České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Jan Holý**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Rozšíření programu OpenChrom o integrované řízení sběru dat**

Pokyny pro vypracování:

1. Prostudujte zdrojové kódy analytického software OpenChrom a seznamte se s převodníkem analogového signálu ULAD32 určeného pro sběr signálu chromatografického detektoru
2. Navrhněte, jakým způsobem integrovat sběr dat do projektu OpenChrom
3. Implementujte modul pro sběr dat a příslušné grafické prvky pro řízení sběru, monitorování průběhu a předání nasnímaných dat pro zpracování analytickými metodami obsaženými v programu OpenChrom
4. Práci zdokumentujte a zaměřte se především na úpravy v obecných částek kódu a ty konzultujte s vedoucími vývojáři projektu
5. V rámci časových možností se pokuste dohodnout začlenění úprav do hlavního vývojového stromu projektu

Seznam odborné literatury:

[1] Philip Wenig: Softwarebasierte Verfahren Zur Datenbankgestützten Identifizierung Organischer Substanzen Mittels Analytischer Pyrolyse Gekoppelt Mit Gaschromatographie, Massenspektrometrie (Py-GC, MS), dizertační práce, Universita Hamburk 2012
[2] projekt OpenChrom - zdrojové kódy a dokumentace, online http://www.openchrom.net/
[3] Pavel Píša: Matematické a elektronické zpracování signálu kapalinového chromatografu, dizertační práce, ČVUT FEL 2010

Vedoucí: Ing. Pavel Píša, Ph.D.

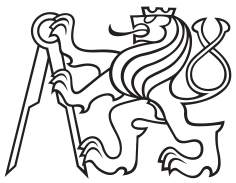
Platnost zadání: do konce letního semestru 2015/2016

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 29. 1. 2015

Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídící techniky

Rozšíření programu OpenChrom o integrované řízení dat

Jan Holý
Program: Kybernetika a robotika
Obor: Systémy a řízení

Leden 2016
Vedoucí práce: Ing. Pavel Píša, Ph.D.

Poděkování / Prohlášení

Děkuji svému vedoucímu bakalářské práce, panu Ing. Pavlu Píšovi, Ph.D. , za velkou ochotu, trpělivost a čas, který mi poskytl při vedení této práce. Také bych chtěl poděkovat Dr. Philip Wening z Univerzity v Hamburku za pomoc při tvorbě rozšíření do programu OpenChrom.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 12. 1. 2016

.....

Abstrakt / Abstract

Chromatografie patří mezi separační metody. Separační metody se používají buď k získávání čistých látek, nebo pro analýzu jednotlivých složek vzorku. Pro vyhodnocení záznamu, který se získává při analytické chromatografii, byla vytvořená celá řada specializovaných programů. K dispozici jsou rozsáhlé komerční systémy, ale i otevřený software, který využívá mnoho týmů. Mezi otevřený software patří například program CHROMuLAN nebo program OpenChrom. Program CHROMuLAN umožňuje nejen vyhodnocení dat, ale i komunikaci a sběr dat ze zařízení připojených k počítači.

Nově vzniklý projekt OpenChrom získal silné zázemí a přísun investic v rámci Eclipse Foundation. Pro využití toho projektu i v institucích, kde je v současné době používaný projekt CHROMuLAN, je potřeba rozšířit program Openchrom o monitorování a provádění sběru dat. Návrh a realizace tohoto rozšíření byly hlavním úkolem mé práce.

Klíčová slova: OpenChrom, CHROMuLAN, Eclipse RCP, OSGi, uLan, Analytická chromatografie, Chromatografická akvizice dat

Chromatography is one of the separation techniques in chemistry. Separation techniques are used either for obtaining pure substances or compounds from some mixture of substances or analysing individual substances of the mixture. Variety of specialized programs are used for processing the chromatography records. There are complex commercial software solutions available yet many teams choose to use some open software instead such as CHROMuLAN or OpenChrom. The CHROMuLAN is capable to communicate and acquire data from instruments connected to computer in addition to data processing and evaluation.

New project OpenChrom is becoming more and more popular today and its development is backed by funding from the Eclipse Foundation. It is necessary to extend The OpenChrom program by creating data acquisition and monitoring extension in order to be able to fully replace CHROMuLAN in institutions in which CHROMuLAN is used. The main core of the thesis is the design and implementation of this extension.

Keywords: OpenChrom, CHROMuLAN, Eclipse RCP, OSGi, uLan, Analytical chromatography, Chromatography data acquisition

Title translation: Online data acquisition support for OpenChrom analytical software

Obsah /

Obrázky

Kapitola 1
Úvod

Cílem mé práce je rozšířit program OpenChrom[1]) způsobem, který mu umožní řízení sběru dat, monitorování sběru dat a přípravu těchto dat k dalšímu zpracování přímo z prostředí programu OpenChrom. Sběr dat bude probíhat prostřednictvím analogově digitálního převodníku ULAD 31 nebo ULAD 32, který je určen ke sběru dat z chromatografických detektorů, které jsou vybaveny analogovým výstupem. Tyto převodníky komunikují prostřednictvím síťového protokolu uLan. Pro dosažení cíle mé práce bylo potřeba realizovat dva projekty. Prvním projektem je knihovna napsaná v jazyce Java. Tato knihovna slouží jako ovladač pro komunikaci se zařízeními, které komunikují prostřednictvím síťového protokolu uLan. Druhým projektem je rozšíření programu OpenChrom, které umožní zaznamenat data ze zařízení ULAD 31 a ULAD 32. Zaznamenaná data umožňuje rozšíření uložit do souboru, který je možné zpracovat v programu OpenChrom.

Zmíněné programy, rozbor a řešení projektu jsou popsané v následujících kapitolách.

V druhé kapitole je shrnut princip chromatografického stanovení kvalitativního a kvantitativního složení roztoků a směsí.

Ve třetí kapitole se věnuji komponentám vyhodnocujícího řetězce. V této kapitole se nejdříve věnuji obecně vyhodnocujícímu řetězci. Poté se již věnuji těm komponentám vyhodnocujícího řetězce, které se týkají mé bakalářské práce. V kapitole popisuji analogově digitální převodníky ULAD 32 a ULAD 31 a síťový protokol uLan, kterým tyto převodníky komunikují. Dále se pak věnuji popisu programu CHROMuLAN[2]) a programu OpenChrom.

Ve čtvrté kapitole se zabývám nejdůležitějšími nástroji, které jsem použil při vývoji ovladače a při vývoji rozšíření programu OpenChrom. Společným jmenovatelem obou projektů je objektově orientovaný programovací jazyk Java. Pro vývoj ovladače jsem musel použít rozhraní JNI (Java Native Interface). Toto rozhraní umožňuje spolupráci s knihovnami, které jsou napsané v jiných jazycích. Dále pak popisuji základní platformy Eclipse RCP, na které je program OpenChrom vytvořen.

V páté kapitole popisuji objektový model ovladače ulan-java, který slouží ke komunikaci prostřednictvím síťového protokolu uLan.

V šesté kapitole popisuji jakým způsobem byl program OpenChrom rozšířen tak, aby umožňoval sběr dat, monitorování sběru dat a přípravu těchto dat k dalšímu zpracování přímo z prostředí programu OpenChrom. Nejdříve popisuji objektový model, který je použit v tomto rozšíření. Poté se věnuji základnímu principu fungování toho rozšíření z programátorského hlediska.

V sedmé kapitole se věnuji popisu uživatelského rozhraní nově vytvořeného rozšíření programu OpenChrom.

V osmé kapitole shrnuji výsledky testování ovladače ulan-java a testování vytvořeného rozšíření programu OpenChrom.

[1]) `https://openchrom.net/`
[2]) `http://www.chromulan.org/`

Kapitola 2
Chromatografie

Chromatografie patří mezi separační metody. Separační metody se používají k získání čistých látek, nebo ke kvalitativní i kvantitativní analýze vzorku.

Chromatografie je metoda založená na mobilní a stacionární fázi. Látka, která obsahuje několik složek, je unášena mobilní fází skrz statickou fázi. Ve statické fázi jsou různé složky látky rozdílně pozdrženy, čímž dochází k separaci jednotlivých složek látky. Retenční čas je doba, po kterou je složka látky pozdržena v stacionární fázi[1]. Při analytické chromatografii je nutné po separaci jednotlivých složek látky v mobilní fázy detekovat. K detekci složky látky se používá detektor.

2.1 Základní dělení chromatografie

Chromatografii můžeme rozdělit podle různých kritérií[1]

- Podle povahy mobilní fáze se dělí na plynovou (GC) a kapalinovou (LC).
- Podle povahy stacionární fáze na kolonovou (sloupcovou) a plošnou (planární).
- Podle principu separace složek látky na rozdělovací, adsorpční, iontově výměnnou a gelovou.
- Podle pracovního postupu na eluční (analytická), frontální, vyměšovací.
- Podle účelu na analytickou a preparativní.

2.2 Detektor

Pro detekci koncentrace separovaných látek se používají detektory. Detektory se liší pro plynovou a kapalinovou chromatografii[2]. V plynové chromatografii se využívá plamenověionizační detektor a hmotnostní spektrometr. V kapalinové chromatografii se využívá více typů detektorů, a to především spektrofotometrický detektor (UV-VIS), fluorescenční detektor, hmotnostní spektrometr či refraktometrický detektor.

2.3 Chromatogram

Chromatogram se nazývá záznam z detektoru. Chromatogram v sobě nese v zásadě dvě informace. Odezvu detektoru na složku látky a čas, který se měří od začátku analýzy, kdy tato odezva nastala. Odezvy na jednotlivé složky látky jsou od sebe odděleny základní linií[1] (base-line). Odezva na složku odpovídá většinou impulsu. Název pro tento impuls byl přezvat[1] z anglického označení peak a označuje se slovem pík. Průběh tohoto impulsu se blíží průběhu hustoty pravděpodobnosti normálního rozdělení[1].

2.4 Kolonová chromatografie

Kolonová chromatografie je označení pro druh chromatografie, kdy je statická složka umístěna v pevně ohraničeném prostoru. Tento prostor nazýváme kolona. Někdy se pro kolonovou chromatografii též používá název sloupcová chromatografie.

2.5 Kapalinová chromatografie

Při kapalinové chromatografii slouží jako nosná složka pro detekovanou látku kapalina. Jednou z nejvíce rozšířených metod kapalinové chromatografie je HPLC (high-performance liquid chromatography) tzn. vysokoúčinná (nebo vysokotlaká) kapalinová chromatografie. Tato metoda je široce využívána v mnoha oborech např. při zkoumání bioaktivních látek. Vyznačuje se tím, že mobilní fáze je tlačena pod vysokým tlakem, který může dosahovat až 40MPa[1], skrz stacionární fázi pomocí vysokotlakých čerpadel. Vysokého tlaku se používá, protože stacionární fáze klade značný odpor mobilní fázi.

Kapitola 3
Komponenty vyhodnocujícího řetězce

K získání a vyhodnocení chromatografického záznamu je třeba celá řada komponent. Je potřeba přístroj, který slouží k chromatografické separaci složek látky. Tento přístroj se nazývá Chromatograf. Chromatograf obsahuje pro detekci separovaných složek látky detektor. Data z detektoru je potřeba zaznamenat a vyhodnotit. Pro záznam a vyhodnocení dat se často používá běžný počítač, který se připojí k chromatografu a který obsahuje specializované programy pro vyhodnocení a sběr dat. Existuje řada specializovaných programů[3]. Mezi tyto programy patří například program Chromeleon[1]) od firmy Thermo Scientific nebo program ChemStation[2]) od firmy Agilent. Pro pokročilou analýzu chromatografických dat existují specializované programy např. program PeakFit[3]) nebo program SYSTAT[4]) od firmy Systat Software. Možností jak se vyhnout nákupu těchto relativně drahých programů, jejichž ceny mohou převyšovat pořizovací náklady chromatografu, je použít pro vyhodnocení dat program, který je volně dostupné jako například programu Openchrom nebo program CHROMuLAN. Program CHROMuLAN umožňuje nejen vyhodnocení záznamů, ale i sběr dat a řízení sestav přístrojů od firmy Pikron[5]).

3.1 Zařízení od firmy Pikron určená pro analytickou chromatografii

Jedním z dodavatelů zařízení určených pro kapalinovou chromatografii je firma Pikron, která je dodavatelem ucelených řešení pro HPLC. Firma Pikron také nabízí analogově digitální převodníky, které se nazývají ULAD 31 a ULAD 32. Tyto převodníky jsou určena pro sběr dat z chromatografických detektorů, které jsou vybaveny analogový výstupem.

3.1.1 Zařízení ULAD 31 a ULAD 32

Zařízení ULAD 31[4] a ULAD 32[5] jsou analogově digitální převodníky, které se vyznačují vysokým odstupem signálu od šumu[6–7]. Převodníky lze připojit přes USB rozhraní nebo uLan rozhraní.

Převodníky jsou konstruovány zejména pro připojení k detektoru v zařízeních pracujících na principu HPLC UV-Vis a také pro jiné analytické detektory, kde je upřesňována vysoká rozlišovací schopnost nad vysokým počtem vzorků.

Podpora uLan rozhraní přináší několik výhod. Rozhraní uLan umožňuje propojit více zařízení dohromady. Síťový protokol uLan umožňuje také rozmístit jednotlivá zařízení dál od sebe, než to umožnuje USB protokol. Další výhodou je to, že tyto převodníky podporuje program CHROMuLAN, který je určen pro řízení celých sestav HPLC

[1]) http://www.dionex.com/en-us/index.html
[2]) http://www.chem.agilent.com
[3]) http://www.sigmaplot.com/products/peakfit/peakfit.php
[4]) http://www.systat.com/
[5]) http://www.pikron.com

přístrojů[8] navržených ve firmě Pikron. Program CHROMuLAN umožňuje nastavit parametry převodníku a také umožňuje zaznamenávat a zpracovávat data z tohoto převodníku.

Analogově digitální převodník ULAD 32 je nejnovější verzí analogově digitálního převodníku od firmy Pikron. Převodník ULAD 32, který se skládá z nejnovějších součástek, nabízí oproti převodníku ULAD 31 lepší odstup signálu od šumu, lepší přepěťovou a nadproudovou ochranou. Převodník ULAD 32 mimo vylepšení převodních charakteristik nabízí také některé nové funkce.[6]

3.2 Síťový protokol uLan

ULan je devítibitový asynchronní komunikační protokol, jehož řídicí systém není centralizovaný a ani není pevně stanovený master. Každé zařízení proto obsahuje vlastní logiku.

Tento síťový protokol využívá fyzickou vrstvu RS-485[9]. Fyzická vrstva RS-485 přináší výhodu potřeby pouze dvou vodičů pro přenos dat. Navíc těmito vodiči dokáže komunikovat na poměrně velkou vzdálenost (stovky metrů). Standartní rychlost protokolu je 19200 Bd[9].

Tento sítový protokol poskytuje rozhraní, jehož prostřednictvím lze komunikovat s jednotlivými zařízeními. Pro každé rozhraní počítače připojeného do sítě uLan, je vytvářeno ovladačem znakové zařízení pro Linux nebo systémové zařízení pro WindowsNT[9]. Všechny operace, které umožňují komunikovat se zřízeními, jsou přístupné prostřednictvím standartních systémových volání open, close, read, write, ioctl a select[9]. Funkce ioctl je použita pro operaci se zprávami. Prostřednictvím ioctl se posílá datová struktura ul_msginfo[9]. Tato struktura obsahuje informace o příjemci zprávy, odesilateli zprávy, příkazu nebo typu zprávy, příznaku zprávy a o unikátním číslu zprávy.

3.2.1 Zprávy

Síťový protokol uLan nabízí dva základní typy zpráv, a to servisní zprávu a procesní zprávu[9]. Servisní zpráva slouží pro komunikaci mezi dvěma zařízeními. Procesní zprávy odesílají zařízení do sítě spontánně, bez určení příjemce. Je na příjemci, jestli procesní zprávu přijme či nikoliv. Pro příjem procesní zprávy je nutné nastavit filtr, kterým deklarujeme, že danou zprávu chceme přijmout.

3.3 Program CHROMuLAN

Program CHROMuLAN má otevřený zdrojový kód a je napsán ve vývojovém prostředí DELPHI[10]. Tento program je určen především pro sběr a analýzu chromatografických dat. Program spolupracuje se zařízeními, které komunikují prostřednictvím síťového protokolu uLan. Program umožňuje získat data ze zařízení, tyto data uložit a následně data zpracovat. V programu je také možné nastavit parametry jednotlivým zařízením.

3.4 Program OpenChrom

Program OpenChrom je určen pro analýzu chromatogramů. Tento program by se mohl v budoucnu použít jako plnohodnotná alternativa k programu CHROMuLAN, která by přinášela některé výhody např. multiplatformnost, nativní ovládací grafické prvky uživatelského rozhraní a širší komunitu, která stojí za jeho vývojem a postupně rozšiřuje funkčnost tohoto programu.

OpenChrom je program, který byl vytvořen za účelem vyhodnocování chromatogramů založených na měření hmotnostního spektra látek. Jeho hlavní výhody spočívají v otevřeném zdrojovém kódu, multiplatformnosti a podpory mnoha různých datových formátů[11]. Podporované formáty lze v případě potřeby velmi jednoduše rozšířit pomocí zásuvného modulu[11]. Uživatel proto nepotřebuje množství různých programů pro práci s různými formáty, ale stačí mu jediný program. Program OpenChrom také umožňuje vyhodnocovat chromatogramy, které jsou vytvořeny programy, které slouží pouze pro sběr dat a již neumožnují jejich další zpracování.

Program OpenChrom vznikl zejména pro práci s daty, ve kterých se měří hmotnostní spektrum látek. Protože je projekt modulární, bylo možné do programu postupem času přidat nástroje pro analýzu dat z ionizačního detektoru. V této době se projekt rozšiřuje o funkce na podporu chromatogramů, které vznikly měřením vlnového spektra látek. Tyto funkce jsou ovšem ve vývojové fázi a program OpenChrom zatím podporuje pouze zobrazování těchto chromatogramů.

3.4.1 Architektura programu

Program OpenChrom je založen na platformě Eclipse RCP [11], která dovoluje jednoduše rozšiřovat základní jádro programu zásuvnými moduly.

3.4.2 Objektový model

Program OpenChrom obsahuje objektový model pro definování chromatogramů, píků a základních línií. Tento model je abstraktně modelován rozhraními[11]. Objektový model založený na rozhraních snižuje vzájemnou závislost kódu a umožňuje jednoduché rozšíření o nové funkce. Například tento objektový model umožňuje jednoduché přidání podpory nového datového formátu. Objektový model obsahuje také abstraktní třídy implementující tato rozhraní. Abstraktní třída implementuje ty metody, u kterých se předpokládá, že budou společné se všemi třídami odvozenými od tohoto rozhraní.

3.4.3 Rozšiřování programu zásuvnými moduly

Program OpenChrom nabízí jednoduchý způsob pro rozšíření jeho funkčnosti. Program definuje rozšiřující body (extension point)[11], které umožňují jednoduše přidat do programu nové funkce. Tato vlastnost umožňuje například přidat podporu nových formátů, či přidat dalšího způsobů detekci píků.

3.4.4 Uživatelské rozhraní programu

Uživatelské prostředí obsahuje řadu perspektiv. Perspektivy jsou určeny pro konkrétní operaci s daty[11], například existuje perspektiva pro manuální detekci píků či perspektiva pro integraci plochy píku. Perspektiva v sobě sdružuje vybrané pohledy (views) a editor [11]. Editor slouží k zobrazení daného chromatogramu. Pohledy zobrazují data chromatogramu z různých hledisek. Pohledy jsou umístěny v jednotlivých perspektivách tak, aby usnadnily konkrétní práci s daty.

Kapitola 4
Použité nástroje

4.1 Programovací jazyk Java

Program OpenChrom je vytvářen v programovacím jazyce Java, a proto veškerý kód rozšíření programu OpenChrom byl napsán v tomto programovacím jazyce. Ovladač, který umožňuje komunikovat se zařízeními prostřednictvím síťového protokolu uLan, byl z větší části napsán také v programovacím jazyku Java. Části kódů, které přímo využívají systémová volání operačního systému, byly napsány v programovacím jazyku C.

Java je programovací objektově orientovaný jazyk, jenž byl vyvinut společností Sun Microsystem. Jeho hlavní předností je jednoduchost, přehlednost zdrojového kódu, automatická správa paměti a multiplatformnost[12]. Java patří do rodiny jazyků, které vykonává virtuální stroj. Zdrojový kód je nejprve přeložen do mezikódu, který se nazývá bytecodu. Bytecode je pak prováděn virtuálním strojem – Java Virtual Machine[13].

4.2 JNI

Protože Java byla psaná s ohledem na to, aby podporovala mnoho různých operačních systémů, má svá omezení. Jedním z nich je to, že nepodporuje některá systémová volání jako je ioctl[14]. Toto systémové volání se využívá pro komunikaci prostřednictvím síťového protokolu uLan. Bylo proto potřeba napsat některé části kódu v programovacím jazyce C a propojit tento kód pomocí JNI (Java Native Interface).

JNI je rozhraní, které umožňuje propojit knihovny napsané v jiných programovacích jazycích s jazykem Java. Toto rozhraní je obousměrné a dovoluje tedy volání nativního kódu Java aplikací i volání Java kódu nativní knihovnou[14]. Bohužel voláním nativních knihoven ztrácí kód jednu z hlavních výhod Java aplikace, multiplatformnost. Nativní knihovna pak musí být k dispozici pro každou podporovanou platformu.

4.3 OSGi

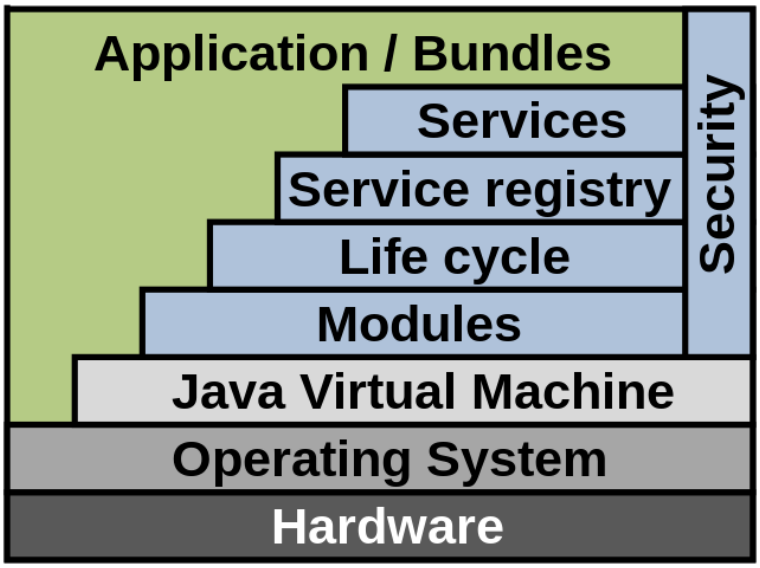
Velkou předností projektu OpenChrom je jeho modularita. Tato vlastnost je dána především tím, že OpenChrom je vytvořen na frameworku, který je založen na OSGi specifikaci.

OSGi(Open Services Gateway initiative) je specifikace pokročilého dynamického modulárního systému pro programovací jazyk Java. Moduly, ze kterých je aplikace vytvořena, budu dále nazývat komponentami. Komponenty se označují v OSGi modelu jako bundle. Tuto specifikaci spravuje společnost OSGi Alliance[1]). OSGi model umožňuje skrýt implementaci jednotlivých komponent. Pro komunikaci mezi jednotlivými komponentami se využívá služeb (services)[15].

[1]) `https://www.osgi.org/`

OSGi model se skládá z následujících částí[15]

- Bundles – Komponenty, které vytváří programátor.
- Services – Služby, které dynamicky spojují jednotlivé komponenty.
- Life-Cycle – Rozhraní slouží k instalaci, spuštění, zastavení a odinstalaci jednotlivých komponent.
- Modules – Vrstva, která definuje jakým způsobem mohou komponeny exportovat a importovat kód.
- Security – Nepovinná vrstva prolínající se celou platformou, která se stará o bezpečnostní stránku.
- Execution Environment – Definuje metody a třídy dostupné na specifické platformě.



Obrázek 4.1. Architektura OSGi Modelu

4.3.1 Modularita

Modularita je základní koncept, který znamená skrytí kódu. Veškerý kód mimo komponentu je standartně skryt ostatním komponentám[15].

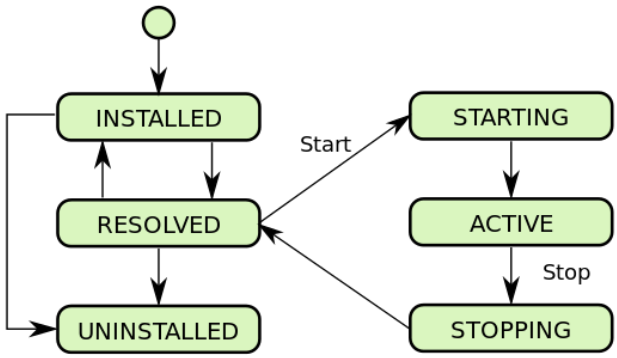
4.3.2 Služby

Služby (Services) umožňují komunikovat mezi jednotlivými komponentami pomocí Java tříd. Služby se registrují pomocí názvu třídy či rozhraní. Pod názvem jedné třídy nebo rozhraní může být definováno více služeb. Každá služba proto může obsahovat soubor klíčů s hodnotami, které jednoznačně rozliší tyto služby.

Služby umožňují dynamické připojování a odpojování za běhu programu[15], a proto je možné instalovat a odinstalovat jednotlivé komponenty za běhu programu.

4.3.3 Životní cyklus komponenty

Komponenta se během svého životního cyklu může nacházet v několika fázích[16]. Poté, co je komponenta načtena z disku, se ocitá ve fázi installed. Následuje řešení závislostí načtené komponenty na ostatních komponentách. Jestliže jsou všechny závislosti úspěšně vyřešeny, pak se komponenta přechází do stavu resolved. Komponenta z tohoto stavu může být následně spuštěna. O běh komponenty se stará třída Activator.



Obrázek 4.2. Životní cyklus komponenty

4.3.4 Specifikace komponenty

Komponenta je uložena v běžném jar archivu. Komponenta musí obsahovat konfigurační soubor Manifest.mf[17]. Tento soubor obsahuje povinné a nepovinné hlavičky, které popisují danou komponentu.

4.4 Eclipse RCP

Eclipse RCP(Eclipse Rich Client Platform)[1]) je platforma, která je využívána pro psaní aplikací, u kterých je vyžadováno rozsáhlé uživatelské rozhraní. Platforma Eclipse RCP vznikla z projektu vývojového prostřední pro programovací jazyk Java. Jádrem této platformy je implementace OSGi specifikace, a proto se aplikace skládá z komponent. Tyto komponenty se označují jako plug-in[2]). Komponenty se mohou sdružovat do logických jednotek, které se nazývají feature[19].

Jádro Eclipse RCP aplikace se skládá z následujících částí

- Equinox framework je implementací OSGi specifikace a je využíván platformou Eclipse jako běhové prostředí.
- EMF Eclipse Modeling Framwork, tento framework nabízí funkce pro vytváření datových modelů. Z těchto datových modelů pak tento framework umožňuje generovat kód či jiný výstup.

Konfigurační nastavení komponenty jsou uložena ve dvou souborech – v povinném souboru Manifest.mf a volitelném soubor plugin.xml[19]. Soubor Manifest.mf obsahuje informace pro Equinox framework. Soubor plugin.xml obsahuje informaci o tom, jakým způsobem komponenta rozšiřuje aplikaci.

[1]) `https://wiki.eclipse.org/Rich_Client_Platform`
[2]) plug-in je pouze jiné označení pro komponentu, která se v OSGi modelu nazývá bundle[18]

4.4.1 Aplikační model

Aplikační model slouží k definování struktury aplikace. Může obsahovat jak viditelné prvky jako jsou okna (windows), panel (parts), nástrojové lišty (toolbars), tak i neviditelné prvky, jako příkazy (commands), handlery (handlers), klávesové zkratky (key binding).

Základní model aplikace je definován v neměnném souboru. Standardně se tento soubor nazývá Application.e4xmi[19] a je umístěn v základním adresáři.

4.4.2 Přehled základních prvků aplikace

- Prvek popisující aplikaci se nazývá MApplication. Všechny ostatní prvky jsou umístěny uvnitř tohoto prvku.
- Funkční prvky se nazývají MAddon . Tyto prvky většinou neobsahují uživatelského rozhraní. Vytvářejí se na začátku běhu aplikace ještě před tím, než je vytvořeno grafické uživatelské prostředí.
- Prvek reprezentující okno aplikace se nazývá MWindows.
- Prvek reprezentující okno aplikace, které obsahuje i nástrojovou lištu, se nazývá MTrimmedWindow.
- Prvek reprezentující perspektivu se nazývá MPerspective. Tento prvek rozvrhuje uspořádání panelů v aplikaci.
- Prvek reprezentující panel se nazývá MPart. Tento prvek obsahuje uživatelské rozhraní aplikace.
- Prvek MDirtyable je vlastností panelu. Tento prvek signalizuje, zda panel obsahuje data k uložení.
- Prvek, který slouží jako šablona pro panel, se nazývá MPartDescriptor. Nový panel může být vytvořen pomocí Eclipse frameworku na základě této šablony.

4.4.3 Vkládání závislostí

Vkládání závislostí je jednou z technik obráceného řízení (Inversion of Control). Pomocí této techniky se snažíme uvolnit závislosti mezi třídami. Tato technika se využívá v případech, kdy jedna třída využívá druhou třídu. Technika vkládání závislostí spočívá v tom, že pevně nedefinujeme vazbu mezi třídami tak, že v závislé třídě vytváříme instanci třídy, kterou závislá třída využívá. Místo toho do závislé třídy objekt pouze vkládáme. Ve frameworku Eclipse jsou místa, do kterých se má vkládat objekt, označena anotacemi[20].

4.4.4 Eclipse Context

Během spouštění aplikace jsou vytvářeny objekty, které implementují rozhraní IEclipseContext[19] . Tento objekt je nazýván Eclipse context, dále již pro název tohoto objektu používám slovo kontext. Kontext obsahuje datovou strukturu, kde jsou objekty uloženy pod klíčem. Jako klíč může být použitý textový řetězec, jméno třídy nebo jména rozhraní.

Třídy, které implementují interface MContext, obsahují vlastní kontext. Těmito třídami jsou MApplication, MWindows, MPerspective, MPart, MPopupMenu. Tyto objekty jsou hierarchicky uspořádány v aplikačním modelu. Kontexty jsou také svázány do stromové struktury kopírující uspořádání aplikačního modelu[19]. Kontext obsahuje referenci na svého rodiče, pokud rodič existuje.

Objekty, které obsahují kontext, mohou vkládat objekty z kontextu pomocí vkládání závislostí. Místo, kam se má vložit objekt, je označeno anotací. Při vkládání objektu se

nejprve prohledává kontext, který obsahuje daný objekt, a jestliže se objekt nenajde, tak se prohledává rodičovský kontext.
Základní objekty, které jsou umístěny v kontextu

- Objekt, ve kterém je kontext umístěn.
- Služby definované platformou Eclipse nebo registrované prostřednictvím OSGi modelu.
- Další objekty jako například objekt Composite. Tento objekt SWT knihovny slouží jako kontejner pro další prvky z SWT knihovny.

4.4.5 Eclipse event system

Eclipse event system je služba definovaná platformou Eclipse, která umožňuje komunikovat mezi jednotlivými komponentami. Tato služba pracuje na principu globální sběrnice pro příjem a odeslání událostí[21]. Jednotlivé komponenty se mohou zaregistrovat pro specifické události, jiné komponenty mohou tyto události odesílat.

Tato služba je přístupná pomocí rozhraní IEventBroker a je modelována na základě OSGi Event. Tato služba umožňuje dynamické přihlašování a odhlašování příjmu zpráv ze sběrnice. Prostřednictvím této služby lze komunikovat synchronně i asynchronně.

4.4.6 Rozšiřování základní aplikace

Základní aplikaci, která je postavena na konceptu Eclipse RCP, můžeme rozšířit pomocí příspěvků (contributions). Existují dva typy příspěvku - dynamický a statický[22]. Statický příspěvek rozšiřuje aplikaci pomocí e4xmi souboru, který se nejčastěji pojmenovává jako fragment.e4xmi. Tento příspěvek se nazývá fragment. Dynamický příspěvek rozšiřující aplikaci pomocí kódu se nazývá procesor (processor). Tento kód je umístěn v běžné třídě, kde metoda, která se má spustit, je označena anotací @Execute.

Dalším způsobem jak umožnit rozšiřování aplikace je definovat rozšiřující bod (extension point), ke kterému se mohou připojit rozšíření (extension)[23]. Tyto rozšiřující body a rozšíření se definují v souboru plugin.xml.

4.5

SWT (Standart widget Toolkit)[1]) je knihovna, která obsahuje grafické uživatelské prvky. Základním rysem této knihovny je to, že se využívá nativních prvků operačního systému[24]. Díky této vlastnosti se přizpůsobuje vzhled aplikace vytvořené za použití této knihovny operačnímu systému. Ke knihovnám operačního systému se přistupuje pomocí JNI. Veškerá komunikace s prvky grafického rozhraní běží v jednom vlákně. Prvky grafického uživatelského rozhraní nepodporují vícevláknový přístup.Platforma Eclipse RCP je pevně svázán s SWT knihovnou.

4.6 JFace

JFace[2]) je knihovna založená na knihovně SWT. JFace obsahuje frameworky a třídy, které využívají SWT komponent. JFace umožňuje asynchronní chod některých komponent a také podporuje automatické uvolňování paměti. JFace nezakrývá implementaci SWT knihoven.[25]

[1]) `https://www.eclipse.org/swt/`
[2]) `http://wiki.eclipse.org/index.php/JFace`

Pro snadnou tvorbu uživatelského rozhraní nabízí JFace několik základních nástrojů[25]

- JFacet Viewers Framework - tento nástroj umožňuje jednoduché mapování datových modelů do prvků uživatelského rozhraní. Těmito prvky jsou například tabulky a stromové struktury.
- Pomocné třídy pro efektivní zprávu barev, obrázků a fontů.
- Třídy, které slouží pro tvorbu průvodců a dialogů.
- Nástroj pro synchronizaci datového modelu s uživatelským prostředím JFate Data Binding

4.6.1 JFace Data Binding

JFace Data Binding je nástroj, který synchronizuje datový model s uživatelským prostředím[26] a to tak, aby se změny v datovém modelu promítly do uživatelského rozhraní, a naopak, aby se změny v uživatelském rozhraní promítly do datového modelu.

Existují dva typy datových modelů, které můžeme synchronizovat pomocí tohoto frameworku. Datovým modelem typu POJO (Plain Old Java Object) a datový model typu Java Bean. Pokud je datový model popsán modelem POJO, pak objekt nemůže informovat o své změně. Mění se pouze proto datový model na základě změny v uživatelském rozhraní. Datový model typu Java Bean je třída, která musí splňovat jisté podmínky. Musí implementovat všechny metody pro nastavování a získávání parametrů. Dále musí tato třída obsahovat objekt třídy PropertyChangeSupport. Prostřednictvím třídy PropertyChangeSupport musí být datový model aktualizován. Třída PropertyChangeSupport obsahuje metody pro registraci posluchačů. Tyto posluchače dostávají informace o změně datového modelu.

Kapitola 5
Síťový ovladač ulan-java

Tato knihovna umožňuje komunikovat se zařízeními, která jsou k počítači připojena prostřednictvím síťového protokolu uLan. Knihovna je napsaná tak, aby zachovávala asynchronní komunikaci, kterou se protokol uLan vyznačuje. Knihovna podporuje servisní i procesní typy zpráv.

Tato knihovna je napsána v jazyku Java. Knihovna využívá kód, který je napsán v programovacím jazyku C. Tento kód je nutný pro použití systémových volání, které se využívají při komunikaci v síti uLan. Obě dvě části kódu jsou spojeny prostřednictvím JNI. V tomto projektu se používá knihovna Guava[1]), která obsahuje složitější datové struktury.

V následujících sekcích popíši nejdůležitější rozhraní a třídy, které knihovna obsahuje.

5.1 Třída ULanDrv a JNI

Tato třída implementuje rozhraní ULanHandle, které definuje metody pro práci se síťovým protokolem uLan. Rozhraní ULanHandle obsahuje vnitřní statickou třídu ULMsgInfo, která je obdobou struktury ul_msginfo.

V hlavičkovém souboru *net_sourceforge_ulan_base_ULanDrv.h* jsou deklarovány nativní funkce. Tyto nativní funkce jsou implementovány v souboru, který se nazývá *net_sourceforge_ulan_base_ULanDrv.c*. Tyto funkce se volají prostřednictvím nativních metod deklarovaných ve třídě ULanDrv.

Dynamická knihovna se nahrává statickou metodou *loadLibrary(boolean)*. Parametr této metody určuje řetězec, který se vkládá do metody *System.loadLibrary(String)*[2]), která se volá v těle metody *loadLibrary(boolean)*. Řetězec obsahuje hodnotu *ulan_drv_jni* v případě, kdy hodnota parametru metody *loadLibrary(boolean)* je *false*. Pro případ, kdy je hodnota parametru *true*, se hodnota řetězec určuje podle systémových proměnných[3])[4]). Nejdříve se vyhodnotí systémová proměnná *os.name*.

- proměnná *os.name* obsahuje řetězec *win*
  - Jestliže hodnota proměnné *sun.arch.data.model* je rovna 32, pak řetězec má hodnotu *ulan_drv_jni-win32*.
  - Jestliže hodnota proměnné sun.arch.data.model je rovna 64, pak řetězec má hodnotu *ulan_drv_jni-win64*.
  - Jestliže hodnota proměnné sun.arch.data.model má jinou hodnotu, pak řetězec má hodnotu *ulan_drv_jni-win*.

[1]) `https://github.com/google/guava`

[2]) Metoda *System.loadLibrary(String)* nahrává dynamickou knihovnu a podle jejího parametru se určuje název hledané dynamické knihovny

[3]) Všechny systémové proměnné jsou nejprve konvertovány na malá písmena

[4]) Jestliže je hodnota systémové proměnné *os.arch* rovna i386, i486, i586, i686, tak se místo této hodnoty pracuje s hodnotou x86 a pokud je hodnota proměnné os.arch rovna amd64, x86_64, x86-64, emt64t pak se pracuje s hodnotou x86_64

- proměnná *os.name* obsahuje řetězec *mac*
  - K řetězci *ulan_drv_jni-macos-* se přidá hodnota systémové proměnné os.arch, výsledný řetězec může být např. *ulan_drv_jni-macos-x86*
- proměnná *os.name* obsahuje řetězec *linux*
  - K řetězci *ulan_drv_jni-linux-* se přidá hodnota systémové proměnné os.arch , výsledný řetězec může být např. *ulan_drv_jni-linux-x86*
- proměnná *os.name* obsahuje jiný řetězec
  - K řetězci *ulan_drv_jni-* se přidá hodnota systémové proměnné os.name a systémové proměnné *os.arch* spojené pomlčkou

5.2 Třída ULanMsg

Tato třída slouží pro uchovávání dat a parametrů zprávy. Třída obsahuje atribut třídy ByteBuffer, který slouží pro uložení dat zprávy a atribut, který implementuje rozhraní MsgInfo, který obsahuje parametry zprávy. Třída ULanMsg zcela zakrývá objekt třídy MsgInfo. Pro podporu vícerámcové komunikace obsahuje třída ULanMsg atribut třídy ULanMsg. Třída ULanMsg definuje statické metody, které se využívají při příjmu a odeslání zpráv. Statická metoda *getMsgInfo(void)* slouží pro získání atributu třídy MsgInfo, který obsahuje třída ULanMsg. Statická metoda *duplicate(ULanMsg)* slouží pro vytvoření nové instance třídy ULanMsg, nová instance obsahuje duplikát objektu ByteBuffer a stejnou instanci objektu MsgInfo jako původní třída, tato metoda se rekurzí volá na atribut třídy ULanMsg, který třída ULanMsg obsahuje.

5.3 Třída AbstractULanNet

Třída AbstractULanNet slouží ke komunikaci prostřednictvím síťového protokolu uLan. Do jejího konstruktoru se vkládá objekt, který implementuje rozhraní ULanHandle. Třída využívá tento objekt pro síťovou komunikaci a zcela tento objekt zakrývá. Zprávy se posílají metodou *write(ULanDrv)* a přijímají metodou *read(void)*. Metoda *read(void)* má návratovou hodnotu objekt třídy ULanDrv, ve kterém je obsažena příchozí zpráva.

5.4 Třída ULanNet

Tato třída obsahuje atribut třídy AbstractULanNet, který využívá pro komunikaci. Třída ULanNet umožňuje vícevláknový přístup. Aby byl umožněn vícevláknový přístup, třída v sobě obsahuje prioritní frontu požadavků, kterou postupně vykonává. Těmito požadavky jsou čtení příchozí zprávy, odeslání zprávy, či požadavek na ukončení síťové komunikace. Čím má požadavek nižší číslo, tím je priorita požadavku větší. Požadavek na ukončení komunikace má prioritu nula. Po odeslání požadavku na ukončení komunikace se komunikace ihned neukončí, ale počká se na dokončení všech zpráv, které čekají na odpověď a zároveň se již nové zprávy neodešlou. Požadavek pro čtení zprávy má prioritu tři. Požadavek na odeslání zprávy má standartní prioritu čtyři, ovšem priorita může být zvýšena až na hodnotu jedna.

Třída obsahuje metodu pro otevírání, zavírání a kontrolu síťové komunikace. Tyto metody pracují synchronně. Dále v sobě třída obsahuje metody pro asynchronní odeslání a asynchronní příjem zpráv. Třída podporuje procesní i servisní zprávy.

5.4.1 Servisní zprávy

Servisní Zprávy se odesílají asynchronně prostřednictvím metod *addServiceMsg*. Pro určení co má nastat po dokončení zprávy, metody dostávají v parametru objekt implementující rozhraní CompletionHandler a atribut, který se vkládá do metod rozhraní CompletionHandler. Metoda rozhraní CompletionHandler se volá po dokončení zprávy[1]). Kdy je zpráva dokončena, je závislé na tom, zda zpráva obsahuje příznak BFL_M2IN. Příznakem BFL_M2IN zpráva deklaruje, že chceme dostat potvrzení o doručení zprávy. Pokud zpráva neobsahuje tento příznak, předpokládá se, že zpráva je dokončena po úspěšném odeslání zprávy. Pokud zpráva tento příznak obsahuje, pak je zpráva dokončena po přijetí potvrzení o doručení zprávy. Jestliže potvrzení o doručení zprávy nepřijde do určitého časového intervalu[2]), pak se ukončí odeslání zprávy chybou.

Existují dva druhy metody *addServiceMsg*, které se liší podle toho, zda obsahuje prioritní parametr. Pokud voláme metodu, která neobsahuje prioritní parametr, zpráva se zařazuje do fronty událostí s prioritou čtyři. Uživatel by měl přednostně využívat této metody. U zprávy obsahující prioritní parametr může být priorita nastavena až na hodnotu jedna.

5.4.2 Procesní zprávy

Třída obsahuje metody *addfilt*, které slouží pro příjem a zpracování procesních zpráv. Do těchto metod se vkládá parametr třídy CompleteMsg. Parametr třídy CompleteMsg slouží ke zpracování příchozí procesní zprávy. Třída CompleteMsg obsahuje atribut implementující rozhraní CompletionHandler, jehož metody se volají po příjmu procesní zprávy [3]), a atribut, který se vkládá do metod rozhraní CompletionHandler. Dvěma způsoby lze nastavit, které procesní zprávy se mají zpracovávat instancí třídy CompleteMsg. Lze nastavit zpracování příchozích zpráv, které vysílá konkrétní zařízení na určitém portu. Nebo lze také nastavit zpracování zpráv ze všech zařízení, která vysílají na určitém portu.

5.5 Třída DeviceDescription

Tato třída slouží k popisu jednotlivých zařízení. Třída v sobě obsahuje informace o adrese zařízení a popisu zařízení.

5.6 Třída ULanCommunicationInterface

Třída slouží pro odeslání servisních zpráv a pro zpracování příchozích procesních zpráv. Tato třída je napsána tak, aby každá instance této třídy mohla být použita jiným vláknem, a tudíž mohou jednotlivé instance nezávisle na sobě komunikovat se zařízeními.

Třída UlanCommunicationInterface využívají pro komunikaci statický atribut třídy ULanNet, který zcela zakrývá. Třída ULanCommunicationInterface obsahuje statické i instanční metody. Statická metoda *setHandle(ULanHandle)* slouží k inicializaci atributu implementující rozhraní ULanHandler. Bez inicializace tohoto atributu nelze komunikovat. Dále třída obsahuje statické metody pro otevření a zavření síťové komunikace.

[1]) Metoda *completed* se volá po úspěšném dokončení zprávy. Do parametru *result* této metody je uložena odpověď na vyslanou zprávu nebo původní zpráva, jestliže není požadována odpověď. Metoda *faild* se volá při neúspěšném dokončení zprávy, nebo neúspěšném odeslání.
[2]) Na potvrzení zprávy se čeká čtyři sekundy.
[3]) Metoda *completed* se volá při příjmu procesní zprávy.Do parametru *result* této metody je uložena přijatá zpráva. Metoda *faild* se volá, jestliže se v síťové komunikaci vyskytne chyba.

Třída obsahuje také statické metody, které vyhledávají zařízení připojené k síti. Instanční metody třídy ULanCommunicationInterface jsou implementací metod, které jsou definované pomocí rozhraní ULanCommunication.

5.7 Rozhraní IULanCommunication

Rozhraní ULanCommunication definuje metody pro síťovou komunikaci.

Pro komunikaci prostřednictvím servisních zpráv se využívá asynchronních metod *sendMsg*, *sendCommand* [1]) a *sendQuery*[2]). Pro dokončení zprávy tyto metody dostávají jako parametr objekt implementující rozhraní CompletionHandler a objekt, který se vkládá do metod rozhraní CompletionHandler [3]).

Příjem a zpracování procesních zpráv je podporováno prostřednictvím metod *addFilt* a *addFiltAdr*. Tyto metody dostávají jako vstupní parametry objekt implementující rozhraní CompletionHandler a objekt, který se vkládá do metod rozhraní CompletionHandler. Tyto parametry slouží ke zpracování příchozí zprávy [4]). Tyto metody také dostávají parametry, které určují jaký typ zprávy se má zpracovávat. Metody vrací objekt implementující rozhraní IFilt. Rozhraní IFilt slouží pro aktivaci a deaktivaci zpracování příchozích zpráv. Všechny instance objektu IFilt se po vytvoření automaticky uloží do datové struktury. Tato datová struktura je přístupná pomocí metody *getFilts(void)*. Rozhraní také definuje metodu pro hromadnou aktivaci a deaktivaci filtrů.

5.8 Třída ULanDevice

Tato třída je potomkem třídy ULanCommucationInterface. Třída ULanDevice slouží především pro komunikaci s konkrétním zařízením. Třída obsahuje atribut třídy DevicedDescription, který slouží pro definování konkrétního zařízení, se kterým bude instance této třídy komunikovat. Tato třída implementuje rozhraní IULanDevice, které rozšiřuje rozhraní IULanCommunication. Rozhraní IULanDevice definuje metody pro komunikaci s konkrétním zařízením.

[1]) Tato metoda by měla odpovídat funkci *send_command_wait*, jejíž vzorová implementace je uvedena na webu[9]

[2]) Tato metoda by měla odpovídat funkci *send_query_wait*, jejíž vzorová implementace je uvedena na webu[9].

[3]) Způsob dokončení zprávy je shodný se způsobem dokončení zprávy metodami addServiceMsg, které jsou deklarovány ve třídě ULanNet. Metoda *sendCommand* a *sendQuery* čeká na potvrzení o doručení zprávy

[4]) Metoda rozhraní *completed*, která je definovaná v rozhraní CompletionHandler, se volá při příjmu procesní zprávy, metoda *faild*, která je definovaná v rozhraní CompletionHandler, se volá, jestliže se v síťové komunikaci vyskytne chyba

Kapitola 6
Rozšíření do programu OpenChrom

Výsledkem tohoto projektu je možnost akvizice dat prostřednictvím programu OpenChrom. Akvizice dat se provádí v nově vytvořené perspektivě. Tato perspektiva se nazývá CHROMuLAN. Perspektiva slouží pro akvizici chromatografických dat zařízeními, které komunikují prostřednictvím síťového protokolu uLan. V perspektivě je nyní možné zobrazit data z detektoru v reálném čase a také tato data zaznamenat. Po dokončení akvizice lze uložit data do formátů, které podporuje program OpenChrom. Nyní je přidána podpora pro zařízení ULAD 31 a ULAD 32, ale projekt je postaven tak, aby bylo možné přidat podporu i pro jiná zařízení. Například podporu celých HPLC sestav od firmy Pikron. Protože v současné době program OpenChrom podporuje pouze zobrazení dat získaných měřením vlnového spektra a neobsahuje žádné nástroje na jejich zpracování, jsou veškerá data ukládána jako data z ionizačního detektoru.

Tento projekt rozšiřuje program OpenChrom o čtyři nové komponenty

- org.chromulan.system.control – komponenta obsahující objektový model
- org.chromulan.system.control.ui – komponenta definující uživatelské rozhraní
- org.chromulan.system.control.net – komponenta obsahující knihovnu ulan-java
- org.chromulan.system.control.ulad3x – komponenta obsahující podporu pro zařízení ULAD 31 a ULAD 32

Tyto komponenty jsou shromážděny v logické jednotce, která se jmenuje org.chromulan.system.control.feature.

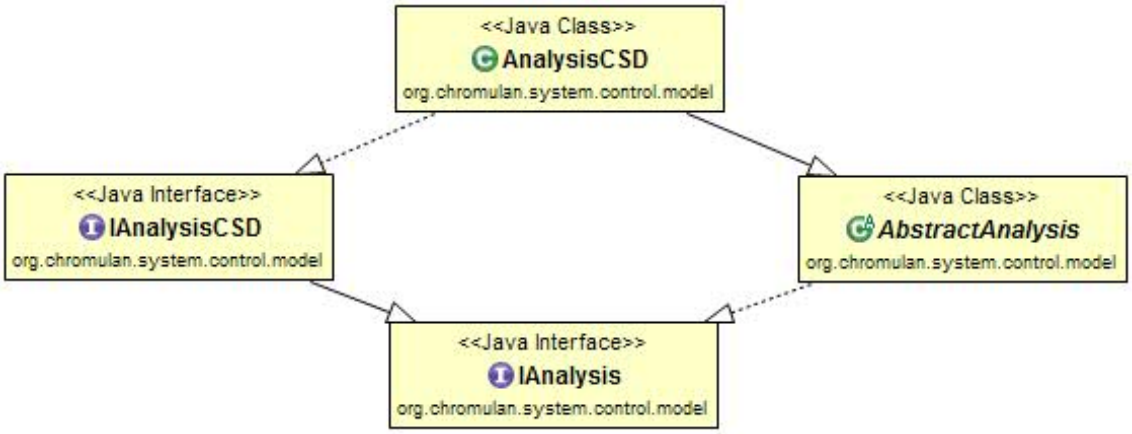
6.1 Popis Komponent

6.1.1 Komponenta obsahující objektový model

Tato komponenta obsahuje obecný objektový model. Objekty z tohoto modelu se využívají při každém typu akvizice chromatografických dat. Dále komponenta obsahuje objektový model určený pro akvizici dat z iontového detektoru, který je odvozen od obecného modelu.

Obecný objektový model je navržen tak, aby bylo jednoduché přidat podporu zaznamenávání dat z detektorů, které měří hmotnostní či vlnové spektrum. Příklad konceptu objektového modelu uvedu na příkladu třídy, jejíž instance reprezentují akvizici. Všechny instance třídy, které zastupují akvizici, mají společné rozhraní IAcquisition. Abstraktní třída AbstractAcquisition implementující rozhraní IAcquisition obsahuje společné metody pro všechny typy akvizic. Pro konkrétní druh akvizice je vytvořeno zvláštní rozhraní (např. IAcquisitionCSD pro akvizici dat z iontového detektoru). Toto rozhraní rozšiřuje obecné rozhraní IAcquisition. Třída AcquisitionCSD reprezentuje konkrétní druh akvizice, je potomkem obecné abstraktní třídy AbstracAcquisition a implementuje rozhraní IAcquisitionCSD. Tento objektový návrh je použit pro třídy,

které implementují rozhraní IAcquisition, IAcquisitions, IAcquisitionSaver a IChromatogramAcquisition.



Obrázek 6.1. UML diagram třídy AcquisitionCSD

6.1.2 Komponenta definující uživatelské rozhraní

Tato komponenta rozšiřuje aplikační model o nové grafické a negrafické prvky.

6.1.3 Komponenta obsahující knihovnu ulan-java

Tato komponenta obsahuje Java knihovnu ulan-java a dynamické knihovny. Tyto knihovny umožňují komunikovat se zařízením prostřednictvím síťového protokolu uLan.

6.1.4 Komponenta obsahující podporu pro zařízení ULAD 31 a ULAD 32

Tato komponenta rozšiřuje aplikační model o nové grafické a negrafické prvky. Tyto prvky slouží pro kontrolu zařízení ULAD 31 a ULAD 32. Komponenta umí získávat dat z analogově digitálního převodníku, které tato zařízení obsahují.

6.2 Popis obecných rozhraní

6.2.1 Rozhraní IAcquisition

Toto rozhraní rozšiřuje rozhraní IAcquisitionProcces. Rozhraní IAcquisition shromažďuje základní údaje o akvizici, jako například jméno, popis, délku akvizice a také obsahuje možnost registrovat posluchače, kteří se aktivují při změně těchto údajů. Posluchači těchto proměnných se registrují prostřednictvím řetězce a objektu implementujícího rozhraní PropertyChangeListener. Tyto řetězce jsou definovány jako konstanty ve třídě IAcquisition. Rozhraní IAcquisitionProcces obsahuje atribut implementující rozhraní IAcquisitionSaver a atribut implementující rozhraní IProfileDevices. Rozhraní IAcquisitionSaver definuje metody, které slouží pro ukládání naměřených dat do souboru. Rozhraní IProfilDevice obsahuje objekty, které reprezentují jednotlivá zařízení sloužící pro akvizici. Rozhraní IAcquisitionProcces dále definuje metody, které určují, v jaké fázi se daná akvizice nachází. Akvizice může být připravená, spuštěná a dokončená. Stav akvizice se mění prostřednictvím metod *start(void)* a *stop(void)*.

6.2.2 Rozhraní IAcquisitions

Toto rozhraní umožňuje shromažďovat akvizice v datové struktuře. Navíc toto rozhraní obsahuje ukazatel na aktuální akvizici. Aktuální akvizice se nastavuje automaticky jako první akvizice vložená do struktury. Aktuální akvizice je umožněno pomocí metod definované v rozhraní IAcquisitions měnit.

6.2.3 Rozhraní IAcquisitionSaver

Toto rozhraní umožňuje ukládat akvizici do formátů, které podporuje program OpenChrom. Rozhraní obsahuje následující atributy. Atribut složky, do které se má akvizice uložit a atribut implementující rozhraní ISuplier. Pomocí tohoto atributu se určuje do jakého formátu se má daná akvizice uložit. Pomocí metody *save(IProgressMonitor, IChromatograMaker)* lze uložit všechny chromatogramy, které vrací metoda *getChromatograms(void)*. Tato metoda je definována v rozhraní IChromatograMaker.

6.2.4 Rozhraní IControlDevice, IControlDevices a IDevicesProfile

Rozhraní IControlDevice nese informace o zařízení, které komunikuje prostřednictvím síťového standartu uLan. Rozhraní IControlDevice obsahuje atribut třídy DeviceDescription. Dále obsahuje atributy jméno a typ zařízení, pro tyto atributy lze registrovat posluchače obdobným způsobem jako posluchače pro atributy rozhraní IAcquisition. Atribut typ zařízení je výčtovým typem DeviceType. Rozhraní IControlDevice obsahuje dále metodu *getId(void)*, která vrací pro každé zařízení unikátní řetězec (pro stejná zařízení vrací tato metoda stejné řetězce). Rozhraní IControlDevice také nese informaci o tom, zda je zařízení připojeno k síti. Rozhraní IControlDevices shromažďuje v datové struktuře, kterou obsahuje, objekty implementující rozhraní IControlDevice. Rozhraní IDevicesProfile definuje zařízení, která se podílejí na akvizici. Toto rozhraní obsahuje název profilu a atribut implementující rozhraní IControlDevices.

6.2.5 IAcquisitionData, IDeviceData, IDetectorData

Tato rozhraní slouží pro shromažďování a popis dat, které se získávají při akvizici. Rozhraní IAcquisitionData obsahuje atributy pro popis dat. Rozhraní IDevicesData rozšiřuje rozhraní IAcquisitionData. Rozhraní IDeviceData obsahuje atribut implementující rozhraní IControlDevice. Tento atribut slouží k určení z jakého zařízení se data získala. Rozhraní IDevicesData se dále rozšiřuje pro konkrétní typy zařízení. Pro shromažďování dat z detektorů je určeno rozhraní IDetectorData.

6.2.6 Rozhraní IChromatogramAcquisition

Toto rozhraní umožňuje vícevláknový přístup k datům chromatografického záznamu. Rozhraní IChromatogramAcquisition je určené pro to, aby bylo možné data ukládat z jednoho vlákna a zároveň z druhého vlákna tato data zobrazovat. Metoda *getChromatogram(void)* slouží k získání uložených dat v podobě objektu implementujícího rozhraní IChromatogram. Metoda *newAcquisition(int,int)* zahajuje nový sběr dat. Jestliže chce uživatel provádět nějaké změny v objektu, který vrací metoda *getChromatogram(void)*, musí tento objekt uzavřít do synchronizované sekce následujícím způsobem.

```
IChromatogram chromatogram = chromatogramAcquisition.getChromatogram();
synchronized(chromatogramAcquisition)
{
zde je možné bezpečně pracovat s objektem chromatogram
}
```

6.3 Řídící události

V následujících sekcích se budu věnovat událostem, které se používají pro komunikaci mezi jednotlivými komponentami. Tyto události se posílají a přijímají pomocí služby Eclipse event system, kterou definuje framework Eclipse RCP. Každá událost obsahuje textový řetězec a objekt. Podle typu objektu a obsahu řetězce lze registrovat posluchače pro příjem konkrétní události.

6.3.1 Události, které se vysílají při změně stavu akvizice

Každá akvizice se může dostávat do několika stavů. Při změně stavu akvizice se vysílá událost. Tato událost obsahuje textový řetězec, který je definovaný v rozhraní IAcquisitionEvents a objekt implementující rozhraní IAcquisition, ke kterému se daná akvizice vztahuje.

Stavy akvizice se řídí pomocí následujících událostí.

- TOPIC_ACQUISITION_CHROMULAN_SET
  acquisition/chromulan/set
  Událost nastaví akvizici jako aktuální.
- TOPIC_ACQUISITION_CHROMULAN_END
  acquisition/chromulan/end
  Událost ruší nastavení akvizice jako aktuální.
- TOPIC_ACQUISITION_CHROMULAN_START_RECORDING
  acquisition/chromulan/startrecording
  Událost značí začátek zaznamenávání dat.
- TOPIC_ACQUISITION_CHROMULAN_STOP_RECORDING
  acquisition/chromulan/stoprecording
  Událost značí konec zaznamenávání dat.

6.3.2 Události, které se vysílají při změně stavu síťového spojení uLan

Tyto události obsahují textový řetězec, který je definovaný v rozhraní IULanConnectionEvents a objekt třídy ULanConnection.

- TOPIC_CONNECTION_ULAN_CLOSE
  connection/ulan/close
  Událost síťové spojení zavřeno – událost je vysílána při zavření síťové komunikace.
- TOPIC_CONNECTION_ULAN_OPEN
  connection/ulan/open
  Událost síťové spojení otevřena – událost je vysílána při otevření síťové komunikace.

6.3.3 Události, kterými se vysílá stav zařízení

Události, při kterých se odesílá objekt implementující rozhraní IControlDevice a řetězec, který je definován v rozhraní IControlDeviceEvents. Události se nemusí posílat pouze při změně stavu zařízení.

- TOPIC_CONTROL_DEVICE_ULAN_CONNECT
  controlDevice/ulan/connect
  Událost zařízení připojeno – událost značí, že zařízení je připojeno.

- TOPIC_CONTROL_DEVICE_ULAN_DISCONNECT
  controlDevice/ulan/disconnect
  Událost zařízení odpojeno – událost značí, že zařízení je odpojeno.

6.3.4 Události, vztahující se ke skupině zařízení

Události, při kterých se odesílá objekt implementující rozhraní IControlDevices a řetězec, který je definován v rozhraní IControlDevicesEvents

- TOPIC_CONTROL_DEVICES_ULAN_AVAILABLE
  controlDevices/ulan/available
  Událost stav zařízení - událost posílá aktuální stav všech připojených zařízení. Tato zařízení jsou uložena v objektu, který je vysílán v této událostí.
- TOPIC_CONTROL_DEVICES_ULAN_CONTROL
  controlDevices/ulan/control
  Událost kontrola zařízení - událost se vysílá jako požadavek na zkontrolování stavu zařízení, například kontrolu zda jsou zařízení připojena. Zařízení ke kontrole jsou uložena v objektu, který je vysílán v této události.

6.4 Třídy určující chování panelů

6.4.1 Třída AvailableDevicesPart

Tato třída definuje chování panelu, který spravuje všechna zařízení připojená k síti. Vysílá událost *stav zařízení* s objektem, který obsahuje aktualizovaná data o stavu zařízení. Jestliže bude tuto událost vysílat i jiná komponenta, program se dostane do nedefinovaného stavu.

Panel přijímá událost *kontrola zařízení*. Po přijetí této události zkontroluje stav zařízení, které dostane prostřednictvím této události. Poté co zkontroluje stav zařízení, vyšle tento panel událost *stav zařízení*.

Panel při nalezení připojeného zařízení vysílá událost *zařízení připojeno* a při odpojení zařízení vysílá událost *zařízení odpojeno*. Panel také vysílá událost *síťové spojení otevřeno* a událost *síťové spojení uzavřeno*. Všechny tyto události by měly být vysílány pouze tímto panelem.

V kontextu panelu je uložen objekt implementující rozhraní IControlDevices, který obsahuje všechna zařízení, která byla od spuštění aplikace připojena. Tento objekt je totožný s objektem, který se vysílá při události *stav zařízení*. Tento objekt je uložen v kontextu pod názvem rozhraní IControlDevices.

6.4.2 Třída DevicesProfilesPart

Tato třída definuje chování panelu, ve kterém jsou spravovány profily. Profil obsahuje seznam zařízení.

V kontextu panelu je uložen list objektů implementující rozhraní IProfileDevices, ve kterém jsou obsaženy všechny vytvořené profily. Tento objekt je uložen v kontextu pod řetězcem *Profiles*.

6.4.3 Třída AcuisitionsPart

Tato třída definuje chování panelu, ve kterém jsou spravovány akvizice. Tento panel vysílá události, které určují stav akvizice, tyto události vysílá synchronně. Nepředpokládá

se, že nikdo jiný bude tyto události vysílat a také nikdo jiný nezavolá metody *start(void)* a *stop(void)*, které obsahuje objekt akvizice (Tyto metody jsou definované v rozhraní IAcquisitionProcces). Pokud tak nastane, akvizice se dostane do nedefinovaného stavu. Pro zaznamenání akvizice jsou události odeslány v tomto pořadí.

- Odeslání události, která nastavuje akvizici jako aktuální.
- Odeslání události značící začátek nahrávání dat.
- Odeslání události značící konec nahrávání dat.
- Odeslání události, které zruší natavení události jako aktuální

Akvizice může být po nastavení jako aktuální zrušena, bez toho aniž by se začala nahrávat data.

Tato třída nastavuje nanejvýš jednu akvizice jako aktuální. Před nastavením akvizice jako aktuální se vyšle synchronní událost *zkontroluj zařízení*. Jestliže nejsou poté zařízení připojena, pak se akvizice nenastaví jako aktuální. Kontrola, zda jsou připojena všechna zařízení, a tudíž se může nastavit událost jako aktuální, se znovu provádí po zachycení události *stav zařízení*. Aktuální akvizice je nastavena do kontextu perspektivy CHROMuLAN pod názvem rozhraním IAcquisition.

6.5 Funkční prvek umožňující zobrazení dat v reálném čase

Funkční prvek přijímá událost, ve které je objekt implementující rozhraní IChromatogramCSDAcquisition a řetězec, který je definován v rozhraní IAcquisitionUIEvents pod názvem TOPIC_ACQUISITION_CHROMULAN_UI_CHROMATOGRAM_DISPLAY.

Na základě této události vytvoří funkční prvek nový panel. V panelu jsou zobrazována data z objektu, který dostane funkční prvek v události. Zobrazená data jsou pravidelně aktualizována.

6.6 Komponenta obsahující podporu pro zařízení ULAD 31 a ULAD 32

Tato komponenta obsahuje funkční rozhraní, které vytváří při prvním připojení každého zařízení nový panel. Funkční rozhraní tento panel vytvoří na základě přijetí události *zařízení připojeno*. Nově vytvořený panel musí mít stejné id jako návratová hodnota metody *getID(void)*. Tuto metodu obsahuje objekt, který je posílán s událostí *zařízení připojeno* (V této události je posílán objekt implementující rozhraní IControlDevice). Funkční rozhraní přidá nově vytvořený panel do zásobníku panelů, který má identifikátor *org.chromulan.system.control.ui.partstack.devicesSetting*. Tento zásobník se zobrazuje se až poté, co se vytvoří první panel pro zařízení. Vytvořený panel nelze zavřít.

Při příjmu události, která nastavuje akvizici jako aktuální, se panel vytvořený pro zařízení rozhoduje, zda se bude zařízení na akvizici podílet. Rozhoduje se podle toho, jestli akvizice obsahuje profil (objekt implementující rozhraní IProfileDevices), ve kterém je uloženo zařízení, které má stejnou návratovou hodnotu metody *getID(void)*. Jestliže se zařízení bude podílet na akvizici, pak začne zaznamenávat data po příjmu události značící začátek zaznamenání dat a ukončuje zaznamenávání dat po události značící konec zaznamenávání dat.

Po příjmu události značící konec zaznamenávání dat se také uloží data do objektu, který implementuje rozhraní IDetectorData, a tento objekt se uloží do datové struktury implementující rozhraní List. Tato datová struktura se uloží do panelu do datové

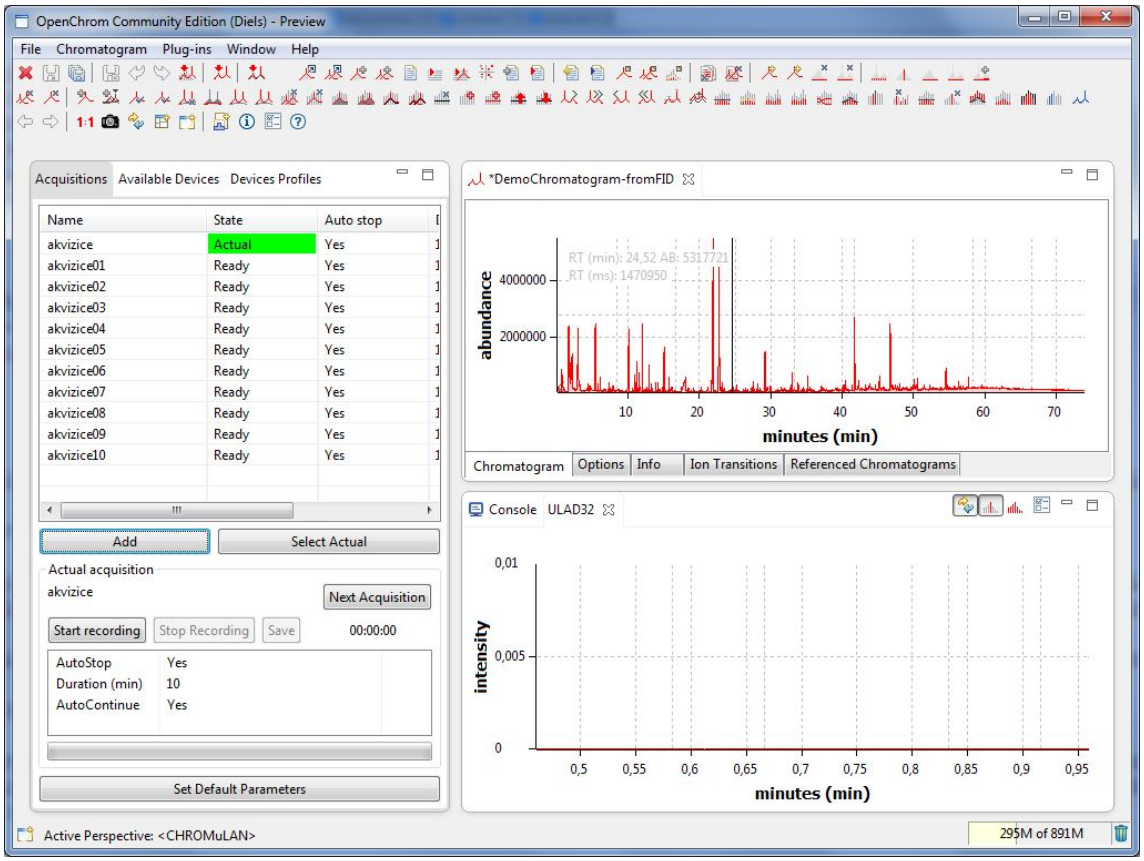
struktury Transient Data pod klíčem *detectorsData*. Tento klíč je definován v rozhraní IDetectorData. Předešlý proces uložení dat ukáži ve zdrojovém kódu.

```
List list = new LinkedList();
list.add(data);  \\ objekt data implementuje rozhraní IDetectorData
\\ a jsou v něm uložena data z detektoru
part.getTransientData().put(IDetectorData.DETECTORS_DATA, list);
```

Tato komponenta také nastavuje atribut typ zařízení v objektu, který dostane prostřednictvím události *zařízení připojeno*.

Kapitola 7
Uživatelské rozhraní vytvořeného rozšíření

Pro sběr dat byla přidána do programu OpenChrom nová perspektiva, která se jmenuje CHROMuLAN. Tato perspektiva sdružuje panely, které se podílejí na akvizici. V následujících sekcích popíši jednotlivé panely.

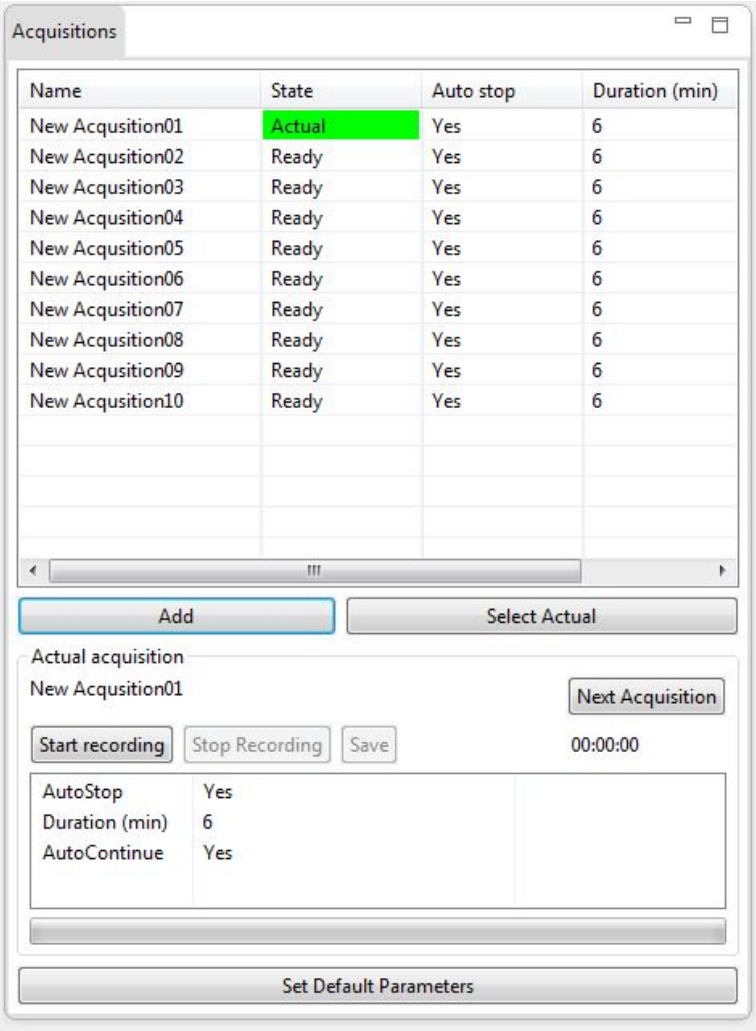


Obrázek 7.1. Perspektiva CHROMuLAN

7.1 Panel Acqusitions

Tento panel slouží ke správě akvizic a také k záznamu a uložení dat akvizice. Panel obsahuje tabulku, ve které jsou zobrazeny všechny akvizice. Panel také obsahuje tlačítko *set deafult parameters*, které slouží pro nastavení společných parametrů pro všechny akvizice. Po kliknutí na toto tlačítko se otevře průvodce, ve kterém se nastaví složka, do které se budou data ukládat a také se nastaví výchozí formát, do kterého se budou data z akvizice ukládat. Nová akvizice se přidá tlačítkem *add*, jestliže se akvizice nenahrává, lze jí po označení v tabulce smazat stiskem klávesy *Delete*.

Pro přidání nové akvizice se spustí průvodce, který se skládá z několika částí.



Obrázek 7.2. Panel Acqusitions

- v první části se vybere profil, který obsahuje zařízení, která se budou podílet na akvizici.
- v druhé části se nastaví základní parametry akvizice
  - Nastaví se jméno akvizice. (Name)
  - Nastaví se, zda má být po skončení nahrávání dat akvizice automaticky uložena a automaticky nastavena další akvizice jako aktuální. (Auto Continue)
  - Jestliže je nastaveno automatické ukončení nahrávání dat (Auto Stop), tak se musí nastavit doba, za kterou se ukončí nahrávání(Duration). Tato doba se nastavuje v minutách.
  - Do akvizice se mohou vložit poznámky, například druh zkoumaného vzorku. (Description)
- V třetí části se nastaví, kolik akvizic s těmito parametry se má vytvořit. Jestliže je nastavena více než jedna akvizice, pak nově vytvořené akvizice budou mít všechny stejné parametry až na jméno, které se vytvoří tak, že se k základnímu jménu přidá pořadová číslice.

Parametry akvizice můžeme měnit i po jejím vytvoření. Parametry akvizice se mohou měnit po dvojkliku na danou akvizici. Můžeme měnit základní údaje o akvizici a také

lze změnit složku, kam se akvizice uloží, a formát, do kterého se akvizice uloží. Tyto údaje můžeme měnit do doby, než se ukončí nahrávání dat.

Po dokončení akvizice a uložení dat do chromatogramu lze otevřít soubory, ve kterých je výsledný chromatogram uložen. Tento soubor se otvírá po dvojkliku na danou akvizici.

Jednotlivé akvizice lze postupně vykonávat. Akvizice, která se právě vykonávaná, se nazývá aktuální akvizice.

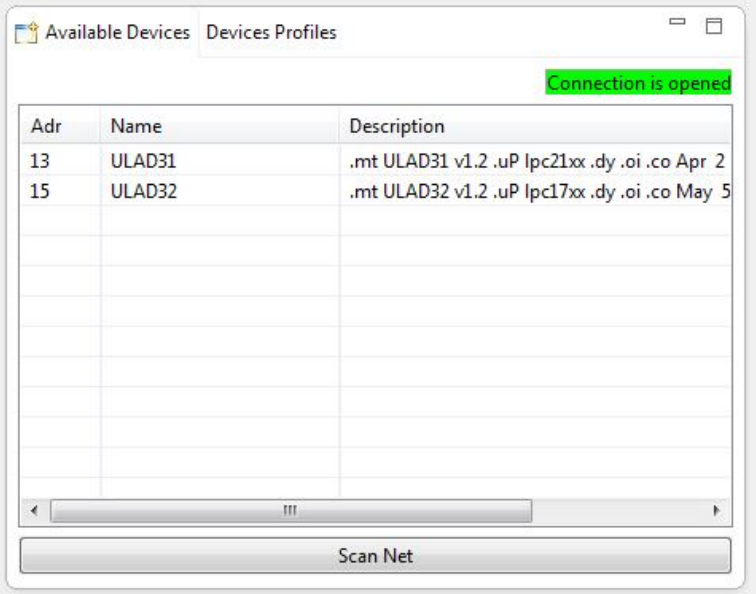
V tomto panelu je také definováno uživatelské rozhraní aktuální akvizice. Zobrazují se zde parametry o aktuální akvizici a také se zde zobrazuje doba, která uplynula od zahájení nahrávání dat. Začátek akvizice dat se může zahájit tlačítkem *start*, nebo se může zahájit příjmem značky pro začátek akvizice ze sítě uLan[1]). Ukončení akvizice dat je možné buď automaticky, nebo tlačítkem *stop*. Tlačítkem *save* se nahraná akvizice uloží a tlačítkem *Next aqusition* se nastaví další akvizici jako aktuální.

Vzniklý chromatogram, který vznikne akvizicí dat, se ukládá na místo, které určí uživatel do nově vytvořené složky. Nová složka má stejné jméno jako akvizice. Soubor obsahující chromatogram má stejný název zařízení, ze kterého jsou data pořízena.

7.2 Panel Available Devices

Panel zobrazující stav síťového připojení a zařízení připojená k síti.

Obsahuje tabulku ve, které se zobrazují zařízení připojená k síti. V tabulce je možné upravit jméno zařízení. Každé zařízení by mělo mít unikátní jméno[2]). Tlačítko *Scan Net* slouží pro otevření a vyhledání všech zařízení připojených k síti uLan.



Obrázek 7.3. Panel Available Devices

7.3 Panel Devices Profiles

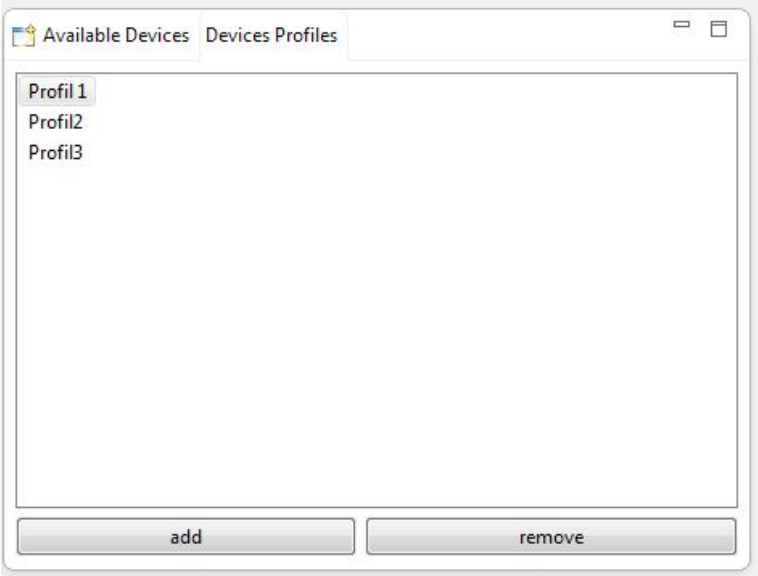
Panel slouží pro správu profilů. Profily seskupují zařízení, které se mají podílet na akvizici.

[1]) Tuto značka mohou vysílat převodníky ULAD 31 a ULAD 32.

[2]) Jestliže budou mít zařízení stejná jména, vzniká nebezpečí, že jednotlivá zařízení si přepíší uložená data.

Tento panel obsahuje tabulku, ve které jsou zobrazeny všechny profily. Rozhraní umožňuje přidat nový profil kliknutím na tlačítko *add* a smazat profil označením profilu a zmáčknutí tlačítka *remove*.

Po kliknutí na tlačítko add se otevře průvodce. Tento průvodce obsahuje dvě položky. Nastavení jména profilu a seznam všech zařízení, která byla připojena k síti. Z tohoto seznamu uživatel vybírají zařízení, které bude obsahovat nově vytvořený profil. Po dvojkliku na vytvořený profil se zobrazí zařízení, která tento profil obsahuje.

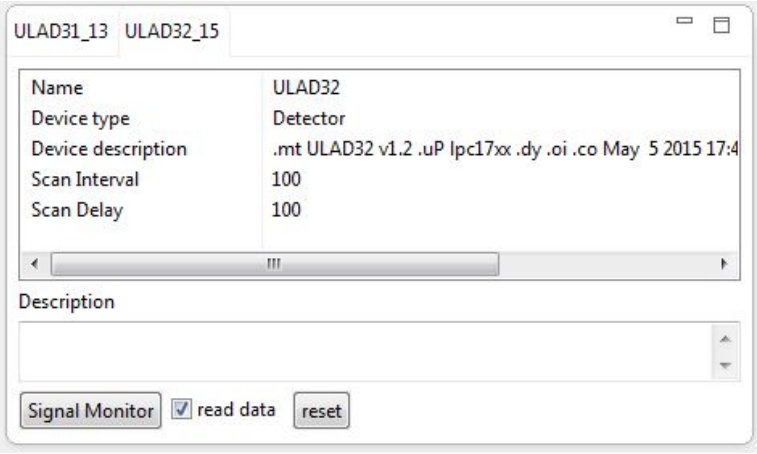


Obrázek 7.4. Panel Devices Profiles

7.4 Panel zobrazující nastavení zařízení ULAD 31 a ULAD 32

Tento panel slouží především k zaznamenání dat z analogově digitálního převodníku. Později bude sloužit i k nastavení parametrů převodníku. Zaznamenaná data lze zobrazit tlačítkem *signal monitor*.

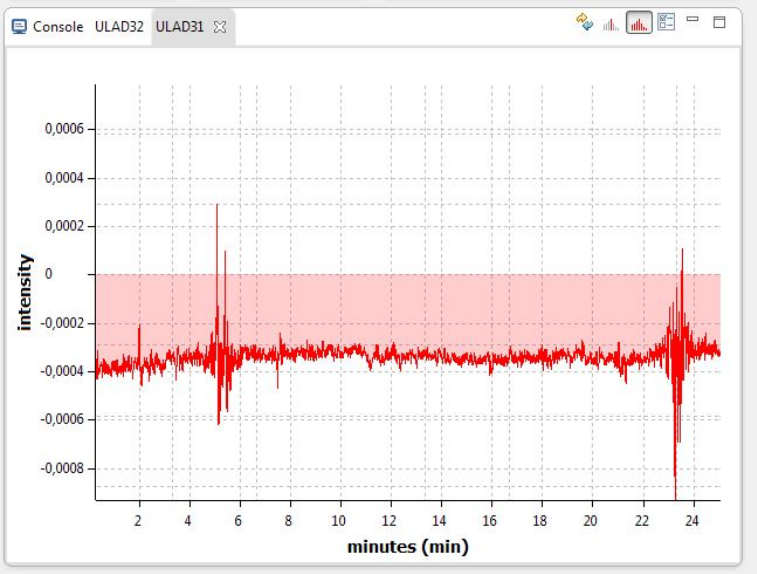
Jestliže neprobíhá akvizice dat lze rozhodnout, jestli se budou data ukládat tlačítkem *read data*, či zaznamenaná data vymazat tlačítkem *reset*.



Obrázek 7.5. Panel zobrazující nastavení zařízení ULAD 31 a ULAD 32

7.5 Panel zobrazující aktuální data ze zařízení

Tento panel umožňuje zobrazovat chromatogram při akvizici dat. Panel zobrazuje data ve dvou módech. První mód zobrazuje všechna data. Ve druhém módu se zobrazují data, která byla zaznamenána naposledy. Také je možné nastavit to, zda se budou data automaticky překreslovat. V nastavení lze určit, jaká má být nejmenší velikost zobrazené plusové osy Y, či jak velký má být časový úsek, který se ukazuje při zobrazení posledních dat.



Obrázek 7.6. Panel zobrazující aktuální data ze zařízení

Kapitola 8
Testování

Testování probíhalo na operačním systému Windows 7 v jeho 64 bitové variantě, na 64 bitové vituálním stroji[1]). Jako testovací zařízení sloužily analogově digitální převodníky ULAD 31 a ULAD 32.

8.0.1 Testování ovladače ulan-java

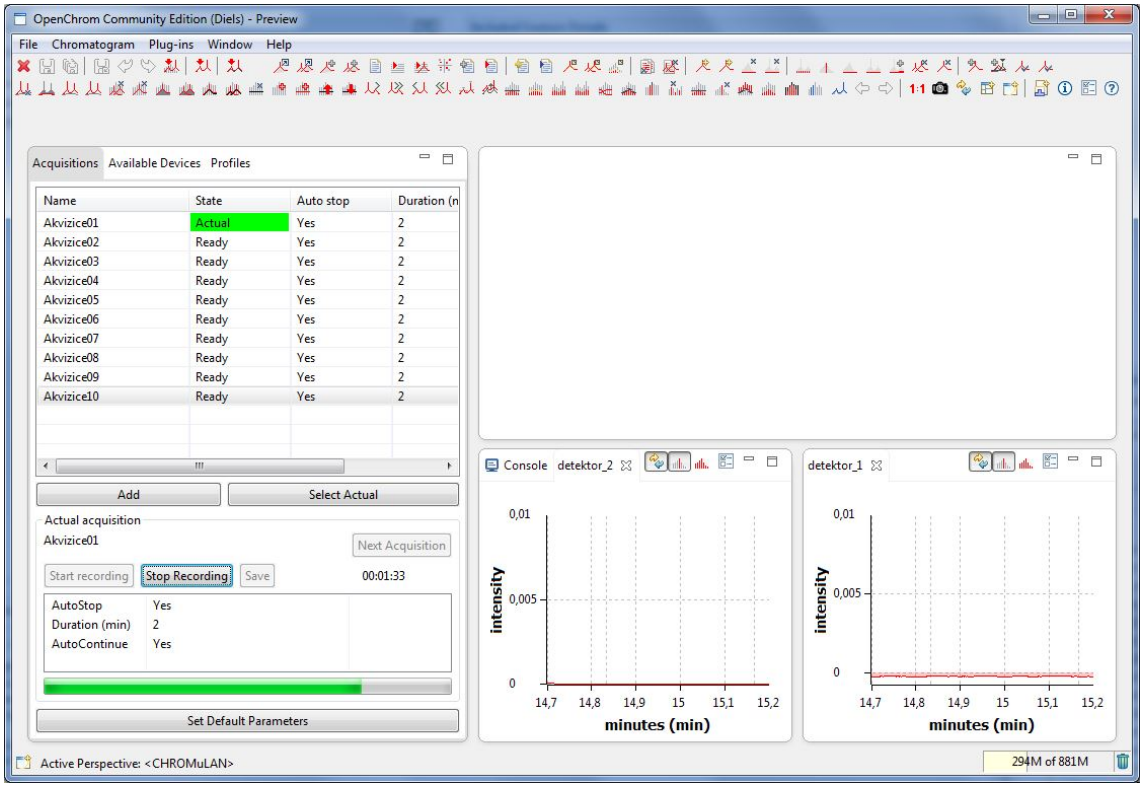
Byla vyzkoušena podpora procesních i servisních zpráv. Potvrdilo se, že knihovna umožňuje komunikovat s více zařízeními najednou. Komunikace prostřednictvím servisních zpráv byla vyzkoušena vyhledáváním zařízení na síti. Příjem procesních zpráv byl vyzkoušen na přijímání zpráv, které jsou vysílány z analogově digitálního převodníku ULAD 31 a ULAD 32. Knihovna si poradila i s vyšší zátěží sítě, které bylo dosaženo zvýšením vzorkovací frekvence analogově digitálního převodníku. Byl také úspěšně vyzkoušen vícevláknový přístup k instanci třídy ULanNet. Ve třídě main jsou definovány Testovací metody, kterými lze knihovnu otestovat.

8.0.2 Testování rozšíření programu OpenChrom

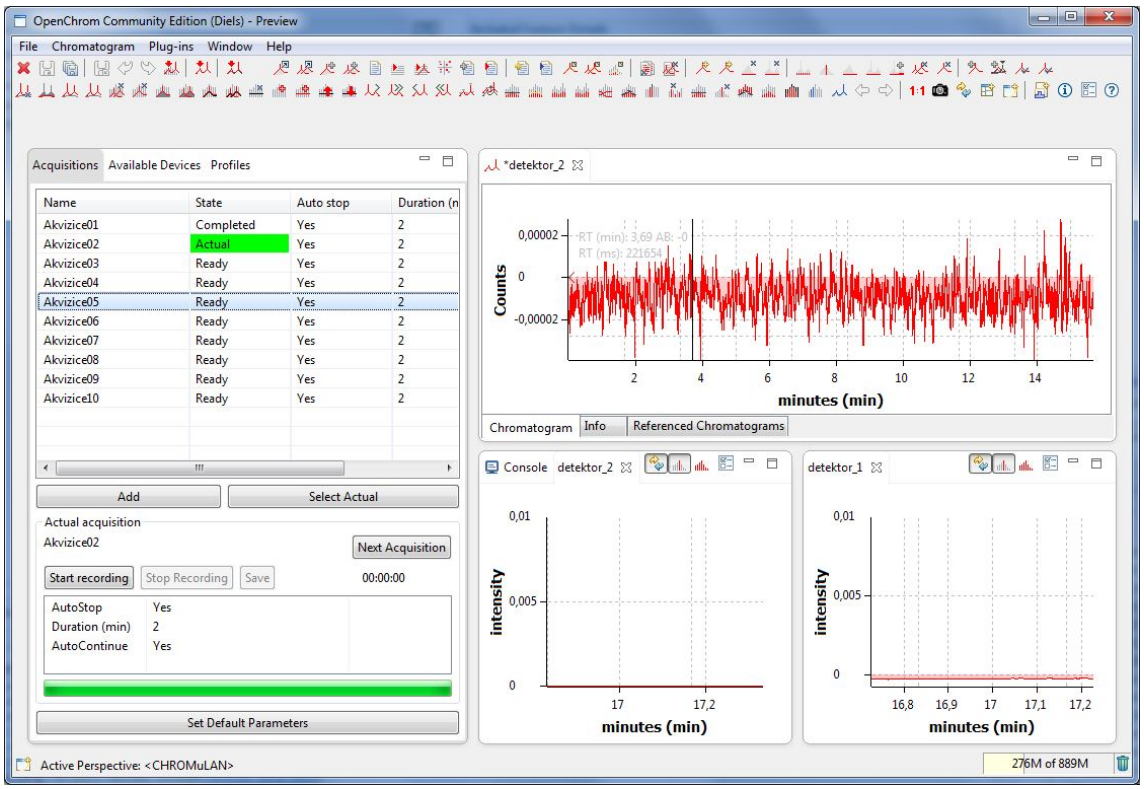
Testování probíhalo zkoušením funkčnosti uživatelského rozhraní. S úspěchem byly vyzkoušeny všechny funkce programu. Při běžném používání nově vytvořeného rozšíření funguje vše bez problému.

Při testování rozšíření se vyskytl problém, který neomezuje běžné využívání rozšíření. Problém nastal s odezvou panelu, který zobrazuje data v reálném čase. Tento problém nenastane při obvyklých délkách chromatografického záznamu, při kterém se používá typická vzorkovací frekvence. Obvyklá vzorkovací frekvence je 10 až 100 vzorků za sekundu. Při delších záznamech se vzorkovací frekvence snižuje a při kratších záznamech se vzorkovací frekvence zvyšuje. Mezi nejdelší typy záznamů patří analýza aminokyselin v látce, tato analýza probíhá až 3 hodiny. Při analýze, která probíhá 3 hodiny se vzorkovací frekvencí 10 vzorků za sekundu, modul zobrazuje data bez problému. Problém s odezvou panelu nastane až při záznamu čítající milióny vzorků.

[1]) V projektu ovladače i v projektu rozšíření do programu OpenChrom jsou přidány nativní knihovny pro podporu operačního systému Windows i Linux. Pro oba dva operační systémy jsou přidány nativní knihovny pro podporu 32 bitového virtuálního stroje i 64 bitového virtuálního stroje.



Obrázek 8.1. Zaznamenávání dat v pespektivě CHROMuLAN



Obrázek 8.2. Ukázka nově vytvořeného chromatografu

Kapitola 9
Závěr

Tato práce ověřila, že je možné program OpenChrom rozšířit tak, aby umožňoval sběr dat z chromatografických zařízení. Aby bylo možné program rozšířit, byly vytvořeny dva projekty. Oba dva projekty jsou umístěné ve vzdáleném repositáři[1]) [2]) a mají otevřený zdrojový kód.

Prvním projektem je knihovna ulan-java, která umožňuje komunikovat se zařízeními připojených prostřednictvím síťového protokolu uLan. Tento ovladač podporuje servisní i procesní typy zpráv. Komunikace se zařízeními probíhá asynchronně. Tento projekt je dostupný pod licencemi MPL[3]), GNU GPL[4]) a EPL v1.0 [5]).

Druhým projektem je rozšíření do programu Openchrom. Toto rozšíření umožňuje sběr chromatografických dat z chromatografů, které obsahují detektor s analogovým výstupem, ke kterému lze připojit analogově digitální převodník ULAD 31 a ULAD 32. Tento převodník lze pak připojit k počítači s využitím rozšíření do programu Openchrom provádět sběr dat. Vytvořené rozšíření přidává do programu OpenChrom perspektivu pro nastavení a provádění sběru dat. Tato perspektiva se nazývá CHROMuLAN. V této perspektivě je umožněno definovat akvizice, které se postupně vykonávají. Každá akvizice má své parametry, které je možné definovat např. jméno, seznam zařízení, která se podílejí na akvizici, doba trvání akvizice. Akvizice lze spouštět ručně nebo je možné spustit akvizici signálem vyslaným z analogově digitálního převodníku. Ukončení sběru dat lze provést automaticky nebo ručně. Zaznamenaná data je možné uložit do jakéhokoliv souboru, který podporuje program Openchrom. Tento projekt je dostupný pod licencí EPL v1.0.

[1]) `http://sourceforge.net/p/ulan/ulan-java/ci/master/tree/`
[2]) https://github.com/holyjan3/ulan-openchrom
[3]) Mozilla Public License `https://www.mozilla.org/en-US/MPL/`
[4]) GNU General Public License `http://www.gnu.org/licenses/gpl.html`
[5]) Eclipse Public License `https://www.eclipse.org/legal/epl-v10.html`

Literatura

[1] Pavel Píša. *MATEMATICKÉ A ELEKTRONICKÉ ZPRACOVÁNÍ SIGNÁLU KAPALINOVÉHO CHROMATOGRAFU*. 2010.
https://support.dce.felk.cvut.cz/mediawiki/images/a/a3/Diz_2010_pisa_pavel.pdf.

[2] *Chromatografie – učební text - 3.LF UK 2015*.
old.lf3.cuni.cz/chemie/cesky/materialy_B/chromatografie.doc.

[3] Wikipedia. *Chromatography software*.
https://en.wikipedia.org/wiki/Chromatography_software.

[4] Pikron. *ULAD 31 - User Guide*. 2013.
http://www.pikron.com/manuals/ulad_31-en.pdf.

[5] Pikron. *ULAD 32 - User Guide*. 2014.
http://www.pikron.com/manuals/ulad_32-en.pdf.

[6] *USB/uLan Analog to Digital Converter - ULAD 32*.
http://www.pikron.com/pages/products/hplc/ulad_32.html.

[7] *USB/uLan Analog to Digital Converter - ULAD 31*.
http://www.pikron.com/pages/products/hplc/ulad_31.html.

[8] *Liquid chromatograph LC5000*.
http://www.pikron.com/pages/products/hplc/lc_5000.html.

[9] Pavel Píša. *uLan RS-485 Communication Driver*.
http://ulan.sourceforge.net/index.php?page=3.

[10] Jindřich Jindřich. *CHROMuLAN*.
http://www.chromulan.org/.

[11] Philip Wenig. *OpenChrom: a cross-platform open source software for the mass spectrometric analysis of chromatographic data*.
http://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-11-405.

[12] James Gosling, Bill Joy, Guy Steele, Gilad Bracha a Alex Buckley. *The Java Language Specification*. 2015.
https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf.

[13] *Is the JVM (Java Virtual Machine) platform dependent or platform independent? What is the advantage of using the JVM, and having Java be a translated language?*
http://www.programmerinterview.com/index.php/java-questions/jvm-platform-dependent/.

[14] Terran Lane. *I Dream of JNI*.
http://www.cs.unm.edu/~terran/downloads/classes/cs351-s05/lectures/l24_may02/l24_may02.pdf.

[15] *OSGi Architecture*.
https://www.osgi.org/developer/architecture/.

[16] Lars Vogel. *OSGi Modularity - Tutorial*. 2015.
http://www.vogella.com/tutorials/OSGi/article.html.

[17] *Manifest*.
http://wiki.osgi.org/wiki/Manifest.

[18] *Plug-ins and bundles*.
http://help.eclipse.org/mars/index.jsp?topic=%2Forg.eclipse.platform.doc.isv%2Fguide%2Fruntime_model_bundles.htm.

[19] Lars Vogel. *Eclipse RCP (Rich Client Platform) - Tutorial*. 2015.
http://www.vogella.com/tutorials/EclipseRCP/article.html.

[20] Lars Vogel. *Using dependency injection in Java - Introduction - Tutorial*. 2012.
http://www.vogella.com/tutorials/DependencyInjection/article.html.

[21] Lars Vogel. *Eclipse 4 event system (EventAdmin) - Tutorial*. 2015.
http://www.vogella.com/tutorials/Eclipse4EventSystem/article.html.

[22] Lars Vogel. *Eclipse application model modularity with fragments and processors - Tutorial*. 2013.
http://www.vogella.com/tutorials/Eclipse4Modularity/article.html.

[23] Lars Vogel. *Eclipse Extension Points and Extensions - Tutorial*. 2013.
http://www.vogella.com/tutorials/EclipseExtensionPoint/article.html.

[24] Lars Vogel. *SWT - Tutorial*. 2014.
http://www.vogella.com/tutorials/SWT/article.html.

[25] Lars Vogel. *Eclipse JFace Overview - Tutorial*. 2014.
http://www.vogella.com/tutorials/EclipseJFace/article.html.

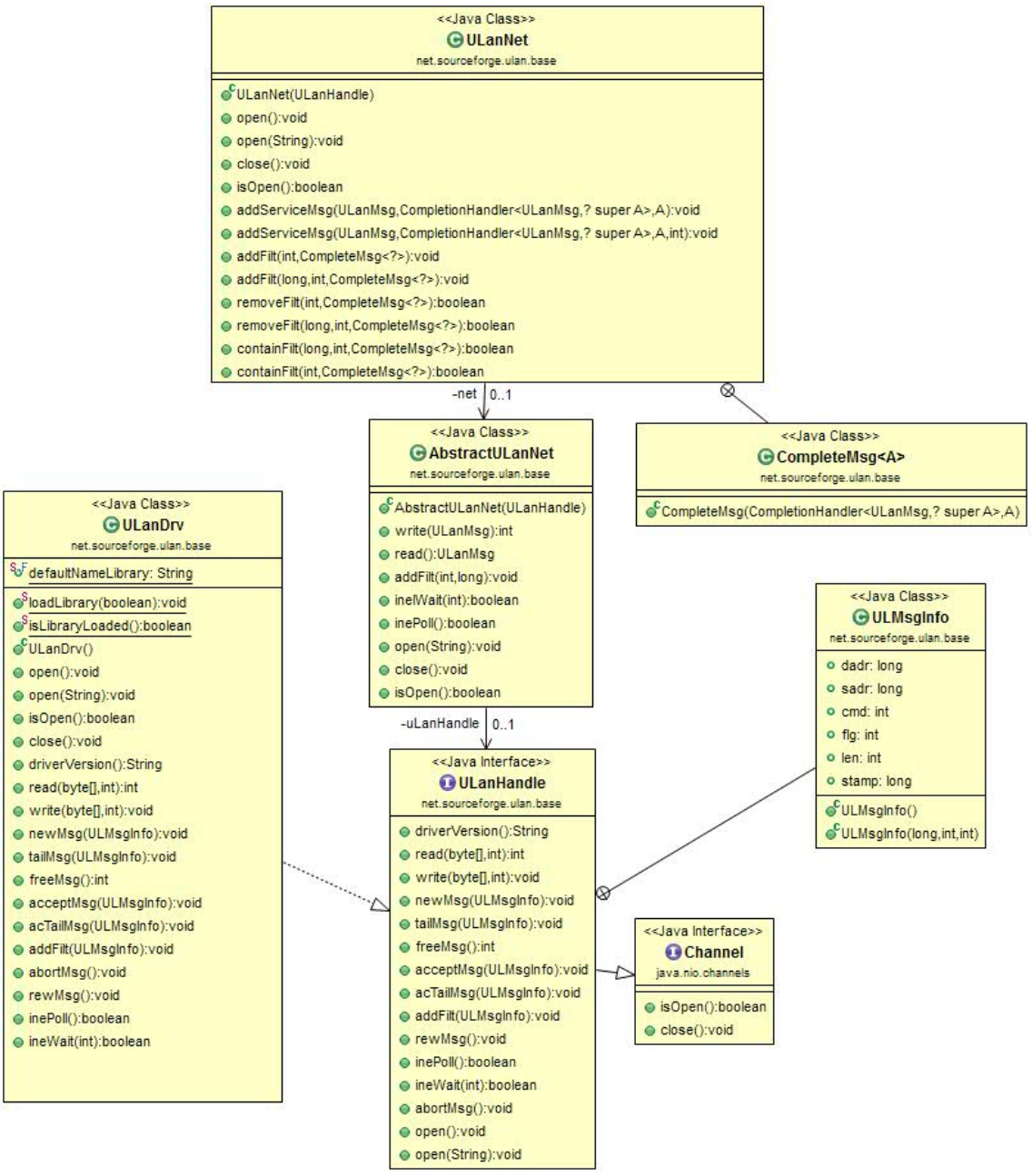
[26] Lars Vogel. *JFace Data Binding - Tutorial*. 2015.
http://www.vogella.com/tutorials/EclipseDataBinding/article.html.

Příloha A
CD

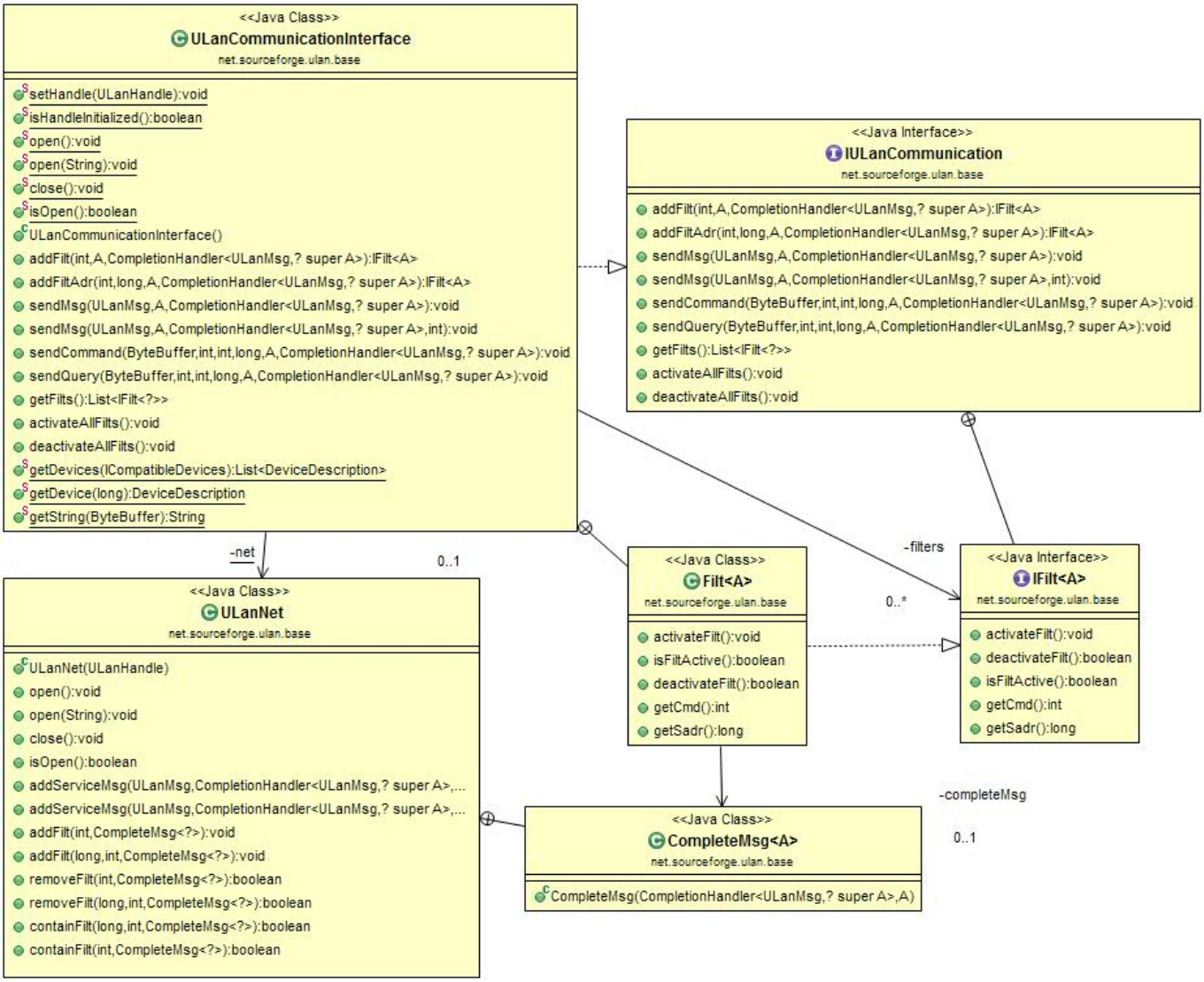
Přiložené CD obsahuje následující položky

- bakalářskou práci v souboru *bp_2016_holy_jan.pdf*
- zdrojový kód ke knihovně ulan-java, který se nachází ve složce *ulan-java*
- zdrojový kód k rozšíření do projektu Openchrom, které je umístěno ve složce *ulan-openchrom*

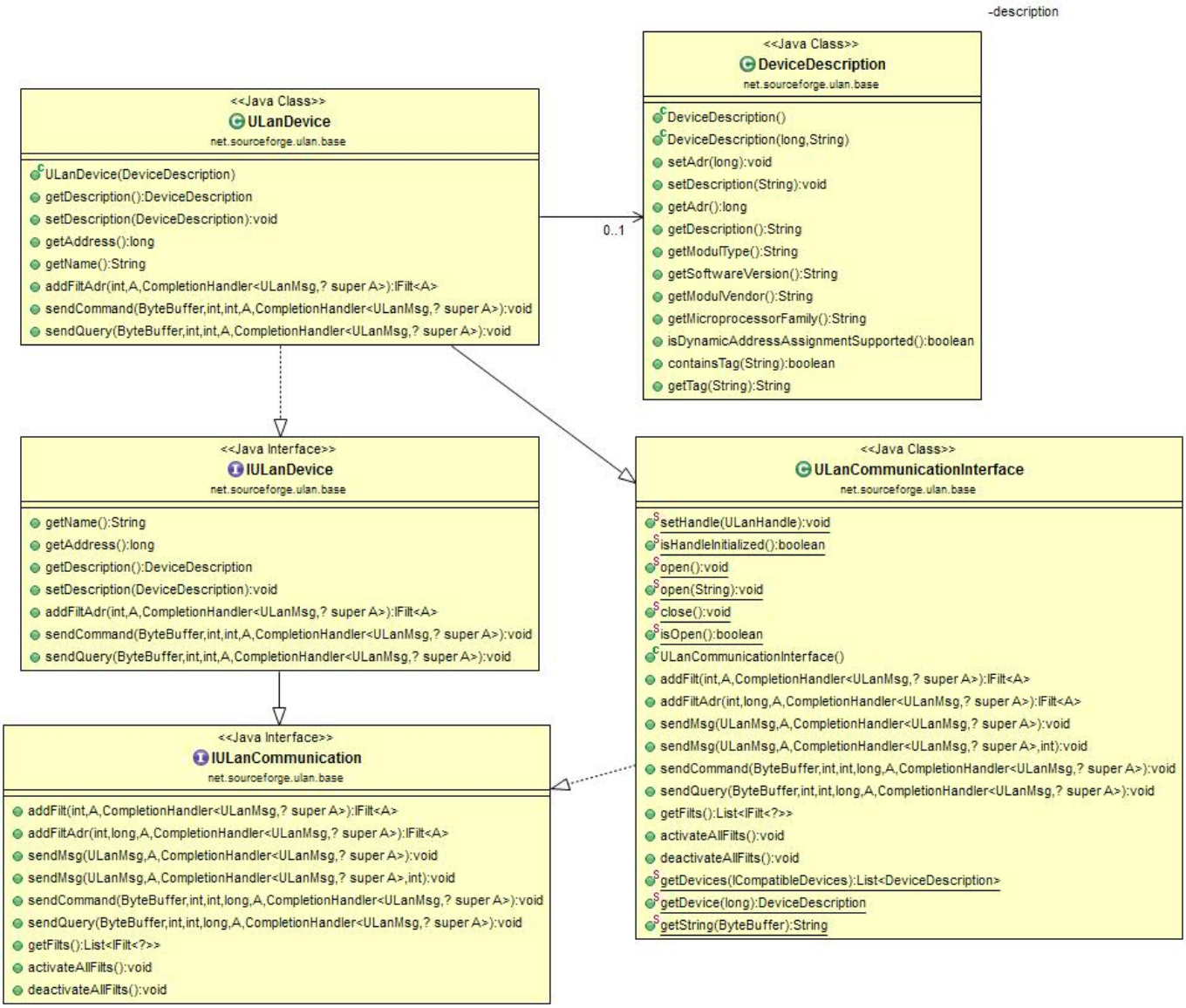
Příloha B
UML diagramy tříd knihovny ulan-java



Obrázek B.1. UML diagram třídy ULanNet



Obrázek B.2. UML diagram třídy ULanCommucationInterface



Obrázek B.3. UML diagram třídy ULanDevice