednoduše rozšiřitelná o další parametry programů a také by mělo být snadné přidat nové rozhraní ve formě webového formuláře pro další programy. Pro tento účel jsem vytvořil vlastní generátor rozhraní, který zpracuje uživatelem definovaný konfigurační soubor ve formátu XML. Uživatel aplikace nemusí pro přidání rozhraní pro další program znát značkovací jazyk HTML a ani skriptovací jazyk PHP. Stačí vytvořit XML soubor. Tímto způsobem lze jednoduše vytvořit rozhraní pro téměř libovolný konzolový program. K dispozici jsou základní formulářové prvky: Textové pole, přepínače (radio), zaškrtávací políčka (checkboxy), rozbalovací seznam a textové pole přijímající pouze čísla (HTML5 prvek number). Návod na vytvoření XML konfiguračního souboru je uveden v příloze.

Generování a obsluhu formuláře probíhá ve skriptu `makeModule.php`. Rozhraní je reprezentováno třídou `CustomForm` a jednotlivé formulářové prvky mají své třídy odvozené od třídy `FormItem`. V konstruktoru třídy, kterému je předán XML soubor, se soubor zpracuje pomocí funkce `parse`. Funkce `parse` prochází XML soubor pomocí v PHP obsažené funkce `simple_xml_parser` a podle nalezených informací nastavuje objekt třídy `CustomForm`, popřípadě vytváří objekty formulářových polí. Po zpracování konfiguračního souboru je formulář vytištěn funkcí `printForm`. V případě odeslání formuláře s vyplněnými parametry programu je zavolána funkce `copyFiles`, která zkopíruje vstupní soubory do dočasné složky a následně funkce `makeQuery`, která sestaví příkaz pro spuštění programu. Poté funkce `processOutput` zajistí Vytvoření položky historie v adresáři `history` a podadresáři příslušného modulu včetně nakopírování všech vstupních a výstupních souborů. Grafická podoba výsledku popsaná v kapitole 4.4 je vytvořena funkcí `makeHistory`.

```php
class Select extends FormItem{

    var $options;

    function Select($items){

        parent::check_required($items, get_class($this));

        //constructor of parent class
        $this->FormItem($items['param'], $items['name'], $items['label']);

        if(isset($items['required']))
            $this->required = true;

        foreach($items->option as $option)
            $this->options[] = new Option($option, $option['value']);

    }

    function toString(){
        $compose = "";
        $req = ($this->required) ? "<font color='red'>*</font>" : "";

        $compose .= "\n<span class='lbls'>".$this->label.$req."</span>\n
                    <select name='".$this->name."' id='".$this->name."'>\n";

        $name= $this->name;
        foreach($this->options as $option){
            $val = $option->value;
            $checked = (isset($_POST["{$name}"]) and
                                $_POST["{$name}"] == $val) ? "selected='selected'" : "";
            $compose .= "\t<option value='$val' $checked />".$option->label."</option>\n";
        }
        $compose .= "</select>\n";

        return $compose;
    }

}
```

Obrázek 14: Třída prvku Select

Po zpracování formuláře a výsledků programu jsou vypsány položky historie aktuálního sezení (není-li použita volba „show all history items“) řazené dle data sestupně.

5 Závěr

Cílem této bakalářské práce bylo navržení a vytvoření aplikace pro automatické testování simulací z programu `Ferrodo` a navržení a vytvoření rozhraní pro obsluhu programů pro další zpracování simulací.

Oddíl práce věnovaný automatickému testování splňuje všechny požadavky určené zadáním. V průběhu vývoje bylo často nutné reagovat na některé aktuální požadavky uživatelů aplikace. Webová aplikace se používá k včasnému a adresnému odhalování chyb v programu `Ferrodo`, a to už od prvních vyvinutých verzí.

Druhá část práce, to jest rozhraní programů pro postprocesing výsledků programu `Ferrodo` taktéž splňuje všechny body zadání. Rozhraní je rozšiřitelné pro téměř libovolný konzolový program pomocí jednoduchého XML konfiguračního souboru. Vytvořením vlastního generátoru pro zpracování XML souboru a následnou obsluhu programu jsem, troufnu si říci, přesáhl rámec zadání a aplikace by po několika vylepšeních mohla být využívána i mimo program `Ferrodo`.

Jako nedostatek vidím nepříliš elegantně vyřešenou volbu souborů v druhé části práce. Vždy je zobrazována možnost volby osmi souborů, přestože dané programy mohou pracovat s méně soubory, a dokonce nemusí využívat soubory žádné. Přítomnost zbytečných voleb znepřehledňuje ovládání programu.

Při dalším vývoje aplikace by bylo vhodné formulář pro volbu souborů přepracovat do více přizpůsobivé podoby.

Seznam použité literatury

[1] ONDREJKOVIČ, Petr. *Studium doménových struktur ve feroelektrickém $BaTiO_3$.* Praha: ČVUT, 2008

[2] MANDAU, Markus. *Největší softwarové katastrofy*. [online]. Dostupné z: http://earchiv.chip.cz/cs/earchiv/rubriky/zaverem/nejvetsi-sw-katastrofy.html

[3] PATTON, Ron. *Testování softwaru*. Vyd. 1. Praha: Computer Press, 2002, xiv, 313 s. Programování. ISBN 80-722-6636-5.

[4] KUČEROVÁ, Helena. *Specifikace požadavků na informační systém* [online]. 2012 [cit. 2013-05-08]. Dostupné z: http://web.sks.cz/users/ku/pri/specifik.htm

[5] SMARTBEAR. *Automation Testing - TestComplete* [online]. [cit. 2013-05-13]. Dostupné z: http://smartbear.com/products/qa-tools/automated-testing-tools

[6] HP. *HP Quality Center Software* [online]. [cit. 2013-05-13]. Dostupné z: http://www.hp.com/go/qualitycenter

[7] VÁCLAVÍK, Jiří. *Linux v příkazech - porovnávání souborů*. [online]. 2006. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=1136

[8] JAN BŘEZINA: S*kript ndiff, vyvinut jako součást programového balíku Flow123d, NTI, TUL*

[9] *LEVKOWETZ, Henrik. Rfcdiff Tool. [online]. Dostupné z: http://tools.ietf.org/tools/rfcdiff/*

[10] NETCRAFT. *March 2013 Web Server Survey* [online]. Dostupné z: http://news.netcraft.com/archives/2013/03/01/march-2013-web-server-survey.html

[11] WEBTVORBA. *Úvod do PHP.* [online]. Dostupné z: http://www.webtvorba.cz/php/uvod-do-php.html

[12] PHP.NET. *Description of core php.ini directives*. [online]. 2013. Dostupné z: http://www.php.net/manual/en/ini.core.php

[13] ZORN, Walter. *High Performance JavaScript Vector Graphics Library.* [online]. Dostupné z: http://www.walterzorn.de/en/jsgraphics/jsgraphics_e.htm

[14] NEKVASIL, Michal. *Systém automatického sestavování a testů pro simulátor proudění Flow123d.* Liberec: TUL, FM, 2011

Příloha A

Obsah přiloženého CD

- `Testovani` – adresář obsahující zdrojové kód aplikace pro testování programu Ferrodo včetně některých sad testů a referencí
- `Rozhrani` – adresář se zdrojovými kódy aplikace pro obsluhu konzolových programů. Obsahuje také rozhraní pro program `Manipulate` a konfigurační soubor pro program `Fview`
- `Hrdina_bp.pdf` – elektronická verze bakalářské práce

Příloha B

Návod k vytvoření XML konfiguračního souboru

XML soubor obsahuje elementy (v našem případě formulářové prvky a další nastavení), které jsou vyznačeny pomocí počátečního a ukončovacího tagu.

Příklad: `<nadpis>Text nadpisu</nadpis>`

Element může obsahovat atributy, které upřesňují význam elementu. Atributy se uvádějí v počátečním tagu a vždy musí mít nějakou hodnotu uzavřenou v uvozovkách.

Příklad:

```
<kniha autor="Jan Nový" rok="2000">Tajemství</kniha>
```

Výsledný XML soubor je nutné umístit do adresáře `modules`.

Vytvářený modul je uzavřen v kořenovém elementu **module**. Element obsahuje kromě formulářových polí tři povinné a dva nepovinné elementy, a to:

name

Obsahuje text zobrazovaný jako nadpis rozhraní.
Příklad: <name>Fview 2.0</name>

software

Obsahuje cestu ke skriptu ve složce `software` a může přebírat dva atributy:

- `system` s libovolnou hodnotou určuje, že se jedná o program spustitelný bez zadání cesty (tj. cesta k programu je v systémové proměnné PATH, typicky systémové programy jako ping, ifconfig, apod.).

- `command` – zajistí spuštění programu pomocí daného příkazu. (například „python“ pro python skripty)

output

Určuje typ výstupu, neobsahuje nic.

- `type` – povinný atribut. Hodnota „files“ znamená, že výstupem budou soubory, hodnota „stdout“ znamená textový výstup na konzolu.
- `show` – pouze pro textový výstup. Pokud je atribut nastaven, bude výstup vypsán na obrazovku, v opačném případě bude přítomen pouze odkaz na soubor s výstupem.
- `copytodata` – určuje zda mají být výstupní soubory kopírovány do složky s daty.
- `scaleimages` – určuje, zda mají být obrázky výstupu zmenšovány na velikost maximálně 500 pixelů na šířku.

help

Nepovinný element. Hodnota atributu `param` určí, který parametr programu vypíše nápovědu.

description

Nepovinný element. Hodnota elementu je vypsána menším písmem pod názvem modulu a může sloužit jako krátká nápověda nebo popis programu.

Formulářové prvky

Jednotlivé elementy reprezentující formulářové prvky mají čtyři společné atributy.

- `required` – atribut s libovolnou hodnotou nastaví prvek jako povinný, tj. musí být vždy vyplněn
- `name` – hodnota je jednoslovný unikátní identifikátor prvku
- `label` – název prvku vypisovaný ve formuláři
- `param` – hodnota určuje parametr předávaný programu

File

Element reprezentující soubor. Ve formuláři není viditelný a určujeme jím, který z 8 souborů z formuláře pro výběr souborů bude použit a jak.

Parametry:

- `usefile` – hodnota z intervalu „file1“ až „file8“ určuje, který soubor bude použit
- `noparam` – při nastavení tohoto parametru bude soubor předán jako argument bez parametru

Text

Textové pole. Obsah elementu bude použit jako výchozí text. Nepřebírá žádné další atributy

Number

HTML5 prvek, který se v nekompatibilních prohlížečích tváří jako obyčejné textové pole. V kompatibilních prohlížečích umožňuje zadat pouze číslo (i s plovoucí desetinnou čárkou) a obsahuje šipky pro zvýšení a snížení hodnoty o nastavený krok

Parametry:

- `min` – nejmenší přípustná hodnota
- `max` – největší přípustná hodnota
- `step` – krok zvyšování/snižování (výchozí hodnota je 1)

Option

Volba. Nejedná se o samostatný element. Elementy `option` jsou vkládány do elementů `Select`, `Radio` a `Checkbox`. Obsah elementu určuje text volby.

Parametry:

- `value` – určuje hodnotu parametru
- `checked` – atribut s libovolnou hodnotou. Při jeho nastavení bude volba předem vybrána

Radio

Přepínače. Element obsahuje jednotlivé volby (elementy `option`).

Select

Rozbalovací seznam. Stejně jako `Radio` obsahuje pouze jednotlivé volby(elementy `option`).

Checkbox

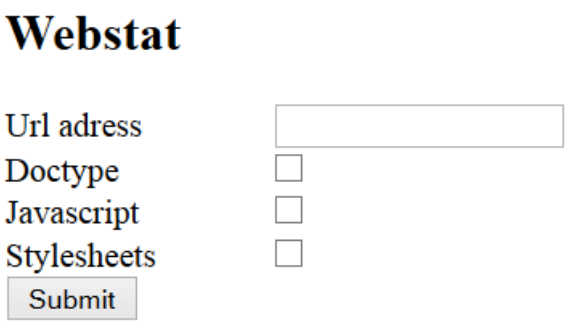
Zaškrtávací políčka. Mají více možností využití. Pokud mají pouze jednu možnost (`option`) můžeme využít atribut `usefile`, čímž můžeme použít již vybraný soubor s jiným parametrem, nebo atribut `noargs`, kterým určíme, že bude v příkazu použit pouze atribut bez argumentů. Při použití více možností atributem `delimiter` určíme spojovací výraz (výchozí je čárka).

Příklad 1:

XML konfigurační soubor:

```
<module>
	<name>Webstat</name>
	<software command="python">webstat/webstat.py</software>
	<text param="u" name="url" label="Url adress"></text>
	<checkbox param="d" noargs="1" name="doctype" label="Doctype">
		<option value="1" />
	</checkbox>
	<checkbox param="j" noargs="1" name="js" label="Javascript">
		<option value="1" />
	</checkbox>
	<checkbox param="c" noargs="1" name="css" label="Stylesheets">
		<option value="1" />
	</checkbox>
	<output type="stdout" show="1"></output>
</module>
```

HTML výstup:



Při adrese „http://seznam.cz“ a zaškrtnutých políčkách „Doctype“ a „Javascript“ bude spouštěný následující příkaz:

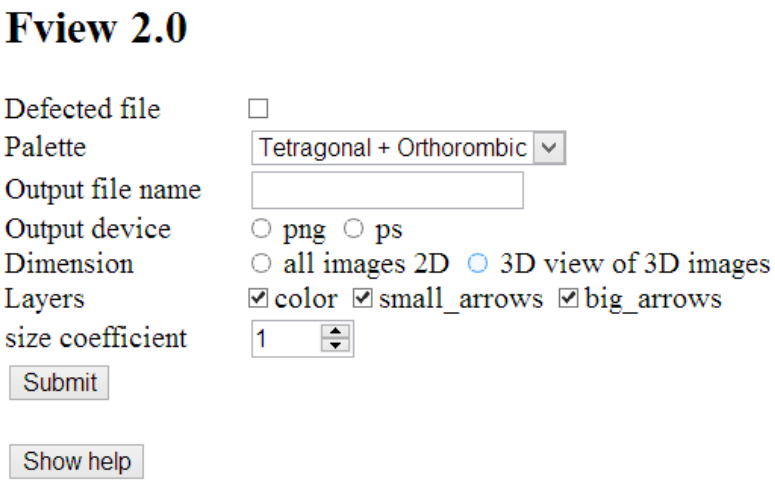
```
python ./software/webstat/webstat.py -u "http://seznam.cz" -d -c > output.txt
```

Příklad 2:

XML konfigurační soubor pro program `Fview`. Tento modul je v aplikaci již vytvořen a slouží pro dodatečné generování vizualizace vektorového pole. Vstupní soubor s parametrem `i` je určen pomocí „file1“ z formuláře pro výběr souborů. Pokud ovšem zatrhneme tlačítko „Defected file“ bude soubor použit s parametrem `t`. Dále se jedná o standardní formulářové prvky. V případě zaškrtnutí více položek u checkboxů „layers“ budou jednotlivé hodnoty odděleny svislicí.

```
<module>
	<name>Fview 2.0</name>
	<software>fview_2.0/fview_2.0_static</software>
	<help param="h"></help>
	<file required="1" param="i" label="input file" name="inputfile" usefile="file1" />
	<checkbox param="t" name="def" label="Defected file" usefile="file1">
		<option value="1"></option>
	</checkbox>
	<select name="palette" label="Palette" param="palette">
		<option value="0">Tetragonal + Orthorombic</option>
		<option value="1">Tetragonal</option>
		<option value="2">Orthorombic</option>
	</select>
	<text param="o" name="outfile" label="Output file name"></text>
	<radio param="d" name="device" label="Output device">
		<option value="png">png</option>
		<option value="ps">ps</option>
	</radio>
	<radio param="m" name="dimension" label="Dimension">
		<option value="2d">all images 2D</option>
		<option value="3d">3D view of 3D images</option>
	</radio>
	<checkbox delimiter="|" param="l" name="layers" label="Layers">
		<option value="color" checked="1">color</option>
		<option value="s_arrows" checked="1">small_arrows</option>
		<option value="b_arrows" checked="1">big_arrows</option>
	</checkbox>
	<number param="s" min='1' max='5' step="1" name="size" label="size coefficient">1</number>
	<output copytodata="1" type="files"></output>
</module>
```

HTML výstup:



Spouštěný příkaz může vypadat takto:

```
./software/fview_2.0/fview_2.0_static -i tmp/p__0140000.arr
--palette 0 -m 2d -l "color|s_arrows|b_arrows" -s 1
```