VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Absolvovanie individuálnej odbornej praxe

Individual professional practice in the company

2016 **Jozef Vojtek**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Jozef Vojtek**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: TELE DATA SYSTEM, spol.s.r.o.
2. Struktura závěrečné zprávy:
a. Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta
b. Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti
c. Zvolený postup řešení zadaných úkolů
d. Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe
e. Znalosti či dovednosti scházející studentovi v průběhu odborné praxe
f. Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vedl odbornou praxi studenta

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeňka Chmelíková, Ph.D.**

Konzultant bakalářské práce: Ing. Ivo Hajduček

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

VYSOKÁ ŠKOLA BÁŇSKÁ - TECHNICKÁ UNIVERZITA * OSTRAVA * Fakulta elektrotechniky a informatiky

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlásenie študenta

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave dňa: *27. apríla 2016*

……..……
podpis študenta

Poďakovanie

Rád by som poďakoval Ing. Zdenke Chmelíkovej, PhD. a Ing. Ivovi Hajdučekovi za odbornú pomoc a konzultáciu pri vytváraní tejto bakalárskej práce. Ďalej by som rád poďakoval spoločnosti TELE DATA SYSTEM, spol. s.r.o. za možnosť absolvovať odbornú prax u nich.

Prehlásenie zástupcu spolupracujúcej právnickej alebo fyzickej osoby

„Súhlasím so zverejnením tejto bakalárskej práce podľa požiadaviek čl. 26, ods. 9 Študijného a skúšobného poriadku pre štúdium v bakalárskych/magisterských programoch VŠB-TU Ostrava."

Dňa: *21. apríla 2016*

.....

podpis zástupcu

TDS TELE DATA SYSTÉM spol. s r.o. 28. října 1418/125, 702 00 Mor. Ostrava DIČ: CZ18051286

Abstrakt

Táto práca popisuje priebeh individuálnej odbornej praxe absolvovanej v spoločnosti TELE DATA SYSTEM, spol. s.r.o.. Popisujem tu odborné zameranie firmy a ich produkty a práve na jednom z nich som pracoval počas celého trvania odbornej praxe. Ide o SCADA software IDS HIGH-LEIT™. Ďalej opisujem technológie a nástroje, ktoré som používal pri práci. Zameriavam sa na konkrétne úlohy a projekty, ktoré som v rámci údržby a rozširovania softwaru IDS HIGH-LEIT™ vykonával. A v neposlednej rade uvádzam prínosy bakalárskej práce a využitie mojich riešení v softwari IDS HIGH-LEIT™.

Kľúčové slová

TELE DATA SYSTEM; spol.; s.r.o.; odborná prax; programovanie; SCADA; IDS HIGH-LEIT™; Java;

Abstract

This thesis describe course of individual professional practice in the company TELE DATA SYSTEM, spol. s.r.o.. I describe specialism of this company and their products. I worked on one of company's products for the duration of professional practice. This product was the SCADA software IDS HIGH-LEIT™. It further describes technologies and tools, which I used at work. I focus on concrete tasks and project, which I done for the maintenance and extension of software IDS HIGH-LEIT™. Last but not least I submit benefits of professional practice and usage of my solutions in the software IDS HIGH-LEIT™.

Key words

TELE DATA SYSTEM; spol.; s.r.o.; professional practice; programing; SCADA; IDS HIGH-LEIT™; Java;

Zoznam použitých skratiek

Skratka	Význam
WWW	World Wide Web
HTML	Hyper Text Markup Language
IDE	Integrated Development Enviroment
SDK	Software Development Kit
JAR	Java ARchive
XML	Extensible Markup Language
RGB	Red Green Blue
GUI	Graphic User Interface
tj.	To jest
atď.	A tak ďalej
Obr.	Obrázok
napr.	Napríklad

Obsah

Úvod

Cieľom tejto bakalárskej práce je popísať priebeh odbornej praxe, ktorú som absolvoval vo firme TELE DATA SYSTEM, spol. s.r.o.. Práca je rozdelená do štyroch kapitol. V prvej kapitole predstavujem firmu TELE DATA SYSTEM, spol. s.r.o. a svoje pracovné zaradenie. Druhá kapitola sa venuje popisu nástrojov a technológií, použitých pri vykonávaní odbornej praxe. V tretej časti sa venujem konkrétnym úlohám a projektu, na ktorých som pracoval, predstavujem zadanie a následné riešenie. Posledná kapitola je venovaná získaným a chýbajúcim znalostiam počas vykonávania odbornej praxe. Na záver zhodnocujem výsledky bakalárskej práce a moje pôsobenie vo firme.

1 Predstavenie firmy a pracovného zaradenia

1.1 TELE DATA SYSTEM, spol. s.r.o.

Firma pôsobí na českom trhu od roku 1992. Od roku 2004 je väčšinovým majiteľom nemecká firma IDS GmbH. TELE DATA SYSTEM, spol. s.r.o. sa špecializuje na komplexnú dodávku a realizáciu telemetrických, dispečerských a riadiacich systémov v plynárenstve, vodohospodárstve a energetike.

TELE DATA SYSTEM, spol. s.r.o. má v ponuke široké spektrum produktov ako sú SCADA systémy, telemetrické a batériové stanice, informačné systémy a softwarové aplikácie. Ich produkty využívajú napr. Ústredná čistiareň odpadových vôd Ostrava, čistiarne odpadových vôd v Albrechticiach, Opave a Sviadnove. Severomoravská plynárenská a.s. využíva systém pre sledovanie tlakových hladín miest (THM) a dispečerský riadiaci systém pre dispečing vysokotlakovej siete. V jadrovej elektrárni Dukovany je uvedená v prevádzke významná aplikácia z oblasti energetiky a to "Skupinová regulácia jalového výkonu v uzle Slavětice" (ASRU).

V roku 2007 firma získala certifikáciu Systému integrovaného managementu podľa noriem ČSN EN ISO 90011:2001 a ČSN EN ISO 14001:2005.

1.2 IDS GmbH

IDS GmbH je nemecká spoločnosť založená v roku 1975, so sídlom v meste Ettlingen. Pobočky tejto firmy sú po celej Európe a aj v zámorí.

IDS GmbH je špecialista na riadenie bezpečnostného managementu distribučných sietí ako sú kontrolné systémy, automatizácia, SCADA systémy, telemetria a komunikačné systémy. V ponuke produktov firmy je SCADA systém IDS HIGH-LEIT™, určený na aplikáciu v energetike a vodohospodárstve. Je to otvorený systém s klient - server architektúrou, ktorý okrem bežných SCADA funkcií ponúka rôzne špecifické funkčné moduly. IDS HIGH-LEIT™ je flexibilný, modulárny systém, je voľne škálovateľný s ohľadom na rozsah funkcií, dátový model a hardvérovú konfiguráciu.

Počas celého trvania mojej odbornej praxe som pracoval na rozširovaní a údržbe systému IDS HIGH-LEIT™.

1.3 Popis pracovného zaradenia

Do spoločnosti som bol prijatý po úspešnom prijímacom pohovore na pozíciu "Java programátor". Bol som zaradený do týmu pod vedením pána Ing. Iva Hajdučeka. Tento tým sa zaoberá rozširovaním a údržbou SCADA systému IDS HIGH-LEIT™.

Moja úloha spočívala v riešení a odstraňovaní chýb systému IDS HIGH-LEIT™ a neskôr vo vytváraní rozširujúceho nástroja na porovnávanie užívateľských modelových súborov systému IDS HIGH-LEIT™.

2 Použité technológie a nástroje

2.1 Java

Java [1] je objektovo orientovaný programovací jazyk, ktorého základy môžeme nájsť v projekte Oak, ktorý vznikol vo firme Sun na počiatku deväťdesiatych rokov pre riadenie elektronických výrobkov. V roku 1994 bol prenesený ako programovací jazyk do prostredia počítačov pod názvom Java.

Z počiatku sa Java využívala hlavne v plne internetových aplikáciách, ktoré sprostredkuvávajú komunikáciu medzi klientom na internetovom prehliadači a službami prístupnými cez internetový server. Tu sa využili výhody Javy, jej robustnosť, stabilita, rozsah funkcií a hlavne bezpečnosť. Aj preto podporu Java appletov nájdeme v najväčšej miere na WWW stránkach internetových bankovníctiev a ďalších aplikáciach, vyžadujúcich vysokú mieru stability a zabezpečenia.

Výhodou Javy je jej multiplatformita, deklarovaná heslom: "Write once run everywhere.". Zdrojový kód je pri vývoji preložený do spustiteľného medzi kódu (bytecode), ktorý je potom možné spúšťať pomocou nainštalovaného runtime prostredia (Java Virtual Machine), priamo na rôznych typoch počítačov alebo technických zariadeniach bez nutnosti nového prekladu.

Pretože však táto prenositeľnosť nie je a nemôže byť stopercentná (hlavne z technických príčin rozhrania), vyvinulo sa niekoľko edícií Javy, líšia sa drobnými rozdielmi a rozšíreniami, spojených spoločným jazykom a veľkou skupinou knižníc.

Medzi hlavné platformy Javy je v súčasnej dobe možné uviesť:

- J2SE (Java 2 Standart Edition) - spustiteľné aplikácie na počítačoch
- Applety - programy v jazyku Java, ktoré sa sťahujú a púšťajú spoločne s WWW stránkou
- J2EE (Java 2 Enterprise Edition) - servlety, webové služby na WWW serveroch, generujúce dynamické HTML stránky
- JSP (Java Server Pages) - umožňuje kombinovať v jednom WWW dokumente HTML kód a príkazy Javy
- J2ME (Java 2 Micro Edition) - spustiteľné aplikácie na mobilných zariadeniach

2.2 Eclipse IDE

Eclipse IDE [2] je integrované vývojové prostredie a programátorský nástroj. Obsahuje základné pracovné prostredie a rozširujúci plug-in systém pre vlastnú úpravu prostredia. Eclipse IDE je napísaný prevažne v Jave a jeho primárne určenie je na vývoj Java aplikácií, ale taktiež môže byť použitý na vývoj aplikácií aj v iných programovacích jazykoch.

Užívateľ môže rozširovať možnosti Eclipse IDE nainštalovaním plugin-ov napísaných pre Eclipse Platform, ako sú vývojové toolkit-y pre rôzne programovacie jazyky a taktiež môže vyvíjať a používať vlastné plug-in moduly.

Spoločnosť Eclipse Foundation vydáva svoj software pod Eclipse Public License (je voľne využiteľný). Eclipse SDK je taktiež zadarmo a open-source.

2.3 SCADA

SCADA [3] je priemyselný automatizačný kontrolný systém a tvorí jadro mnohých priemyselných odvetví vrátane:

- energetiky,
- potravinárstva,
- ropného priemyslu,
- dopravy,
- vodohospodárstva,
- odpadového hospodárstva,
- a mnoho ďalších.

SCADA systémy v sebe združuje niekoľko softwarových a hardwarových elementov, ktoré umožňujú organizáciu v priemysle tj.:

- monitorovanie, zber a spracovanie dát,
- interakcia a kontrola strojov a zariadení ako sú ventily, pumpy, motory a ďalšie, ktoré sú pripojené cez HMI (human-machine interface) software,
- záznam udalostí do log súboru.

Základná SCADA architektúra spočíva v tom, že informácie zo senzorov alebo manuálnych vstupov sú posielané na PLC (programmable logic controller) alebo RTU (remote terminal unit) jednotky, ktoré následne posielajú tieto informácie do počítačov so SCADA softwarom. SCADA software analyzuje a zobrazuje zoradené dáta, ktoré pomáhajú operátorom a ďalším pracovníkom redukovať odpad a zvyšovať efektivitu výrobných procesov. Vo výsledku efektívny SCADA systém pomáha k výraznej úspore času a peňazí.

2.4 SWT - JFace

SWT [4] je open-source widget toolkit firmou Eclipse Foundation pre vývoj aplikácií v jazyku Java, navrhnutý aby efektívne a prenositeľne pristupoval k užívateľskému rozhraniu operačného systému, na ktorom je implementovaný. Ponúka široké spektrum widget-ov, ktorých použitie uľahčuje následný vývoj aplikácií.

Je distribuovaný ako voľne stiahnuteľný plug-in modul pre Eclipse IDE.

2.5 JUnit

JUnit [5] je Java knižnica, ktorá nám pomáha pri jednotkovom testovaní. Jednotkové testovanie je proces skúmania malej "jednotky" softwaru (zvyčajne jedinej triedy) za účelom overenia očakávanej funkčnosti alebo špecifikácie.

JUnit nie je súčasťou štandardnej Java knižnice, ale dá sa získať ako plugin pre Eclipse IDE.

2.6 Apache Maven

Apache Maven [6] je software pre riadenie a spoluprácu na Java projektoch. Software je založený na koncepte POM (project object model). Maven môže riadiť zostavenie Java projektu,

dokumentáciu, zdieľanie JAR súborov naprieč rôznymi projektmi, hlásenie udalostí z centrálneho zdroja informácií.

2.7 SVN

SVN [7] Apache Subversion je systém na verzovanie, revíziu a kontrolu softwaru distribuovaný pod open-source licenciou. Subversion bol vytvorený spoločnosťou CollabNet Inc. v roku 2000, ale dnes je vyvíjaný ako projekt spoločnosti Apache Software Foundation.

2.8 Atlassian JIRA

Atlassian JIRA [8] je aplikácia designovaná pre podporu vývojárskych tímov počas všetkých fáz dodávky a prevádzky softwarových riešení. Dnes však už kompletne zahŕňa efektívne riadenie a sledovanie úloh a požiadaviek v projekte (task and project management). Software JIRA podporuje a uľahčuje proces riadenia projektov a požiadaviek - ponúka flexibilný a užívateľský nástroj pre riadenie a sledovanie pracovníkov pri plnení úloh.

2.9 Scrum metodika

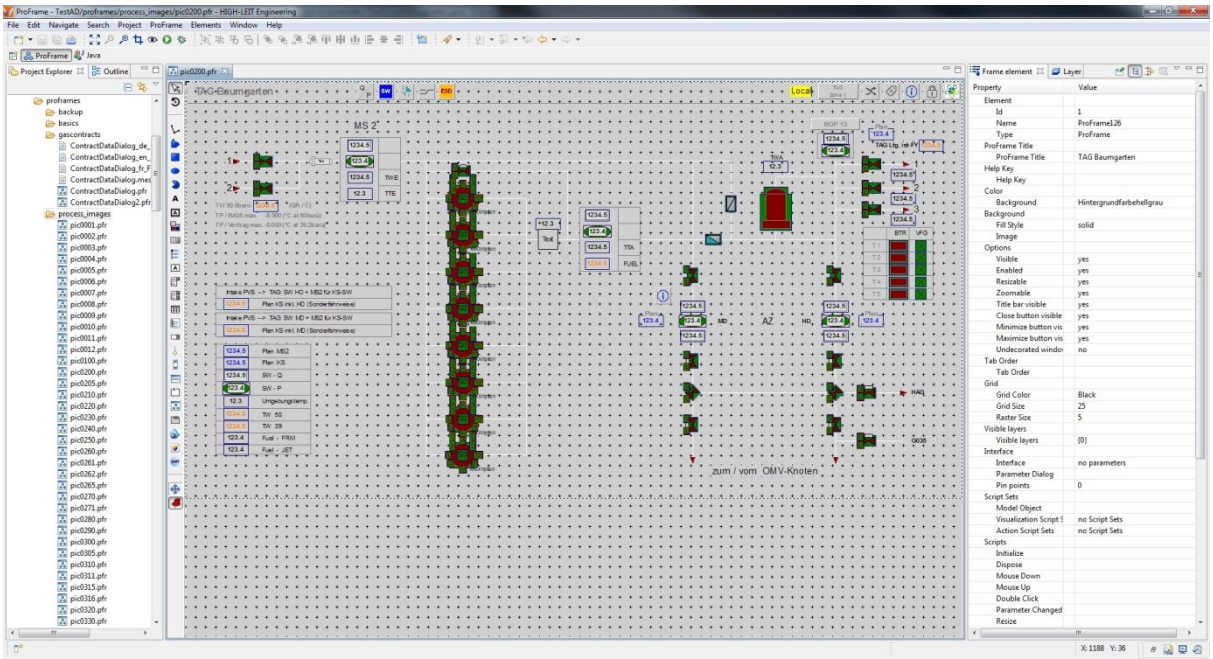
Scrum metodika [9] je založená na empirickej teórií riadenia procesov. Táto teória tvrdí, že poznanie pochádza zo skúseností a rozhodovanie je založené na tom, čo je známe. Scrum metodika zastáva iteratívny a inkrementálny prístup k optimalizácií predvídateľnosti a riadenia rizika, ktorý stojí na troch pilieroch a to: transparentnosť, kontrola a adaptácia.

- **Transparentnosť** - významné aspekty procesu musia byť viditeľné osobám zodpovedným za výsledok. Transparentnosť si vyžaduje definovanie spoločného štandardu, napr. je presne definované kedy je proces v stave "dokončený".
- **Kontrola** - užívatelia Scrum metodiky musia často kontrolovať Scrum artefakty a pokrok smerom k cieľu, aby odhalili prípadne problémy a odchýlky. Kontroly by ale nemali byť až tak časté, aby sa nedostávali do konfliktu s prácou.
- **Adaptácia** - v prípade, že sa pri kontrole zistí vybočenie procesu z prijateľných medzi, proces musí byť upravený. Úprava musí byť vykonaná čo najskôr, aby sa minimalizovali ďalšie odchýlky.

3 Práca na firemných projektoch

Ako som už uviedol v úvode, počas trvania mojej odbornej praxe som pracoval v týme Ing. Iva Hajdučeka na SCADA softwari IDS HIGH-LEIT™. Pôvodný SCADA software IDS HIGH-LEIT™ bol napísaný v jazyku C, ale firma sa rozhodla prekonvertovať software do jazyka Java. V pôvodnom software sa používal interný formát ProBild™ a v konvertovanom software sa používa interný formát ProFrame™. Implementácia v rôznych jazykoch viedla k rozdielom vo funkčnosti medzi pôvodnou a konvertovanou verziou softwaru. Spoločne s týmom Ing. Iva Hajdučeka som pracoval na tvorbe riešení pre odstránenie rozdielov, prípadne chýb vo funkčnosti softwaru IDS HIGH-LEIT™.

Obrázek 1.1: *IDS HIGH-LEIT™*



Obrázek 1.2: *IDS HIGH-LEIT™ pracovné prostredie*

Počas môjho pôsobenia vo firme som pracoval asi na ôsmych menších úlohách a na jednom hlavnom rozsiahlejšom projekte. V nasledujúcich odstavcoch popíšem riešenie vybraných troch menších úloh a hlavného projektu.

3.1 Nástroj na hľadanie najbližšej farby

Konvertovaný formát ProFrame™ softwaru IDS HIGH-LEIT™ ponúkal užívateľovi na výber väčšie spektrum farieb ako pôvodný formát ProBild™. Pri spätnej konverzií, tj. z ProFrame™ na ProBild™, boli statické ProFarme™ farby prevedené pomocou mapovania reťazca na indexy definované v mapovacom XML súbore vytvoreného z pôvodných ProBild™ farieb. Problém nastal, keď názov určitej ProFrame™ farby nebol obsiahnutý v mapovacom XML súbore. Preto bolo potrebné implementovať nástroj, ktorý z pôvodných farieb z mapovacieho XML súboru vyberie najbližšiu možnú farbu na základe RGB hodnôt.

Najskôr som začal skúmať možnosti a hľadať na Internete nejaké už overené algoritmy na porovnávanie farieb. Prvý adept bola implementácia formule na výpočet rozdielu farieb CIEDE94. Napriek odbornosti a zložitosti algoritmu formule CIEDE94, výsledná najbližšia farba sa nezhodovala s očakávanou farbou.

Po ďalšom hľadaní sa mi podarilo nájsť jednoduchý algoritmus, ktorý spočítal absolútnu hodnotu rozdielu jednotlivých zložiek RGB dvoch farieb, požadovanej a porovnávanej. Výsledný rozdiel bola najväčšia hodnota z vypočítaných absolútnych hodnôt jednotlivých RGB zložiek.

Implementácia porovnávacieho algoritmu vyzerala nasledovne.

ColorMatcherImpl.java

```
private double getDifference(RGB color, RGB wantedColor) {
    double diffR = Math.abs(wantedColor.red - color.red);
    double diffG = Math.abs(wantedColor.green - color.green);
    double diffB = Math.abs(wantedColor.blue - color.blue);
    double difference = Math.max(diffR, (Math.max(diffG, diffB)));
    return difference;
}
```

Metóda *getLowestDifference* hodnotu RGB požadovanej farby porovnala s RGB hodnotami získanými z XML súboru a uloženými v mape. RGB hodnota, ktorá získala po prechode algoritmom implementovanom v metóde *getDifference* najmenšiu hodnotu rozdielu bola najbližšia farba k hľadanej farbe a metóda vrátila jej index.

ColorMatcherImpl.java

```
private Integer getLowestDifference(RGB wantedColor, Map<Integer,
RGB> colorMap) {
    double minDif = Double.MAX_VALUE;
    int bestMatch = 0;
```

```
    while (entries.hasNext()) {
      Entry<Integer, RGB> entry = entries.next();
      double difference = getDifference(entry.getValue(),
wantedColor);
      if (minDif > difference) {
        minDif = difference;
        bestMatch = entry.getKey();
      }
    }
    return bestMatch;
}
```

Správnu funkčnosť a výsledky som testoval pomocou testovacej triedy a nástroja JUnit. Výsledné farby sa stopercentne zhodovali s očakávanými farbami.

Ukážka jednej testovacej metódy, ktorá testuje zhodu pre určitú RGB hodnotu a výsledok porovnáva s nami očakávanou RGB hodnotou z mapy.

ColorMatcherTest.java

```
@Test
  public void testFind() {
    Map<Integer, RGB> colorMap = createColorMap();
    RGB wantedColor = new RGB(78, 38, 238);
    int bestMatch = colorMatcher.findBestMatch(wantedColor,
colorMap);
    assertEquals(1, bestMatch);
  }
```

3.2 Test konvertora textových objektov z ProFrame™ do ProBild™ formátu

Mojou úlohou bolo otestovať správnu funkčnosť konvertora, tj. či sú vlastnosti konvertovaného ProBild™ textového objektu zhodné s vlastnosťami pôvodného ProFrame™ textového objektu. V zadaní bolo presne uvedené, ktoré vlastnosti je potrebné otestovať.

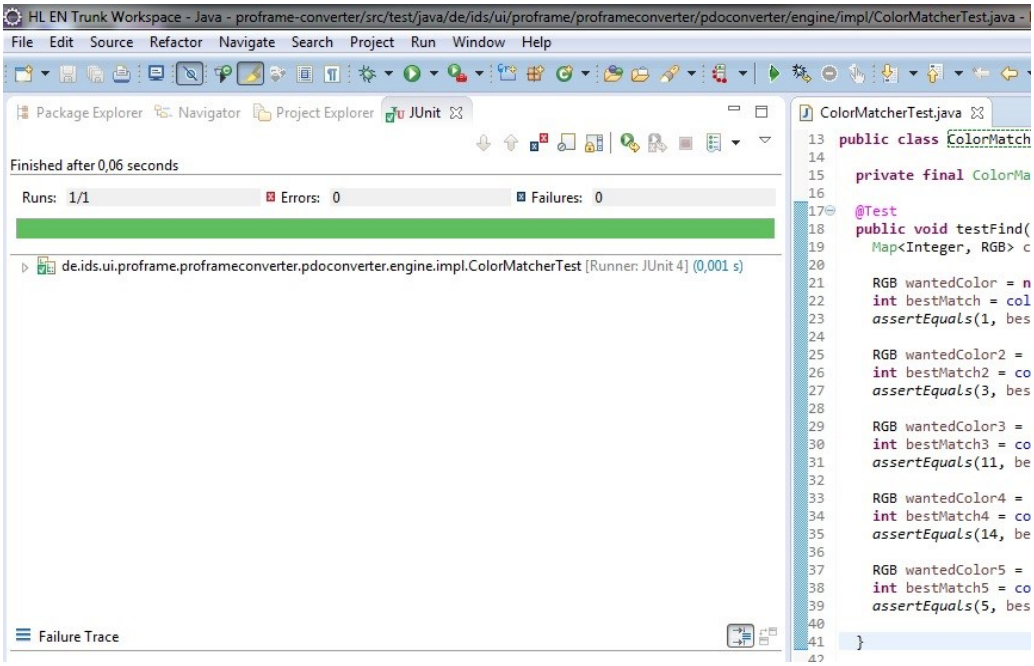
Vytvoril som testovaciu triedu, v nej som deklaroval textový objekt formátu ProFrame™ a pomocou metód som mu nastavil potrebné vlastnosti. Následne som použil nástroj JUnit a vytvoril som testovacie metódy, v ktorých som porovnal výstup z konvertora s očakávanou hodnotou.

Príklad jednej testovacej metódy, ktorá nastavila objektu *PdoText* vlastnosti text, font a možnosť označovať text. Následne sa objekt *PdoText* prekonvertoval na objekt *DatText*. Pomocou metód *assertEquals* a *assertTrue* sa otestovali vlastnosti konvertovaného objektu *DatText*.

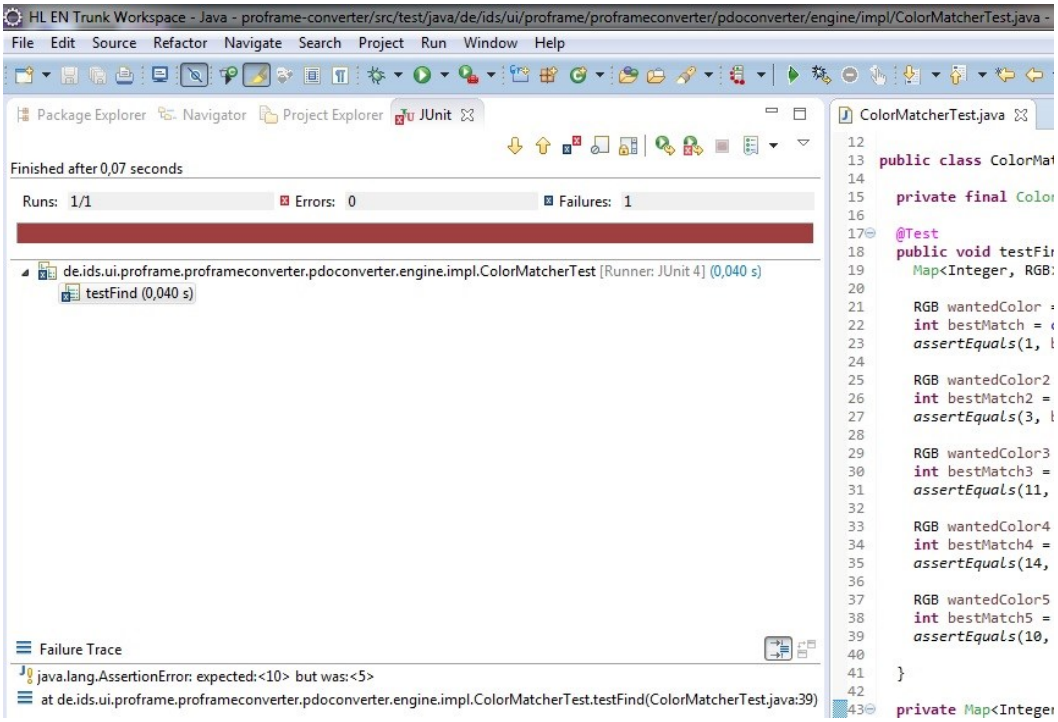
PdoTextConverterTest.class

```
@Test
  public void testConvertSimpleText() {
    pdoText.setText("hello");
    pdoText.setSelectable(true);
    pdoText.setFont(new Font("helvetica", 16, false, true));

    DatObject datObject = TestTools.convert(converter, pdoText,
null);

    assertEquals(DatText.class, datObject.getClass());
    DatText datText = (DatText) datObject;

    assertEquals("hello", datText.getParTxt().getText());
    assertEquals(FontSize.Type.POINT,
datText.getParFnt().getSize().type);
    assertEquals(16, datText.getParFnt().getSize().points);
    assertTrue(datText.getParOpt().isSelectable());
  }
```

Po skompilovaní testovacej triedy sa spracovali všetky testovacie metódy a na výstupe sa zobrazil výsledok JUnit testov. V prípade neúspechu testu, nástroj JUnit zobrazil hlásenie s hodnotou výsledku a s číslom riadku neúspešného testu.



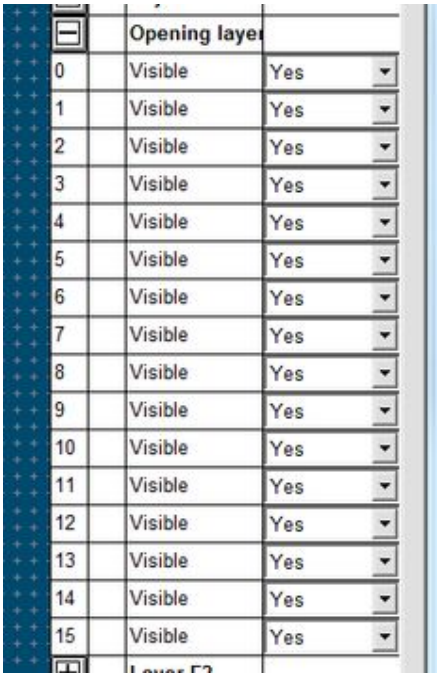
Obrázek 1.3: *JUnit výstup - úspech*



Obrázek 1.4: *JUnit výstup - neúspech*

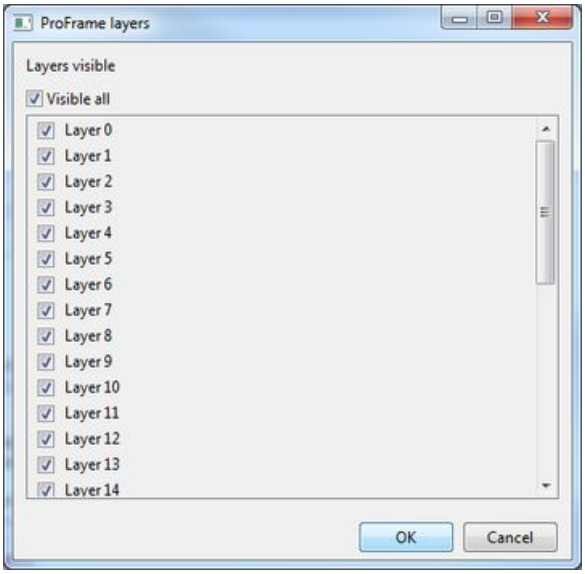
3.3 Dialóg pre nastavenie viditeľnosti vrstiev v ProFrame™ editore

Editor pôvodného softwaru IDS HEIGH-LEIT™ obsahovalo funkciu nastavenia viditeľnosti jednotlivých vrstiev, vrstvy sa zobrazovali podľa hĺbky priblíženia a užívateľ si mohol pomocou funkcie nastaviť ich viditeľnosť. Editor obsahoval 32 vrstiev. Táto funkcia v konvertovanom softwari IDS HIGH-LEIT™ chýbala.



Obrázek 1.5: *Editor z pôvodného softwaru*

Ako prvé bolo potrebné vytvoriť návrh dialógu, v ktorom sa bude funkcia nastavenia vrstiev používať. Konvertovaný software IDS HIGH-LEIT™ používal komponenty užívateľského rozhrania z frameworku SWT - JFace a už obsahoval podobné dialógy. Pre zachovanie jednotného dizajnu som ako základ použil jeden z nich.



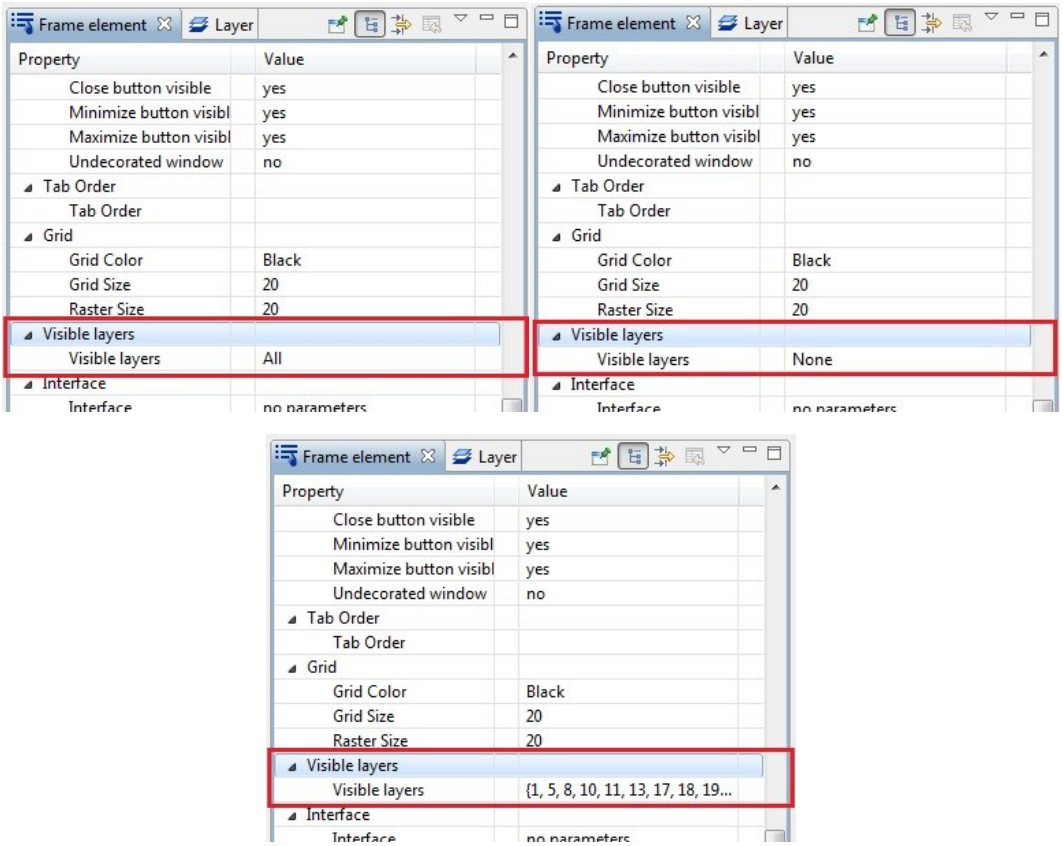
Obrázek 1.6: *Návrh dialógu*

Pre implementáciu novej funkcie do stávajúceho editoru som vytvoril triedu *LayerVisiblePD*, ktorá mala za úlohu nastaviť obsah pre editor a inicializovať dialóg, ktorý sa po kliknutí na editor zobrazil. Obsah pre editor bol textový reťazec, získaný z objektu *PdoLayers*, ktorý uchovával stav viditeľnosti každej vrstvy. Tento reťazec sa dynamicky menil, na základe zmien vykonaných v editačnom dialógu.

Ukážka metódy, ktorá vytvárala obsah pre editor. V prípade, že boli všetky vrstvy viditeľné, metóda vrátila reťazec "All" , v prípade, že boli všetky vrstvy nastavené ako neviditeľné, metóda vrátila reťazec "None", inak metóda vrátila reťazec pozostávajúci z hodnôt *true* alebo *false* v závislosti na tom či bola vrstva viditeľná alebo nie.

LayerVisiblePD.class

```
private String buildStringFromEntries(Object element) {
    PdoLayers pdoLayer = (PdoLayers) element;
    int visibleLayerCount = 0;
    for (int i = 0; i < Element.MAX_LAYER_NUM; i++) {
      if (pdoLayer.isVisible(i)) {
        visibleLayerCount++;
      }
    }
    if (visibleLayerCount == 0) {
      return Messages.LayerVisiblePD_VisibleNone;
    } else if (visibleLayerCount == Element.MAX_LAYER_NUM) {
      return Messages.LayerVisiblePD_VisibleAll;
    } else {
      return pdoLayer.asBitSet().toString();
}
```



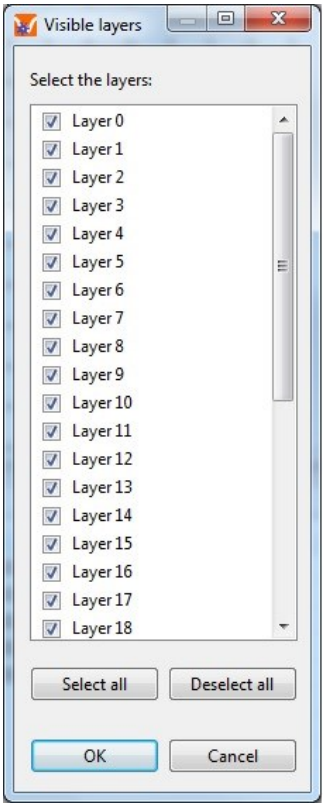
Obrázek 1.7: *Editor vlastností ProFrame™ objektu*

Vytvoril som triedu *LayerVisibleDialog*, ktorá dedila z už vytvorenej triedy *BaseDialog*. Z *BaseDialog* dedila väčšina dialógov v ProFrame™ editore. V tejto triede sa implementoval dialóg, ktorý akceptoval ako parametre objekt *Shell* a objekt *PdoLayers*. Objekt *Shell* bol rodičovský objekt, na ktorom sa dialóg zobrazil a objekt *PdoLayers* obsahoval informácie o jednotlivých vrstvách, ako napr. stav viditeľnosti vrstiev. V dialógu boli vytvorené *checkboxi* pre každú vrstvu a tlačidlá "Select all" a "Deselect all", ktoré umožňovali zaškrtnúť alebo odškrtnúť všetky *checkboxi* naraz.

Ukážka metódy, ktorá zistila či je *checkbox* zaškrtnutý a zistenú hodnotu *true* alebo *false* spolu s indexom *checkboxu* uložila do objektu *BitSet*. Následne metóda vrátila nový objekt *PdoLayers* vytvorený na základe získaných hodnôt z objektu *BiSet*.

LayerVisibleDialog.java

```
private PdoLayers getCheckedLayers(Table table) {
    TableItem[] items = table.getItems();
    BitSet bitSet = new BitSet(LAYERS_COUNT);
    for (int i = 0; i < items.length; i++) {
      boolean visible = items[i].getChecked();
      bitSet.set(i, visible);
    }
    PdoLayers newVisibleLayers = new PdoLayers(bitSet);
    return newVisibleLayers;
}
```



Obrázek 1.8: *Finálny dialóg*

3.4 Eclipse plugin: ProFrame™ komparátor

Pri práci so softwarom IDS HIGH-LEIT™ bolo občas potrebné porovnať obsah dvoch ProFrame™ súborov, najčastejšie pri hľadaní chýb. Pán Ing. Ivo Hajduček už v minulosti pracoval na komparátore a mal vytvorenú logiku porovnávania *PdoProFrame* objektov. Z tohto dôvodu vznikol nápad vytvoriť projekt komparátor ProFrame™ súborov s užívateľským rozhraním.

Projekt sa rozdelil do štyroch častí. Prvá časť bola vytvorenie grafického užívateľského rozhrania pre ProFrame™ komparátor. V druhej časti sa mala rozšíriť funkčnosť komparátoru pre detailnejšie porovnávanie. Tretia časť bola možnosť zistené rozdiely odstrániť alebo premietnúť do porovnávaných súborov. V poslednej štvrtej časti sa finálny komparátor integroval do softwaru IDS HIGH-LEIT™.

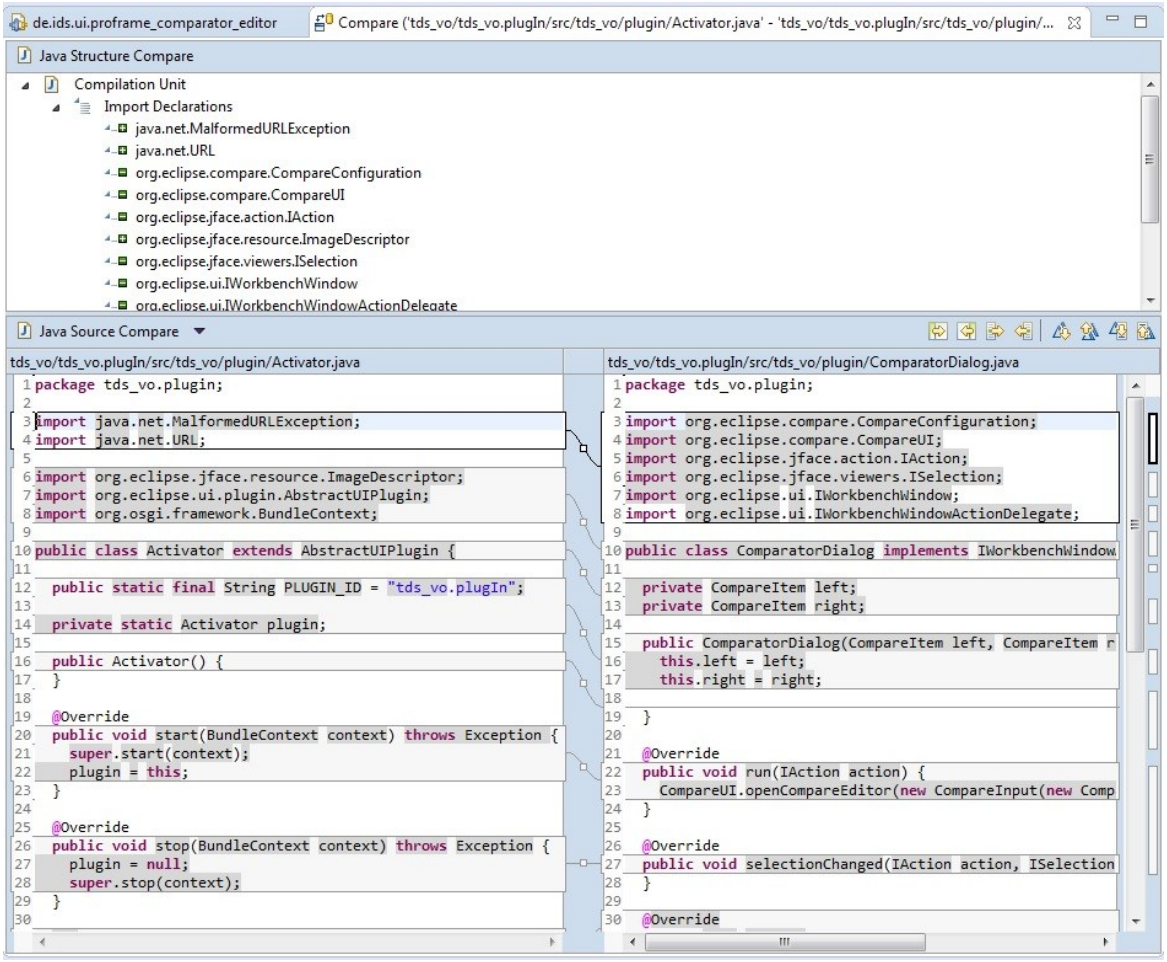
3.4.1 Tvorba GUI pre ProFrame™ komparátor

Software IDS HIGH-LEIT™ je postavený na Eclipse platforme, preto najvýhodnejší spôsob rozšírenia funkčnosti softwaru je vytvorenie nového Eclipse plugin-u. V Eclipse IDE je funkcia na porovnanie dvoch objektov, *Compare With / Each other*. V našom plugin-e som použil ako rozšírenie *contentMergeViewers* z funkcie *Compare With / Each other*, tj. ProFrame™ komparátor sa bude spúšťať rovnako a základné zobrazenie bude tiež rovnaké.

Ukážka špecifikačného XML súboru Eclipse plugin-u ProFrame™ komparátor.

plugin.xml

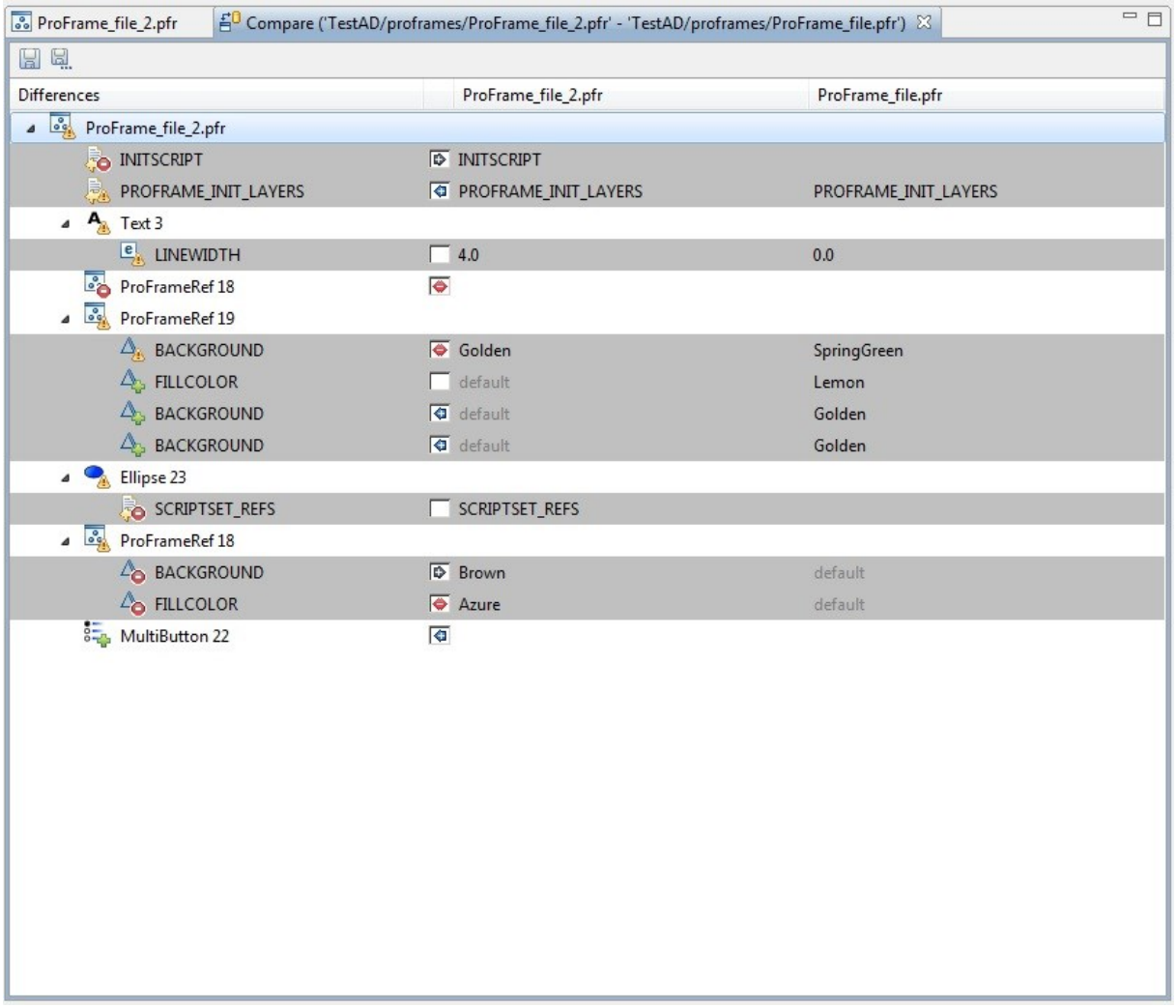
```
<plugin>
   <extension
         id="ProFrame"
         name="ProFrame"
         point="org.eclipse.compare.contentMergeViewers">
      <contentTypeBinding
            contentMergeViewerId="proFrameMargeViewer"
            contentTypeId="de.ids.ui.proframeeditor.editor.proframe-
content-type">
      </contentTypeBinding>
      <viewer
class="de.ids.ui.proframe_comparator.editor.creators.CompareViewCrea
tor"
            id="proFrameMargeViewer"
            label="ProFrame comparator">
      </viewer>
   </extension>
</plugin>
```



Obrázek 1.9: *Pôvodná funkcia "Compare With / Each other"*

Ako som už spomínal v predchádzajúcich odstavcoch, všetky GUI komponenty softwaru IDS HIGH-LEIT™ sú z frameworku SWT - JFace. Pre zobrazenie obsahu komparátora som zvolil *TreeViewer*, ktorý umožňuje zobraziť obsah do stromovej štruktúry. Náš *TreeViewer* má štyri stĺpce. Prvý zobrazuje *PdoProFrame*, *PdoElement* objekty a ich vlastnosti. Druhý zobrazuje checkboxi pre aplikáciu zmien. Tretí a štvrtý zobrazujú hodnoty ľavého a pravého objektu. Každý *Viewer* z frameworku SWT - JFace potrebuje pre nastavenie a spracovanie vstupu *ContentProvider*, v našom prípade je to *ProFrameContentProvider*, a pre nastavenie a zobrazenie obsahu *LabelProvider*, ktorý tiež potrebujú aj jednotlivé stĺpce, v našom prípade sú to *ProFrameLabelProvider*, *MergeCheckboxLabelProvider*, *LeftPropertyLabelProvider* a *RightPropertyLabelProvider*.

V *ProFrameContentProvider*, sa spracujú porovnávane objekty, tj. skontroluje sa požadovaný typ a následne sa predajú komparátoru na porovnanie. LabelProvideri zase spracujú výsledný obsah z *ProFrameContentProvider* do grafickej podoby.



Obrázek 1.10: *ProFrame™ komparátor GUI*

3.4.1.1 ***ProFrameContentProvider.java***

Trieda implementuje rozhranie *ITreeContentProvider* a pri spustení funkcie *Compare With / Each Other* sa volá metóda *inpudChanged*, a nastaví sa ľavý a pravý koreňový objekt. Koreňové objekty sa spracujú v metóde *getElements*, ktorá vráti pole *Object* získane z metódy *getChildren*. Metóda *getChildren* v sebe volá metódu komparátora *getDifference*, ktorá rozdelí koreňové *PdoProFrame* objekty na jednotlivé *PdoElement* objekty a ak je to potrebné, tak ich ešte rozdelí na jednotlivé vlastnosti, porovná ich a pre každý objekt, prípadne vlastnosť, vytvorí objekt *Difference*.

Objekt *Difference* pozostáva z porovnávaného ľavého a pravého objektu, z typu rozdielu (ADDED, REMOVED, CONFLICTED, EQUAL) a prípadne ešte zo zoznamu ďalších *Difference* objektov. Pre potreby aplikácie zmien, ktorým sa budem venovať neskôr sa objekty *Difference* zabalia do objektu *DifferenceWraper*, ktorý má navyše vlastnosť *ChangeType*.

Výsledný obsah *ProFrameContentProvider* je pole objektov *DifferenceWraper*.

3.4.1.2 ***ProFrameLabelProvider.java***

Trieda dedí z triedy *PropertyLabelProvider*, ktorá implementuje rozhrania *IStyledLabelprovider* a *IColorProvider* čo jej umožňuje, okrem nastavenia textového obsahu, nastavenie ikon a farieb pozadia alebo písma.

Pri spustení funkcie *Compare With / Each Other* sa zavolajú metódy *getStyledText*, *getImage*, *getForeground* a *getBackground*. Metódy podľa typu a vlastností objektu nastavia zobrazovaný text, ktorým je názov objektu alebo vlastnosti, ikonu znázorňujúcu objekt alebo vlastnosť s dekoratívnou ikonou znázorňujúcou typ rozdielu a farby pozadia a písma ak to objekt alebo vlastnosť vyžaduje.

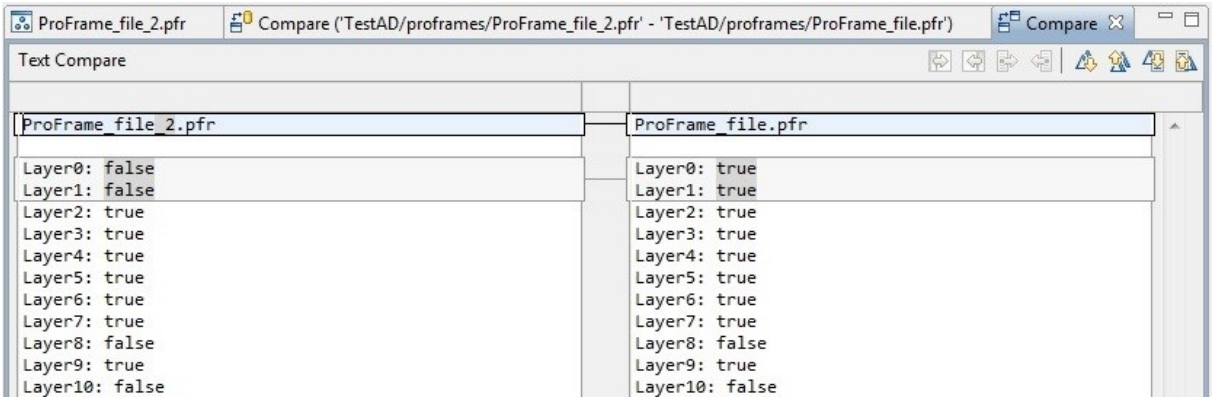
3.4.1.3 ***LeftPropertyLabelProvider.java a RightPropertyLabelProvider.java***

Tieto triedy sú funkčnosťou totožné a tiež dedia z *PropertyLabelProvider*. *LeftPropertyLabelProvider* a *RightPropertyLabelProvider* nastavujú textovú podobu hodnôt vlastností v stĺpcoch pre ľavý a pravý objekt. Princíp je v podstate rovnaký ako pri *ProFrameLabelProvider*, len už sa nenastavujú ikony.

3.4.1.4 ***ItemDoubleClickListener.java***

Niektoré vlastnosti *PdoProFrame* objektov nám v jednoriadkovom zobrazení veľa nepovedia, preto bolo nutné, pre detailnejšie znázornenie rozdielov, zobrazenie v samostatnom editore.

Trieda dedí z *MouseAdapter* a prepisuje metódu *mouseDoubleClick*. Ako už z názvu metódy vyplýva, po dvojkliku na položku v strome sa zavolá metóda *mouseDoubleClick*, v ktorej sa vyhodnotí typ objektu z položky, ak objekt vyhovuje otvorí sa editor s obsahom.



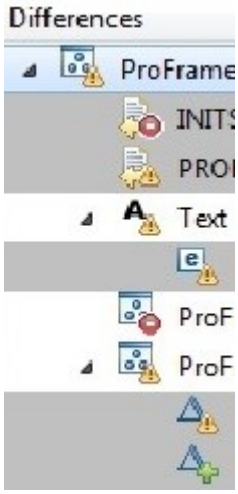
Obrázek 1.11: *Editor pre vlastnosť PROFRAME_INIT_LAYERS*

3.4.2 **Rozšírenie funkčnosti ProFrame™ komparátoru**

Objekt *PdoProFrame* môže v sebe obsahovať ďalšie *PdoProFrame* objekty, napr. rôzne pred-pripravené symboly apod., uložené v podobe *PdoProFrameRef* objektov, tie majú špeciálne vlastnosti *PdoOverloadedProperties*. Komparátor *PdoOverloadedProperties* neporovnával, preto bolo potrebné implementovať porovnávanie aj pre tieto objekty.

3.4.3 Vyhodnotenie výsledkov a aplikácia zmien do ProFrame™ súborov

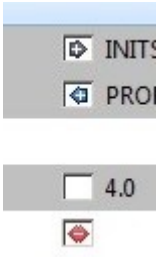
Komparátor v GUI zobrazil pomocou dekoračných ikon typ rozdielu.



Obrázek 1.12: *Dekoračné ikony*

- Objekt alebo vlastnosť je v pravom súbore a chýba v ľavom súbore
- Objekt alebo vlastnosť je v ľavom súbore a chýba v pravom súbore
- Objekt alebo vlastnosť je v oboch súboroch ale má odlišné hodnoty

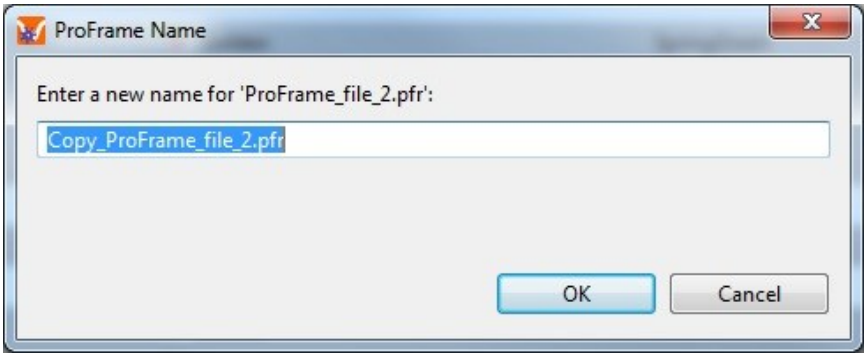
Rozdiely je potrebné spracovať a výsledok uložiť do súboru. Na to som do GUI komparátoru pridal druhý stĺpec s checkbox-ami, ktoré po kliknutí menili svoju ikonu. Na základe významu konkrétnej ikony sa vykonala príslušná akcia.



Obrázek 1.13: *Checkbox-i*

- Odstránenie rozdielu
- Pridanie objektu alebo hodnoty vlastnosti z pravého do ľavého súboru
- Pridanie objektu alebo hodnoty vlastnosti z ľavého do pravého súboru
- Žiadna akcia

Pre uloženie nastavených zmien na checkbox-och som pridal dve ikony jednu na uloženie do stávajúcich súborov "Save" a druhú "Save as", ktorá vyvolala dialóg pre zadanie názvu a zmeny uložila do nových súborov a pôvodné súbory ostali zachované bez zmeny.



Obrázek 1.14: *"SaveAs" dialóg*

3.4.3.1 ***MergeCheckboxLabelProvider.java***

Trieda implementuje rozhrania *IStyledLebalProvider* a *IColorProvider*. Nastavuje defaultnú ikonu checkbox-u ☐ a farbu pozadia.

3.4.3.2 ***MergeCheckBoxStateChanger.java***

Výstup z *ProFrameContentProvider* je pole *DifferenceWraper* objektov, ktoré majú defaultne nastavenú vlastnosť *ChangeType* na hodnotu NOTHING. Každá hodnota vlastnosti *ChangeType* (NOTHING, ADD_INCOME, ADD_OUTCOME, REMOVE) odpovedá jednej ikone checkbox-u.

Táto trieda slúži na dynamickú zmenu vlastnosti *ChangeType*, na základe ktorej sa mení ikona checkbox-u a text uložený pod checkbox-om, tento text sa nezobrazuje, ale slúži len na kontrolu.

Hodnota *ChangeType* sa mení v cykloch uložených v mapách. Aktuálna hodnota *ChangeType* vložená do metódy je použitá ako kľúč a metóda následne vráti novú hodnotu *ChangeType* uloženú pod týmto kľúčom.

Ukážka metódy *getChangeType*, ktorá vráti novú hodnotu *ChangeType* na základe aktuálnej hodnoty *ChangeType* z objektu *DifferenceWraper*.

```
  public ChangeType getChangeType(DifferenceWraper wraper) {
if (!isPropertyCompareContainer(wraper.getDifference())) {
  if (isConflicted(wraper.getDifference())) {
    return conflictedCycle.get(wraper.getChangeType());
  }
  if (isAdded(wraper.getDifference())) {
    return addedCycle.get(wraper.getChangeType());
  }
  if (isRemoved(wraper.getDifference())) {
    return removedCycle.get(wraper.getChangeType());
  }
```

```
    } else {
      return
propertyCompareContainerCycle.get(wraper.getChangeType());
    }
    throw new IllegalArgumentException(Messages.Invalid_situation);
  }
```

3.4.3.3 ***ProFrameCreator.java***

Trieda pracuje priamo s porovnávanými súbormi a na základe objektov *Changes*, uložených v zozname, aplikuje zmeny do aktuálnych súborov alebo vytvorí nové súbory. Objekt *Changes* uchováva vlastnosť *ChangeType*, ľavý a pravý porovnávaný objekt a ich rodičovský objekt zo stromovej štruktúry.

Po stlačení tlačidla "Save" alebo "Save as" sa vyvolá jediná verejná metóda *mergeProFrames*, ktorá dostane ako parametre ľavý a pravý *ResourceNode* objekt, ktorý uchováva odkaz na vstupný súbor, zoznam objektov *Changes*, rodičovský objekt a *boolean* objekt *overWrite* (hodnota sa mení podľa toho, či bolo stlačené tlačidlo "Save" hodnota *true* alebo "Save as" hodnota *false*).

Metóda najskôr z objektov *ResourceNode* získa ľavý a pravý objekt *PdoProFrame*, z nich získa zoznamy jednotlivých *PdoElement* objektov. Potom sa v cykle prechádza zoznam objektov *Change*s. Jednotlivé objekty zo zoznamu sa testujú a podľa príslušného typu objektu sa zavolá metóda, ktorá aplikuje zmeny do *PdoProFrame* objektov. Následne sa volá metóda *saveMergeFiles*, ktorá zmenené *PdoProFrame* objekty uloží do súborov. Podľa *boolean* hodnoty parametru *overWrite*, sa uloženie vykoná do stávajúceho súboru, hodnota *true*, alebo do nového súboru, hodnota *false*.

Ukážka metódy *mergeProFrames*.

```
public void mergeProFrames(ResourceNode left, ResourceNode right,
List<Changes> changes, Composite parent, boolean overWrite)
      throws CoreException {
    try {
      PdoProFrame rightProFrame =
PdoStreamer.read(right.getContents(), null);
      List<PdoElement> rightObjects = rightProFrame.getObjects();
      PdoProFrame leftProFrame =
PdoStreamer.read(left.getContents(), null);
      List<PdoElement> leftObjects = leftProFrame.getObjects();
```

```
      for (int i = 0; i < changes.size(); i++) {
        if (isPropertyCompareContainer(changes.get(i))) {
          mergePropertyCompareContainer(i, changes, leftObjects,
rightObjects);
        } else if (isPdoElement(changes.get(i))) {
          mergePdoElement(i, changes, leftObjects, rightObjects);
        } else if (isPdoOverloadedProperty(changes.get(i))) {
          mergePdoOverloadedProperty(i, changes, leftObjects,
rightObjects);
        }
      }
      saveMergeFiles(overWrite, parent,
left.getResource().getLocation(), right.getResource().getLocation(),
leftProFrame, rightProFrame);
    } catch (Exception e) {
      log.error(Messages.Merge_exception, e);
    }
  }
```

3.4.4 Integrácia ProFrame™ komparátora do softwaru IDS HIGH-LEIT™

Výsledný ProFrame™ komparátor bol testovaný kolegami na správnu funkčnosť, logické a intuitívne ovládanie. Po úspešnom teste, som vytvoril code-review (revízia zdrojových kódov vytvoreného projektu), v ktorom sa kolegovia vyjadrili k implementačným riešeniam, prípadne navrhli prehľadnejšie alebo efektívnejšie riešenie. Následne som podľa návrhov kolegov upravil zdrojové kódy projektu.

Projekt sa po týchto úpravách stal použiteľný a zaraditeľný do distribúcie softwaru IDS HIGH-LEIT™ ako nová užívateľská funkcia.

4 Získané a chýbajúce znalosti a skúsenosti počas odbornej praxe

Pri absolvovaní odbornej praxe som využil znalosti získané počas štúdia z predmetov Programovacie jazyky I a II, Softwarové inžinierstvo a Užívateľské rozhrania.

Chýbajúce znalosti a schopnosti, ako orientácia v rozsiahlom projekte, práca v týme, obsluha verzovacieho nástroja a používanie efektívnych implementačných riešení, som získal až počas odbornej praxe, častým stykom s programovaním a praktikami spojenými s prácou na rozsiahlom projekte.

Záver

V práci som čitateľov zoznámil s firmou TELE DATA SYSTEM, spol. s.r.o., s mojím pracovným zaradením. Popísal som nástroje a technológie, ktoré som používal pri práci na zadaných úlohách, jednotlivé úlohy a projekt.

Moje pôsobenie vo firme, počas odbornej praxe, hodnotím pozitívne. Projekt ProFrame™ komparátor, na ktorom som strávil najviac času, je zaradený do distribúcie softwaru IDS HIGH-LEIT™ a je využívaný zákazníkom. Firma bola s mojou prácou spokojná a ponúkli mi ďalšiu spoluprácu.

Absolvovanie individuálnej odbornej praxe je výborný spôsob ako získať praktické skúsenosti z reálneho prostredia, ktoré študentovi výrazne zvýšia šance pri hľadaní zamestnania.

Referencie

[1] Petr Hatina, Programování v jazyku Java(1) - úvod. [online]. [cit. 2016-04-09]. Dostupné z: http://www.linuxsoft.cz/article.php?id_article=244

[2] Eclipse (software). Wikipedia: the free encyclopedia. [online]. [cit. 2016-04-09]. Dostupné z: https://en.wikipedia.org/wiki/Eclipse_%28software%29

[3] What is SCADA?. Inductive Automation. [online]. [cit. 2016-04-09]. Dostupné z: https://inductiveautomation.com/what-is-scada

[4] SWT. The Eclipse Foundation. [online]. [cit. 2016-04-09]. Dostupné z: https://www.eclipse.org/swt/

[5] Using JUnit in Eclipse. [online]. [cit. 2016-04-09]. Dostupné z: https://courses.cs.washington.edu/courses/cse143/11wi/eclipse-tutorial/junit.shtml

[6] What is Maven?. The Apache Software Foundation. [online]. [cit. 2016-04-09]. Dostupné z: https://maven.apache.org/what-is-maven.html

[7] SVN Tutorial. Tutorials Point: simple easy learning. [online]. [cit. 2016-04-09]. Dostupné z: http://www.tutorialspoint.com/svn/index.htm

[8] Produkty. Atlassian Project Tracking Tools. JIRA. Onlio, a.s.. [online]. [cit. 2016-04-09]. Dostupné z: http://www.myjira.cz/produkty/project-tracking-tools/jira.html

[9] What is Agile? What is Scrum?. CPRIME: an ALTEN Group Company. [online]. [cit. 2016-04-09]. Dostupné z: https://www.cprime.com/resources/what-is-agile-what-is-scrum/