České vysoké učení technické v Praze
Fakulta biomedicínského inženýrství
Katedra biomedicínské informatiky

Framework pro kompilaci, synchronizaci a analýzu naměřených dat

Framework for compilation, synchronization and analysis of measurement data

Diplomová práce

Autor diplomové práce: Bc. Karavaev Aleksei
Vedoucí diplomové práce: Ing. Jan Hejda, Ph.D.

květen 2021

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení:	**Karavaev**	Jméno:	**Aleksei**	Osobní číslo:	**465473**
Fakulta:	**Fakulta biomedicínského inženýrství**				
Garantující katedra:	**Katedra biomedicínské informatiky**				
Studijní program:	**Biomedicínská a klinická informatika**				

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Framework pro kompilaci, synchronizaci a analýzu naměřených dat

Název diplomové práce anglicky:

Framework for compilation, synchronization and analysis of measured data

Pokyny pro vypracování:

Navrhněte univerzální framework pro import, synchronizaci, analýzu a export dat z různých systémů, typicky záznamů ze simulátorů dopravních prostředků a zařízení pro záznam fyziologických dat. Framework musí být modulární a konfigurovatelný tak, aby umožnil integraci nových formátů pro import a modulů pro analýzu dat. Implementujte skript pro konfiguraci frameworku pro dávkové zpracování naměřených experimentálních dat a vizualizaci výsledků.

Seznam doporučené literatury:

[1] Ryant, I., Algoritmy a datové struktury objektově, ed. Vydání první, V Praze : Ivan Ryant,, 2017, ISBN 978-80-270-1660-0

[2] Pianykh, O. S., Digital imaging and communications in medicine (DICOM) : a practical introduction and survival guide, ed. 2nd ed., Berlin; Heidelberg : Springer, 2012, ISBN 978-3-642-10849-5

Jméno a příjmení vedoucí(ho) diplomové práce:

Ing. Jan Hejda, Ph.D.

Jméno a příjmení konzultanta(ky) diplomové práce:

doc. Ing. Patrik Kutílek, MSc., Ph.D., Ing. Petr Volf

Datum zadání diplomové práce: **15.02.2021**
Platnost zadání diplomové práce: **18.09.2022**

doc. Ing. Zoltán Szabó Ph.D.	prof. MUDr. Jozef Rosina, Ph.D., MBA
podpis vedoucí(ho) katedry	podpis děkana(ky)

Zadání práce

Navrhněte univerzální framework pro import, synchronizaci, analýzu a export dat z různých systémů, typicky záznamů ze simulátorů dopravních prostředků a zařízení pro záznam fyziologických dat. Framework musí být modulární a konfigurovatelný tak, aby umožnil integraci nových formátů pro import a modulů pro analýzu dat. Implementujte skript pro konfiguraci frameworku pro dávkové zpracování naměřených experimentálních dat a vizualizaci výsledků.

Prohlášení

Prohlašuji, že jsem diplomovou práci s názvem **Framework pro kompilaci, synchronizaci a analýzu naměřených dat** vypracoval/a samostatně a použil/a k tomu úplný výčet citací použitých pramenů, které uvádím v seznamu přiloženém k bakalářské/diplomové práci.

Nemám závažný důvod proti užití tohoto školního díla ve smyslu §60 Zákona č.121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů.

V . . . . . dne . . . .

Karavaev Aleksei

Poděkování

Rád bych poděkoval mému vedoucímu za podporu a vedení práce.

Abstrakt

V rámci této práce byl řešen návrh modulárního a konfigurovatelného frameworku pro předzpracování dat umožňující mimo jiné import, synchronizaci, analýzu a export. Na základě zadání práce byla vytvořena architektura, podle které byl program implementován v programovacím jazyce Python ve verzi 3.6. Funkčnost frameworku byla následně ověřena na anonymizovaných datech ze simulátoru dopravních prostředků a zařízení pro snímání fyziologických dat. Mezi klíčové charakteristiky vytvořeného nastroje patří rozšiřitelnost a modifikovatelnost, jednoduchá konfigurace pomocí souboru ve formátu json, možnost nastavení odstranění různých druhů artefaktů v datech a následná vizualizace.

Klíčová slova

Analýza dat, předzpracování dat, orientovaný graf, simulátor dopravních prostředků.

Abstrakt v angličtině

In this work we designed a modular and configurable framework for data preprocessing, that allows data import, synchronization, analysis and export. Software architecture was created based on work assignment. Python 3.6 language was then used for implementation. The framework was tested on anonymized data generated by car driving simulator and devices for measurement biological data. Key characteristics of the framework are modularity, easy configuration using json file, ability to set mistake correction methods for data and visualize data.

Klíčová slova v angličtině

Data analysis, data preprocessing, oriented graph, car driving simulator

Obsah

1 Úvod

Ve vědecké praxi často nastává situace, kdy se musejí analyzovat data, která se měřila na různých, často experimentálních zařízeních s různými vzorkovacími frekvencemi. Tato data pak mají různou strukturu a formáty, navíc se v nich můžou vyskytovat chyby.

Aby se podobná data mohla dál zpracovávat, nejprve se musejí odstranit chyby. V některých případech je potřeba zajistit (např. v případě analýzy pomocí rekurentních neuronových sítí [1]), aby signály měly pro každou časovou značku definované hodnoty [2].

Navržený framework by měl poskytovat nástroje pro předzpracování dat, konkrétně pro odstranění typicky se vyskytujících chyb, a pro synchronizaci dat. Čímž by se ulehčila a zkvalitnila další analýza.

Záměrem této práce je vytvořit framework pro předzpracování signálů, který by umožňoval jednoduchou konfiguraci, poskytoval veškeré potřebné funkcionality, a byl rozšiřitelný a modulární.

Jako postup návrhu se používá vodopádový model, který je jedním z nejrozšířenějších přístupů k vývoji softwaru [3].

Testování navrženého nástroje probíhá na anonymizovaných datech ze simulátoru dopravních prostředků a zařízení pro záznam fyziologických dat, která vznikla v rámci experimentu pro hodnocení stresu řidičů.

2 Přehled současného stavu

V rámci práci byla realizována rešerše dostupného softwaru pro předzpracování a analýzu dat. Přehled těchto nástrojů je v tabulce 2.1.

Jedním z nástrojů pro čistění dat a konkrétně pro odstraňování duplikátních záznamů je IntelliClean. Funguje na principu hledání a mazání podobných záznamů v databázích. V případě práce s daty, která mají stejný význam ale rozdílnou reprezentaci, umožňuje přemapování. Též podporuje vyplnění chybějících hodnot[4].

Dalším programem pro manuální předzpracování dat je Potters Wheel[5][6]. Tento software umožňuje přidávání, mazání a kopírování sloupců, přemapování řádků, použití funkce na každou hodnotu sloupce, rozdělení sloupce do několika a spojování sloupců. Uživatel pracuje s daty, která jsou reprezentována pomocí tabulek. Tento SW má i grafické uživatelské rozhraní. Umožňuje čtení ze souboru nebo připojení na databázi pomocí ODBC rozhraní[5].

Problém předzpracování se často řeší v oblastí velkých dat, kde data přichází z různých zdrojů, jsou často zašuměná a navíc se v nich vyskytují chyby[6].

Mezi existující nástroje, které řeší problém čistění velkých dat, patří například Cleanix. Některé jeho funkce jsou například detekce abnormálních dat, vyplnění chybějících hodnot, identifikace a korekce neúplně zaznamenaných nebo duplikovaných hodnot. Uživatel musí určit pravidla, podle kterých probíhá oprava chyb. Umožňuje i paralelní zpracování dat[7].

Software BigDansing se též zabývá předzpracováním dat, taky poskytuje možnost nastavit pravidla opravy chyb ve velkých datech. BigDansing může běžet jako nadstavba nad vetšinou populárních systémů pro zpracování dat typu MapReduce nebo systémů řízení databází[8].

Některé úlohy předzpracování velkých dat se mohou řešit pomocí programového nástroje KATARA. Jeho hlavní funkcionalitou je též čistění dat pomocí opravy nebo odstraňování chybných hodnot, využívá k tomu veřejně přístupných bází znalostí(angl. knowlege base) typu DBPedia, Yago nebo Freebase[9]. Nepodporuje ale paralelní zpracování dat[6]. KATARA načte tabulky, zvaliduje je pomocí bází znalostí a uloží.

Framework SCARE pro odstranění chyb v velkých datech používá bayesovský klasifikátor pro predikci chybějících nebo chybných hodnot. Expert následně musí schválit úpravy[10].

Stejnou úlohu řeší nástroj ERACER, který na rozdíl od SCARE využívá bayesovských sítí[11].

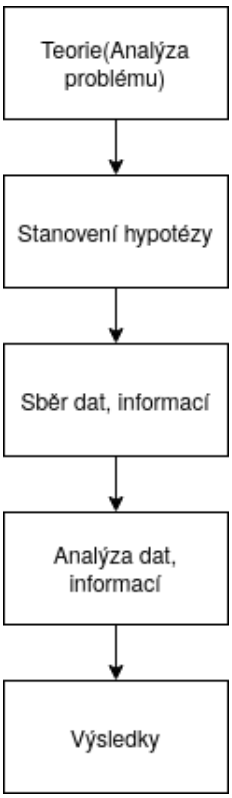
Tabulka 2.1: Porovnání nástrojů pro předzpracování dat

	Velká data	Princip	Paralelní fungování	Zdroj dat
IntelliClean	Ne	Nastavení pravidel, podle kterých se chyby opraví	Ne	Databáze
Potters Wheel	Ne	Nastavení pravidel, podle kterých se chyby opraví	Ne	Databáze, soubor
Cleanix	Ano	Nastavení pravidel, podle kterých se chyby opraví	Ano	Databáze
BigDansing	Ano	Nastavení pravidel, podle kterých se chyby opraví	Ano	Databáze
Katara	Ano	Použití veřejně dostupných zdrojů pro hledání chyb	Ne	Databáze
Scare	Ano	Použití naivního bayesovského klasifikátoru pro hledání a odstraňování chyb	Ano	Databáze
Eracer	Ano	Použití naivního bayesovských sití pro hledání a odstraňování chyb	Ano	Databáze

2.1 Analýza dat ve výzkumné praxi

Zpracování a analýza dat hrají velkou roli ve vědecké praxi a konkrétně v empiricky zaměřeném výzkumu, mezi jehož základní metody patří pozorování, měření a experiment[12].

Kvantitativní výzkum předpokládá, že v rámci experimentu zaměřeného často na ověření hypotézy vznikají seznamy pozorování, které se pak musejí zanalyzovat a interpretovat s využitím statistiky [12]. Tato pozorování obvyklé numerickou podobu. Průběh kvantitativního výzkumu je znázorněn na obrázku 2.1.



Obrázek 2.1: Průběh kvantitativního výzkumu[12]

V současné době se datová analýza často provádí pomocí programovacích jazyků Python nebo R[13].

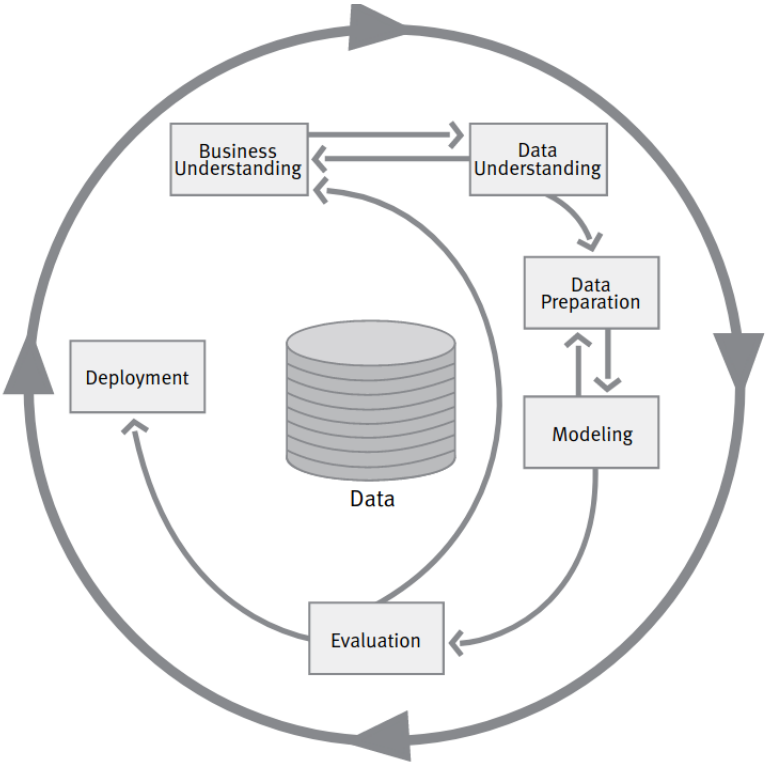
2.2 CRISP DM

Procesu měření dat a získání z nich užitečných informací se říká vytěžování dat (angl. data mining). Dnes de facto standardem procesu vytěžení dat je proces reprezentovaný

modelem CRISP-DM[14]. Tento model popisuje postup nezávislý na konkrétní oblasti zkoumání dat a umožňuje jeho standardizaci. Podle tohoto modelu mezi základní kroky projektů, které jsou spojené s analýzou dat, patří:

- Porozumění problematice (angl. Business understanding)
- Porozumění datům (angl. Data understanding)
- Příprava dat (angl. Data preparation)
- Modelování (angl. Modeling)
- Vyhodnocení výsledků (angl. Evaluation)
- Využití výsledků (angl. Deployment)

Diagram modelu CRISP-DM je znázorněn na obrázku 2.2.



Obrázek 2.2: Model CRISP-DM převzato z [15]

Tento postup ale není rigidní, umožňuje vynechání kroků, pokud nejsou potřebné pro řešení úlohy, anebo naopak zopakování některých etap. Každý z kroků se může následovně rozdělit do pokroků.

2.2.1 Porozumění problematice

V této fázi se stanovují cíle vytěžování dat. Jakých výsledků je potřeba dosáhnout, a podle jakých kriterií se hodnotí, jestli se to podařilo nebo ne. Zjišťuje se, jestli již existují řešení podobných problémů[15]. Na základě těchto informací následně vzniká konkrétní definice úlohy vytěžování dat.

2.2.2 Porozumění datům

Cílem této etapy je porozumět, s jakými daty bude potřeba pracovat, jaké problémy mohou vzniknout, a jestli kvalita dat bude dostačující pro dosazení požadovaného výstupu[15]. Prvním krokem této etapy je sběr dat, poté nasleduje jejich průzkum a hodnocení kvality. V rámci průzkumu dat se data zobrazují, zjišťuje se, jaká data jsou nejzajímavější z hlediska definovaných cílů, vypočítávají se základní statistiky (např. minimum, maximum, rozptyl). Také se určuje, jaké vady mohou data obsahovat, a jestli to neovlivní realizaci projektu. Příklady vad jsou: chybějící hodnoty, různá označení stejných kategorií (M a Muž v různých dokumentech).

2.2.3 Příprava dat

Přípravou dat se myslí taková jejich úprava, aby byla vhodná pro následující analýzu. Často probíhá v několika iteracích. Mezi jednotlivé kroky patří například: volba vhodných příznaků, čistění, generování nových příznaků, integrace dat, formátování [15].

Během volby (selekce) se musí zvážit, jaká data jsou vhodná pro řešení problému, například jestli zda je nutné používat data, která navzájem korelují.

Čištění dat zahrnuje jak vyplnění chybějících hodnot a opravu chyb, tak i zahození nevhodných záznamů.

Proces dopočítání nových příznaků může obsahovat:

- Agregace nových dat, například výpočet rozptylu nebo střední hodnoty.
- Generování nových řádků, například pomocí interpolace a převzorkování.
- Škálování. Některé algoritmy a modely vyžadují data v rozmezí 0 až 1.

Integrací dat se myslí spojení několika záznamů do jednoho.

Formátování dat je nezbytným krokem, pokud nástroje, které se použijí na modelování, vyžadují určitý formát dat.

2.2.4 Modelování

Během této fáze probíhá aplikování matematických modelů a výzkumných metod na předpřipravená data. Často modely vyžadují, aby data byla v určitém formátu, v těchto případech je možný návrat do předchozí fáze[15]. Tento krok probíhá iteračně, vybírají se modely, aplikují se a volí se nejlepší varianta podle zvolených kritérií. Výstupem tohoto kroku je funkční model a nové informace ohledně spojitostí v datech.

2.2.5 Vyhodnocení výsledků

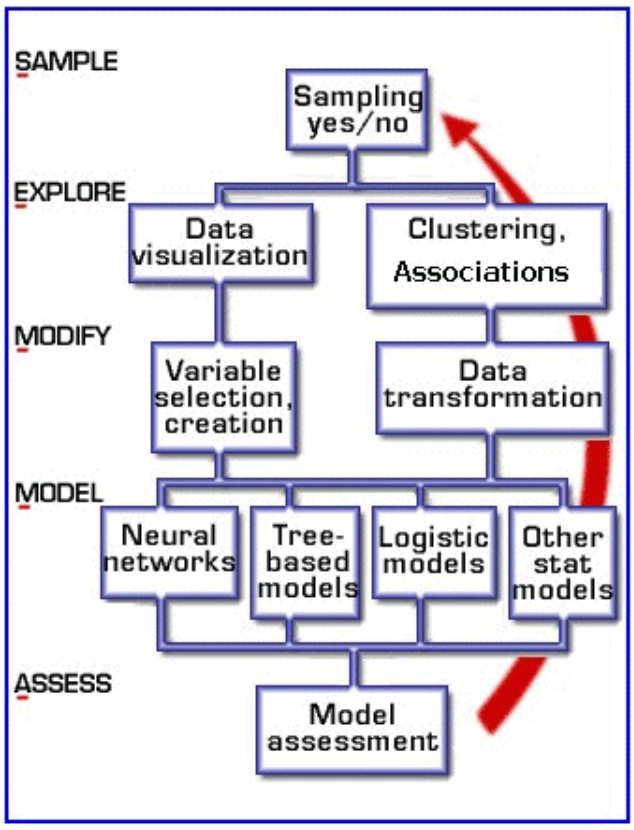
V této etapě už existují modely, které vznikly jako výstup datové analýzy. Měly by mít už i dostačující kvalitu. Než se přejde do finální fáze, je potřeba ještě jednou zhodnotit veškeré kroky, které vedly k vzniku modelu. Na konci tohoto kroku už by měl existovat model, který splňuje všechny stanovené požadavky.

2.2.6 Využití výsledků

Často projekt nekončí vytvořením modelu. Je potřeba finalizovat dokumentaci a popis vytvořeného systému, rozvrhnout plán nasazení systému a jeho údržbu. Následně podle plánu proběhne nasazení. Údržba by měla probíhat pravidelně.

2.3 SEMMA

Dalším populárním postupem pro datovou analýzu je model SEMMA(sample, explore, modify, model, access)[16], který je úzce zaměřen na analýzu dat a na rozdíl od CRISP-DM vynechává krok 'porozumění problematice'. Popis modelu je na obrázku 2.3.



Obrázek 2.3: Model SEMMA převzato z [17]

2.3.1 Volba dat

Volba dat je prvním krokem modelu SEMMA, cílem je vybrat z celého množství dat reprezentativní množinu, která by obsahovala informace dostatečné pro analýzu a zároveň byla dostatečně malá, aby s ní šlo efektivně pracovat. Data se mohou rozdělit do několika skupin: trénovací, validační a testovací[17]. Na trénovací data se bude aplikovat model, na testovacích se zhodnotí jeho funkčnost.

2.3.2 Průzkum

Tento krok zahrnuje iterativní hledání trendů a anomálií v datech, vizualizaci a statistickou analýzu. Odpovídá etapě 'Porozumění datům' v modelu CRISP-DM.

2.3.3 Modifikace

Během modifikace se mohou volit vhodné příznaky (selekce příznaků) anebo vznikat nové (extrakce příznaků). Předzpracování dat, které zahrnuje odstranění chyb a další úpravy, je též součástí tohoto kroku.

2.3.4 Modelování

Účel a průběh této fáze je stejný jako u CRISP-DM modelu. Jelikož těžba dat je dynamický a iterativní proces, je možné se, stejně jako u CRISP-DM modelu, vrátit k předchozí etapě.

2.3.5 Vyhodnocení

Cílem vyhodnocení je určit, jestli navržený model splňuje veškeré stanovené požadavky. Vyhodnocení se často dělá tím způsobem, že se model aplikuje na testovací data, která se vybrala ve fázi 'volba dat'.

2.4 Předzpracování dat

Součástí každého ze zmíněných modelů je předzpracování dat a jejich příprava na následující analýzu. Vstupem jsou nezpracovaná data, výstupem jsou vyčištěná, modifikovaná data ve vhodném formátu.

2.4.1 Čištění dat

Data, která se naměří za různých podmínek a na různých systémech, často obsahují chyby, jež ovlivňují kvalitu a přesnost vygenerovaných modelů [6]. Mezi kroky čistění dat typicky patří identifikace, detekce a odstranění chybových hodnot. Příklady chyb[6]:

- Chybějící hodnoty.
- Duplikovaná data.
- Neúplně zapsané hodnoty.
- Data, která mají různý způsob zápisu ale stejný význam.

Chybějící hodnoty se musejí nahradit a to bud za jednu hodnotu anebo pomocí pokročilejších metod typu interpolace a převzorkování.

Duplikovaná data se odstraňují, pokud se jisté, že se skutečně duplikují.

Neúplně zapsané hodnoty se musejí odstranit anebo opravit, přičemž oprava často není možná.

Data, která mají různý způsob zápisu ale stejný význam, se převedou do jednotné podoby. Příkladem je přemapování hodnoty 'Muž' na 'M' anebo naopak.

2.4.2 Integrace

V některých případech data, která se vztahují k jednomu měření, se zapisují do různých úložišť. Práce s takto uloženými daty není optimální, proto se data muset přepsat do jednoho úložiště. Příkladem takové situace je měření s několika zařízeními, kde každé z nich ukládá data do svého souboru.

2.4.3 Formátování

Tento krok probíhá v tom případě, když následující analýza vyžaduje určitý formát dat. Během formátování se nemění smysl dat, veškeré změny, které se provedou, mají pouze vnější charakter.

2.4.4 Selekce příznaků

Hlavním účelem selekce příznaků je zmenšení komplexity dat pomocí vynechání irelevantních a redundantních dat, což vede k zmenšení výpočetní složitosti, lepšímu porozumění datům a zvýšení kvality výstupních modelů [18][19]. V případě řešení úlohy strojového učení, irelevantními příznaky jsou takové příznaky, které neovlivňují výstup modelu[20].

2.4.5 Extrakce příznaků

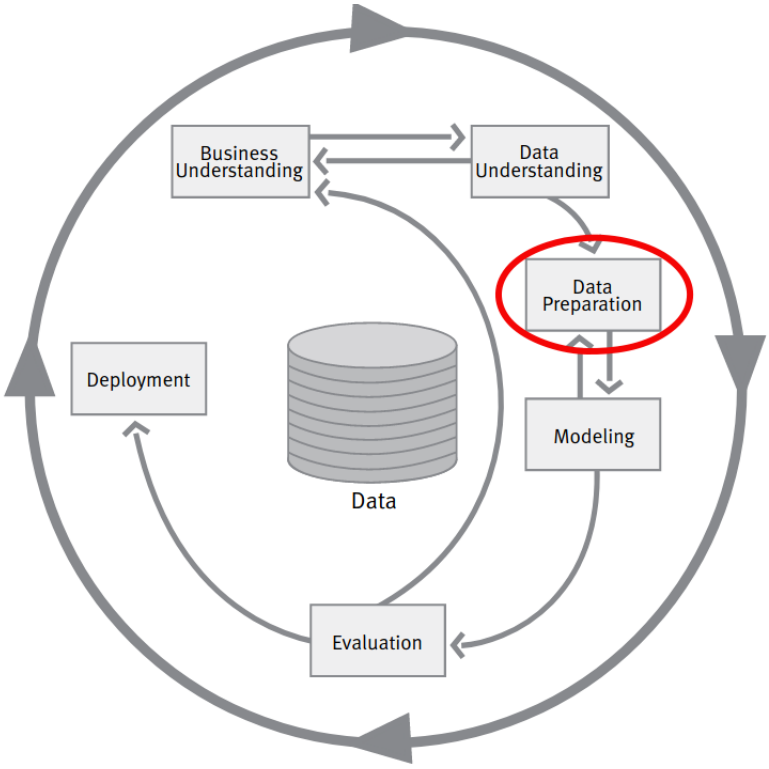
Extrakce příznaků také řeší problém zvětšení kvality výstupního modelu a zmenšení rozměrnosti vstupních dat [21]. Na rozdíl od selekce, během tohoto procesu vznikají nová data. Příkladem metody pro zmenšení dimensionality a generování nových příznaků je

analýza hlavních komponent [21]. Extrakce příznaků se také může provádět pomocí Fourierovy a Vlnkové transformace[22].

2.5 Cíle práce

Cílem práce je navrhnout robustní, modulární a jednoduše konfigurovatelný program, který by umožňoval automatizované předzpracování dat, konkrétně načítání dat v různých formátech, jejich čištění, integraci, formátování a následný export. Program by měl umožňovat rozšíření pomocí integrace nových modulů. Nové moduly by mohly plnit např. vizualizační anebo analytické funkce. Poté se navržený program musí otestovat na datech ze simulátoru dopravních prostředků.

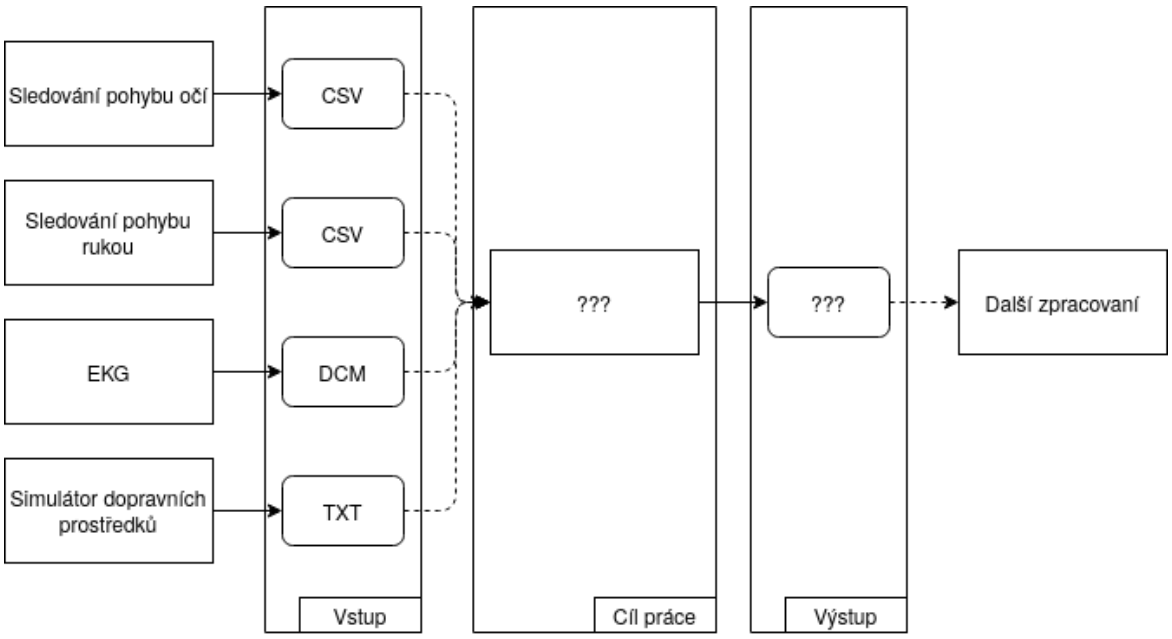
Pokud by se použilo označení CRISP-DM modelu, cílem práce je návrh a implementace modulu pro předzpracování dat.



Obrázek 2.4: Cíl práce v terminologii CRISP-DM [15]

Cíl práce je označen červenou elipsou na obrázku 2.4.

Testování navrženého programu by proběhlo podle diagramu, který je znázorněn na obrázku 2.5.



Obrázek 2.5: Diagram testování výsledného programu

2.6 Postup návrhu

Jako referenční model postupu se použil vodopádový model, který popisuje jeden z nejpoužívanějších vývojových procesu. Fáze vývojového procesu následují za sebou, nová fáze začíná, když se dokončí předchozí [3].

Postup, který vychází z tohoto modelu, lze popsat následovně:

- Provést analýzu požadavků a specifikovat požadavky na program.
- Navrhnout architekturu SW, z jakých modulů se bude skládat, jak moduly budou interagovat. A to tak, aby se splňovaly vydefinované požadavky.
- Implementovat program podle architektury.
- Otestovat navržený software na datech ze simulátoru dopravních prostředků.

3 Metody

3.1 Analýza požadavků

Výstupní program by měl načítat data z různých systémů, transformovat je a ukládat upravenou verzi ve zvoleném formátu. Příkladem použití takového nástroje je předzpracování dat ze simulátoru dopravních prostředků a zařízení pro záznam fyziologických dat.

3.1.1 Formát vstupních dat

Data jsou poskytována ve formě .csv, .dcm anebo .txt souborů. Vstupní soubor vždy obsahuje alespoň jeden signál. Každý vstupní signál je definován pomocí dvou 1D číselných vektorů. První vektor je časový a obsahuje časová razítka ve formátu UNIX. Druhý vektor je datovým vektorem. Data mohou mít libovolnou formu, to znamená, že mohou být reprezentovaná textem. Textová data jsou událostmi, které nemají danou vzorkovací frekvenci. Naopak některé signály, jako např. výstup z EKG, mají jasně definovanou vzorkovací frekvenci, přičemž vzorkovací frekvence signálů z různých zdrojů se mohou lišit.

Navrhovaný program by měl umožňovat načítaní dat ve zmíněných formátech a podporovat interpretaci těchto dat jako časových řad.

3.1.2 Vyskytující se chyby

Vzhledem k vnitřní struktuře dat simulátoru dopravních prostředků, na začátku zápisu několik řádků mohou mít stejnou časovou značku. Po několika sekundách záznamu se tato chyba přestává vyskytovat.

Podobně funguje i zařízení pro snímaní pohybu rukou, některé řádky mají stejný čas. Jediným rozdílem je to, že se tato chyba vyskytuje pravidelně během měření.

Zařízení pro snímaní pohybu rukou a očí generují prázdné záznamy, avšak data se neztrácejí.

Používané EKG zařízení nepodporuje záznam delší než 30 minut proto se několik záznamů mohou vztahovat k jednomu měření v případě, když čas měření přesáhne tuto hodnotu.

U hodnot s plovoucí čárkou se u některých systémů čárka zapisuje jako čárka, u některých jako tečka, což by mohlo bránit správné interpretaci hodnot.

Názvy souborů by měly obsahovat datum podle kterého se určuje jaká data patří jakému záznamu, nicméně princip kódovaní se liší podle měření, což brání zpracování. Příkladem je EKG záznam, u kterého název může ale nemusí zahrnovat doplňující informaci zapsanou na začátku nazvu souboru.

Další zajímavou vlastností je to, že čas některých záznamů je posunut o dvě hodiny, což by se mělo během zpracování vykompenzovat.

U senzorů biologických signálů se občas objevují chybějící hodnoty, které se vyplňují pomocí různých metod, mezi které patří například lineární interpolace, interpolace metodou nejbližšího sousedu nebo metody strojového učení[23] [2].

Navrhovaný framework by měl umožňovat opravu zmíněných typů chyb, a připadnou integraci dalších modulů v případě, když se ve vstupních datech budou objevovat i jiné druhy chyb.

3.1.3 Další potřebné transformace

Data mohou mít rozdílné vzorkovací frekvence z čehož plyne, že pro určité hodnoty času nemusí existovat odpovídající hodnoty signálů. Pro některé druhy analýzy je potřeba aby hodnoty signálů byly definované pro všechny časy[2]. Tento problém se řeší pomocí interpolaci a převzorkování[2].

Z čeho plyne že dalším požadavkem na systém by byla možnost aplikací interpolačních metod a převzorkování na vstupní data.

3.1.4 Volba výstupního formátu

Formát musí umožňovat rychlé a efektivní vyhledávaní vzorku v čase. Synchronizovaná data ze simulátoru dopravních prostředků se budou dál analyzovat pomocí jiných programů z čehož plyne, že čitelnost pro člověka není prioritou. Důležité je ukládaní metadat jako jsou např. vzorkovací frekvence, začátek a konec měření apod. Malá velikost výstupního souboru je výhodou. Na základě těchto informací byla stanovena následující kritéria:

- Formát by měl byt co nejvíce rozšířeným.

- Formát by měl umožňovat vyhledávaní v čase (přímo ze zadaní).
- Formát může být binární (malá velikost, nemusí být čitelný).
- Formát by měl umožňovat ukládaní metadat.

V současné době neexistuje jednotný ukládací formát, který by byl používán všemi výrobci zařízení. Ukládání do .csv nebo .txt je užitečné, pokud jde o malý objem dat.

Zápis do XML souboru využívá metadat, což vede k vyšší přehlednosti dat, ovšem na úkor velikosti souboru.

Často je pro ukládaní a přenos dat ze zdravotnických prostředků využíván formát DICOM, který podporuje práci s obrázky. [24]

Alternativou je formát HDF5 [25][26]. Umožňuje práci s většími a komplexnějšími daty. Podporované datové typy jsou: integer, floating-point, string, bitfield, opaque, compound, reference, enum, variable-length sequence, array. Struktura protokolu umožňuje využití paralelního zpracovávaní dat[26]. Díky možnosti ukládaní metadat je samo-popisujícím formátem[27].

Přehled formátů je v tabulce3.1.

Tabulka 3.1: Porovnaní formátů

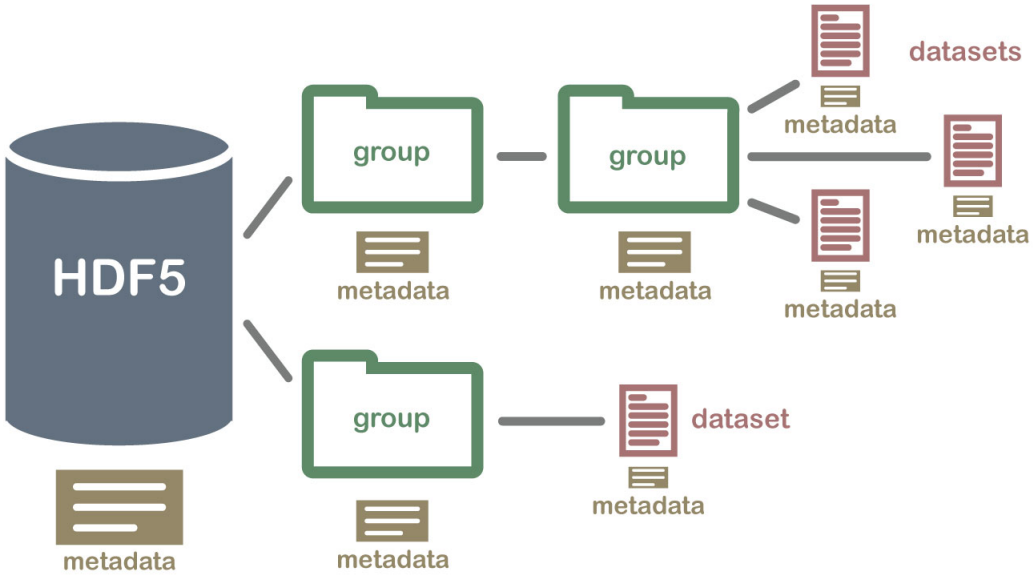
	ASCII	Json	HDF5	DICOM
Binární	Ne	Ne	Ano	Ano
Metadata	Ne	Ano	Ano	Ano
Typizace	Ne	Ano	Ano	Ano
Rychlé vyhledávaní	Ne	Ne	Ano	Ano
Často používán	Ano	Ano	Ano	Ano,ve zdravotnictví

3.1.5 Stručný popis formátu HDF5

HDF5 je z angličtiny "hierarchický datový formát". Základní součásti jsou:

- Skupina
- Dataset
- Metadata

Metadata jsou představené jako atributy skupin a datasetů, jejich počet a datový typ nejsou limitované. Datasety musí mít specifikovaný datový typ, což potenciálně zmenšuje velikost výstupního souboru[25][26]. Model vnitřní struktury tohoto formátu je zobrazen na obrázku 3.1.



Obrázek 3.1: Struktura HDF5 převzato z [25]

3.1.6 Vyhledávaní vzorku v čase

Formát HDF5, umožňuje načítaní vzorku podle indexu, aniž by se celý soubor musel načíst do operační paměti počítače.

Pro vyhledávaní vzorku s určitou časovou značkou stačí načíst vzorkovací frekvenci z metadat a časovou značku prvního vzorku. Index hledaného vzorku se pak stanoví podle vzorce:

$$n = (t_n - t_0)/(1/f) \tag{1}$$

Kde n je index, f je vzorkovací frekvence v hertzech, t_n je čas hledaného vzorku v milisekundách, t_0 je čas nultého vzorku v milisekundách.

3.2 Specifikace požadavku

Jedním ze způsobu dělení požadavků na systém je dělení na funkční požadavky a nefunkční požadavky.

Funkční požadavky vyjadřují klíčové funkcionality systému, co by systém měl umět[28].

Nefunkční požadavky popisují, jaké charakteristiky by měl mít systém jako celek. Nejde je často popsat měřitelným způsobem. Příkladem takových požadavků mohlo být: bezpečnost, spolehlivost nebo použitelnost[29].

3.2.1 Funkční požadavky

Obecný přehled funkcionalit softwaru pro předzpracování dat je popsán v kapitole 2.4. Na základě této kapitoly a analýzy požadavků z kapitoly 3.1 se určily následující funkční požadavky na program:

- Možnost integrace dat z různých systémů, import dat ve formátech .csv, .txt, .dcm a následující interpretace dat jako časových řad.
- Podpora následujících operací.
  - Náhrada chybějících hodnot.
  - Odstraňovaní duplikátních záznamů na základě časové značky.
  - Posun časových značek.
- Export dat ve formátech .csv a .h5.
- Možnost automatického předzpracování dat.
- Možnost vizualizace dat.

3.2.2 Nefunkční požadavky

Nefunkční požadavky na systém plynou ze zadání a cíle práce a lze je vyjádřit takto:

- Modifikovatelnost a rozšiřitelnost.
- Jednoduchá konfigurace.

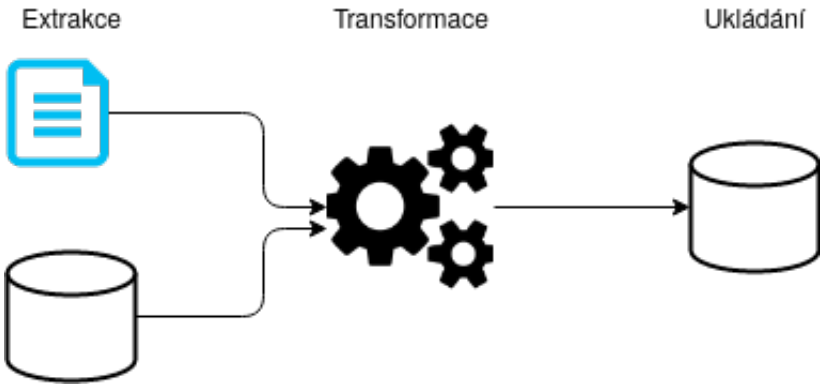
3.3 ETL model

Z kapitoly 2, která popisuje přehled současného stavu a kapitoly 2.4 lze určit typickou architekturu frameworku pro předzpracování dat. Data se načtou ze zdrojů, provedou se potřebné transformace, potom se výstupní data uloží. Tento model se jmenuje ETL model, a často se používá v oblasti velkých dat [30].

Skládá se ze tři základních součástí

- Extrakce dat
- Transformace
- Ukládání

Etl model je znázorněn na obrázku 3.2.

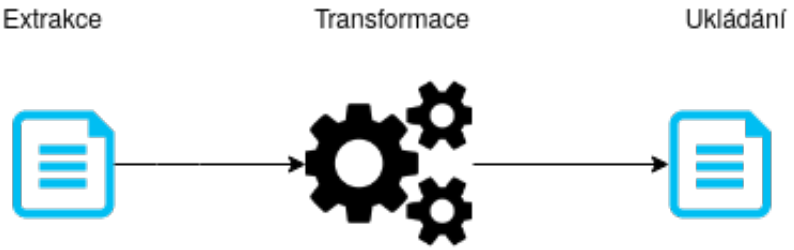


Obrázek 3.2: Etl model

Jako zdroj dat se může používat databáze nebo soubor [30], po transformacích se data nahrají do úložiště.

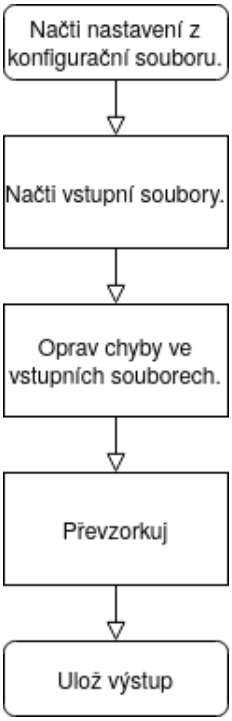
3.4 Model řešení

V případě řešené úlohy se data budou čist ze souborů a ukládat do souboru, z čehož plyne, že tok dat výstupního sytému bude vypadat jako na obrázku 3.3.



Obrázek 3.3: Upřesněná verze etl modelu

Přičemž část pro načítání, by měla umět čist data ze souborů ve formátech, které se vydefinovaly v požadavcích. Následně proběhnou transformace, které opraví chyby a převzorkují data v případě, když je to potřeba. Takto upravená data se uloží do výstupního souboru. Nastavení parametrů každé ze součástí by mohlo probíhat pomocí konfiguračního souboru.



Obrázek 3.4: Obecný algoritmus běhu programu

3.4.1 Extrakce

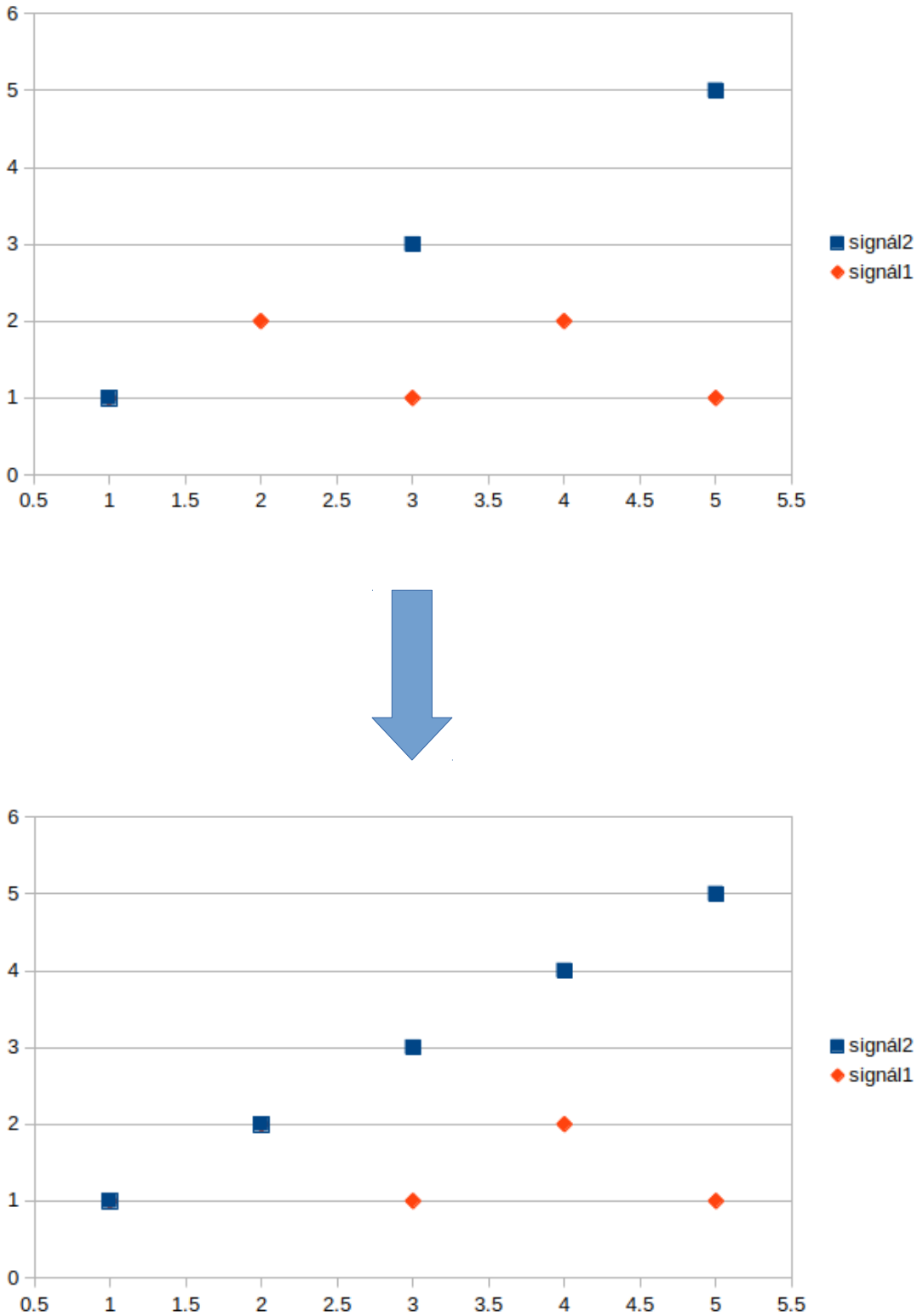
Hlavní funkcionalitou je načtení dat v různých formátech, vstupem jsou cesty k souborům, výstupem jsou načtené signály. V závislosti na formátu se mohou lišit i knihovny pro načítání.

3.4.2 Transformace

Předzpracováním dat je jejich příprava na další analýzu. Vstupem jsou data, výstupem jsou očištěná data.

Vstupní signály nemusí mít ani stejnou pro všechny signály. Mohou mít i měnící se v čase vzorkovací frekvenci[31]. Přičemž v některých případech pro analýzu je potřeba zajistit aby každý z signálu měl definovanou hodnotu pro každou časovou značku[1][2].

Pro tyto účely se používá převzorkování. V případě, když je potřeba dosáhnout vetší frekvenci než frekvence původního signálu, nejprve se musí udělat interpolace. Cílem interpolace a převzorkování je určit chybějící hodnoty tak, aby byly co nejvíce podobné opravdovým hodnotám signálu. Interpolační metoda se volí v závislosti na typu dat, která se analyzují [32]. Příklad dat na vstupu a na výstupu interpolace a převzorkování lze vidět na obrázku 3.5.



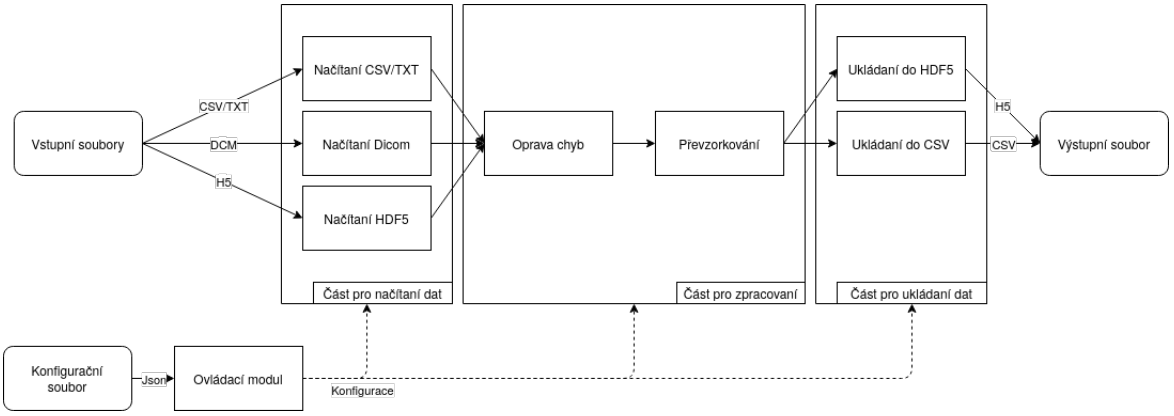
Obrázek 3.5: Příklad vstupu a výstupu po lineární interpolaci a převzorkování

3.4.3 Ukládání

Výstupní data se ukládají do souboru příslušného formátu.

3.4.4 Testovací data

Tok dat při použití takto navrženého systému na testovací data je znázorněn dá obrázku 3.6.

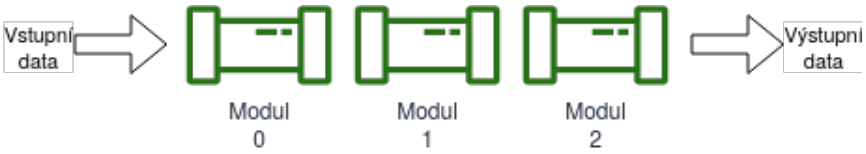


Obrázek 3.6: Tok dat v programu pro předzpracování dat

Nejprve proběhne konfigurace, následně se načtou data simulátoru dopravních prostředků, EKG a dalších zařízení pro sledování fyziologických dat, následně se odstraní vyskytující se chyby, proběhne interpolace a převzorkování, výstupní data se uloží do výstupního souboru.

3.5 Propojení modulů

Framework by měl sloužit pro předzpracování dat, přičemž by měl umožnit jednoduchou konfiguraci a přidání nových modulů. Příkladem architektury, která se může použít pro tyto účely, je datová roura (angl. data pipeline). Uživatel definuje odkud se data mají brát, kam se mají uložit a sadu modulů, kterými mají projít. Tímto se splní požadavek na rozšiřitelnost a modifikovatelnost. Funkční požadavky budou splněny implementací jednotlivých modulů. Model datové roury lze vidět na obrázku 3.7.



Obrázek 3.7: Model datové roury

Každý z těchto modulů může mít několik vstupů, načítací moduly by měly číst data z souborů, analytické - modifikovat je, ukládací - ukládat ve vhodném formátu nebo předávat dál ven ze systému.

3.5.1 Konfigurace a ovládání

Aby se splnily požadavky na jednoduchou konfiguraci a možnost automatizovaného použití, pro ovládání frameworku se používá konfigurační soubor ve formátu .json, podle kterého se nastavují jednotlivé části systému. Tento soubor definuje strukturu datové roury, do které jsou zapojené jednotlivé moduly a určuje nastavení pro každý z nich.

3.6 Výpočetní graf

Důležitým tématem je uspořádání modulů a jeho datová reprezentace. Uživatel frameworku by měl mít možnost nastavit strukturu datové roury podle své aplikace.

Pořadí, ve kterém se vykonávají jednotlivé moduly je určeno strukturou datové roury. Aby algoritmus uvnitř každého z modulů mohl proběhnout úspěšně, je potřeba, aby měl připravená data na vstupu.

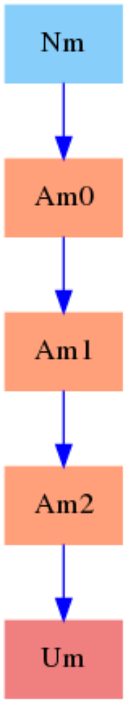
To znamená, že se nejprve vykonají moduly pro načítání dat, a poté moduly pro analýzu a ukládaní.

Obecně program se dá představit pomocí orientovaného grafu. Tento přistup se používá pro vizualizaci toku dat a analýzy vykonání každé části programu[33]. V případě řešené datové roury tento graf nebude obsahovat cykly to znamená bude acyklický.

Jednotlivé moduly jsou representované vrcholy grafu, hrany vyjadřují propojení mezi moduly, směr hran je směr toku dat. Přičemž ve vrcholech modulů, které načítají data ze souborů nebudou končit žádné hrany, budou mít vstupní stupeň rovný nule. Naopak

všechny ostatní vrcholy kromě vrcholu koncového modulu budou mít nenulové vstupní a výstupní stupně.

Model datové roury z obrázku 3.7 může být zobrazen v formě jako grafu:



Obrázek 3.8: Model datové roury z obrázku 3.7 v formě grafu

- Nm – načítací modul.
- Am0, Am1, Am2 – analytické moduly.
- Um – ukládací modul.

Formální zápis:

$$G = (H, U)$$

$H = \{Nm, Am0, Am1, Am2, Um\}$

$U = \{(Nm, Am0), (Am0, Am1), (Am1, Am2), (Am2, Um)\}.$

3.7 Algoritmus vykonání modulů

Hlavním pravidlem pro vykonání modulů je nutnost připravených dat na vstupu.

3.7.1 Algoritmus DFS

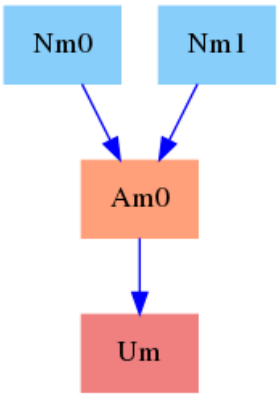
Jedním ze způsobů průchodu vrcholy grafu je tzv. algoritmus prohledávaní do hloubky (angl. Deep-First-Search). Začíná v kořenovém vrcholu, následně prohází každou větev tak hluboko jak může[34]. Algoritmus končí, když projde všechny vrcholy.

3.7.2 Variace algoritmu DFS

Pro proházení stromových struktur existují různé varianty algoritmu prohledávaní do hloubky, viz. 3.10[35]:

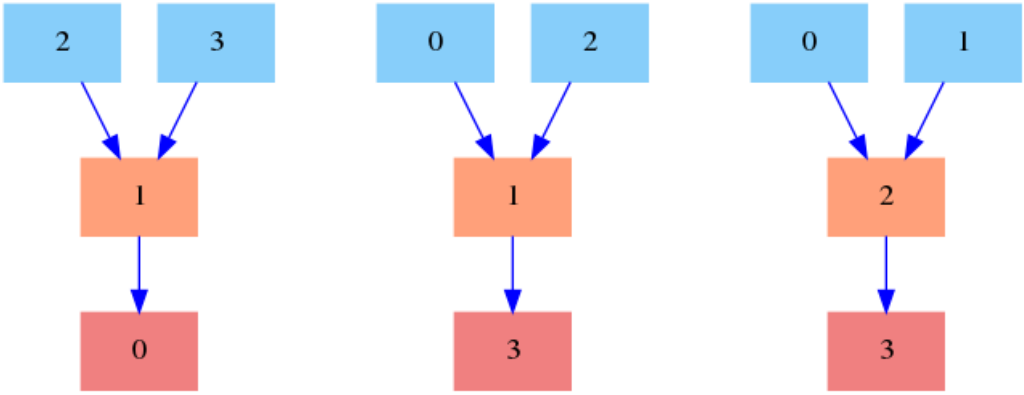
- Preorder – výstupem je pořadí ve kterém algoritmus vstupoval do uzlů, to znamená kořen-větev-větev. Algoritmus je definován následovně:
  1. Navštiv uzel.
  2. Jdi do podstromu zleva.
  3. Jdi do podstromu zprava.
- Inorder – výstupem je pořadí ve kterém uzly jsou zapojené ve stromu, to znamená větev-kořen-větev. Algoritmus je definován následovně:
  1. Jdi do podstromu zleva.
  2. Navštiv uzel.
  3. Jdi do podstromu zprava.
- Postorder – výstupem je pořadí ve kterém algoritmus vystupoval z uzlů, to znamená větev-větev-kořen. Algoritmus je definován následovně:
  1. Jdi do podstromu zleva.
  2. Jdi do podstromu zprava.
  3. Navštiv uzel.

Pokud by se algoritmus DFS aplikoval na graf, který je znázorněn na obrázku 3.9.



Obrázek 3.9: Příklad grafu na který se aplikuje algoritmus prohledávaní do hloubky

Pořadí průchodu uzly by vypadalo následovně:



(a) Preorder DFS (b) Inorder DFS (c) Postorder DFS

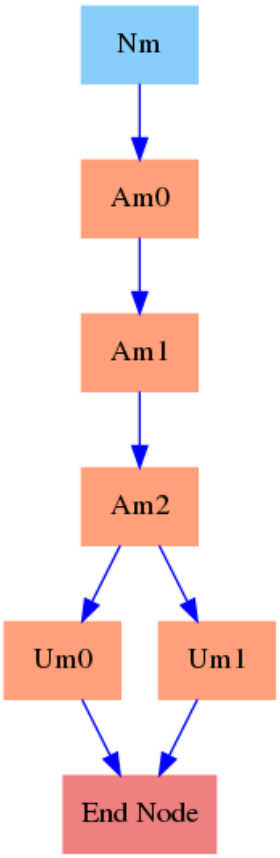
Obrázek 3.10: Pořadí ve kterém algoritmus DFS prohází uzly

3.7.3 DFS postorder

Pokud by se na graf datové roury aplikovala "postorder"varianta prohledávaní do hloubky, výstupem by bylo pořadí, ve kterém se musí volat moduly, aby modul, který se volá, vždy měl připravená data na vstupu.

Zmíněné algoritmy se mohou aplikovat jenom na stromové struktury, ale navržený model datové roury je orientovaným acyklickým grafem. V případě orientovaného acyklického grafu problém nastává, pokud obsahuje více než jeden vrchol, který má vstupní stupeň rovný nule. Jeden vrchol vždy bude kořenem, na který se aplikuje algoritmus, ostatní vrcholy tohoto typu již nebudou dosažitelné.

Tento problém se dá vyřešit tím způsobem, že se všechny vrcholy, ve kterých nekončí žádné hrany se připojí na jeden kořenový vrchol. Příklad takového zapojení lze vidět na obrázku 3.11.



Obrázek 3.11: Příklad grafu datové roury s dvěma ukládacími moduly

3.7.4 Implementace algoritmu

Pořadí ve kterém se moduly vykonávají je definováno rekurzivním algoritmem 1, jedním z argumentů je kořenový vrchol, který je znázorněn na obrázku 3.11 jako endNode:

Algoritmus 1 Algoritmus prohledávaní do hloubky, varianta 'postorder'

Require: *root*: Root node; $Nodes[\,]$: Empty list of nodes; G: The graph
Ensure: Add graph nodes to $Nodes[\,]$ in correct order

```
1: function DFSPOSTORDER(G, Nodes[ ], root)
2:     if root ∉ Nodes then
3:         for each node n ∈ G[root].neighbors() do
4:             DFSPOSTORDER(G, Nodes, n)
5:         end for
6:         Nodes.add(root)
7:     end if
8: end function
```

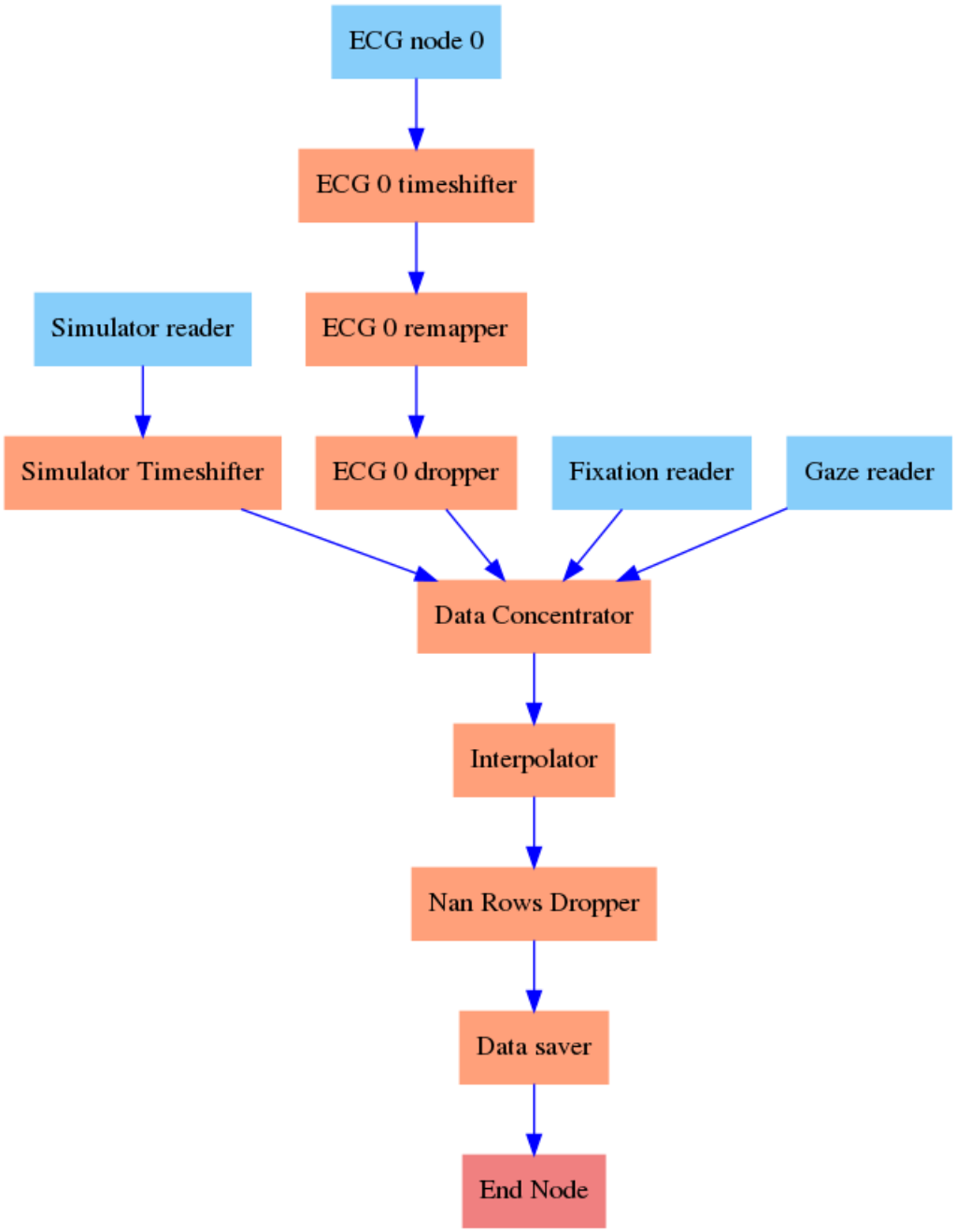
Proměnná Nodes bude obsahovat moduly ve správném pořadí. Následně se spustí metoda computeGraph(). Algoritmus této metody je vyjádřen zde2.

Algoritmus 2 Volání modulů

Require: $Nodes_1 \dots Nodes_N$

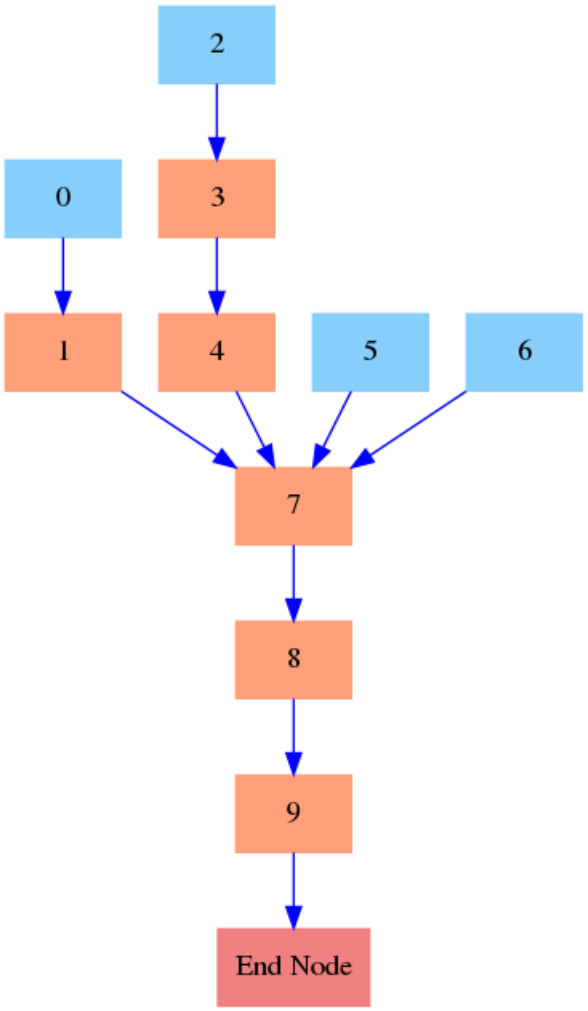
```
1: procedure COMPUTEGRAPH(Nodes[ ])
2:     N ← length(Nodes)
3:     for k ← 1 to N do
4:         COMPUTENODE(Nodes_k)
5:     end for
6: end procedure
```

Jako první se volají moduly v uzlech, ve kterých nekončí žádné hrany. Následně se spočítají ostatní moduly, přičemž pokud se modul už jednou volal, nebude se volat podruhé. V tomto případě graf bude mít strukturu, kterou lze vidět na obrázku 3.12.



Obrázek 3.12: Příklad grafu pro zpracování dat

Pořadí ve kterém se budou volat moduly bude vypadat způsobem, který je znázorněn na obrázku 3.13.



Obrázek 3.13: Pořadí vykonání modulů

Modře jsou označené moduly, které načítají data ze souborů, koncový modul má název "End Node".

3.8 Přenos dat mezi moduly

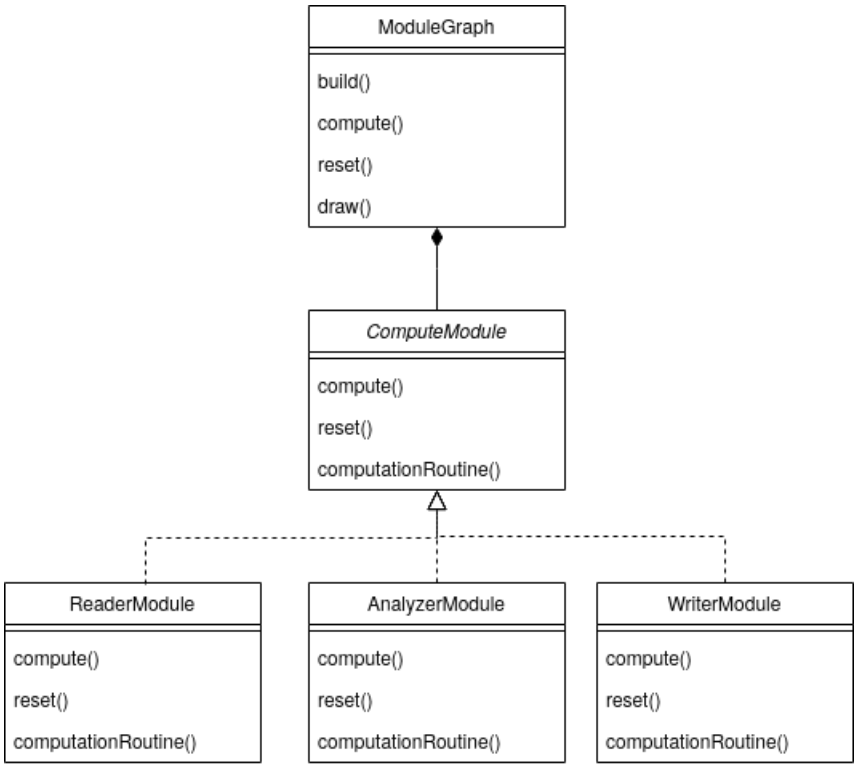
Přenos dat mezi jednotlivými částmi systému by měl být univerzální, aby šlo předávat jakákoliv data. Proto se jako formát zvolil slovník, obsahující metadata a jednotlivé signály ve formě objektu třídy DataFrame knihovny Pandas. Načtená data vždy musí obsahovat časové značky. Data v tomto formátu budou vypadat následovně:

```
outDict={
    "metadata":{},
    "dataframe":dataframe
    }
```

Načítací moduly vygenerují data v tomto formátu, analytické moduly je přečtou, modifikují a následně předají dál, poté výstupní moduly tato data uloží.

3.9 UML diagram

Propojení mezi třídy výpočetního grafu a výpočetních modulů je znázorněno v UML diagramu na obrázku 3.14.



Obrázek 3.14: Uml diagram systému

- **ModuleGraph**: Třída výpočetního grafu.
- **ComputeModule**: Třída výpočetních modulů.
- **ReaderModule, AnalyzerModule, WriterModule**: Třídy vyjadřující načítací, analytické a ukládací moduly.

3.9.1 Třída výpočetní graf

Výpočetní graf ovládá především uspořádaní modulů do grafu a jejich spuštění ve správném pořadí. Do konstruktoru se předá slovník, který definuje uspořádání modulů.

- **build()**: Metoda build() vytvoří požadovanou vnitřní strukturu grafu.
- **compute()**: Zavolá moduly, tvořící strukturu ve správném pořadí. Pořadí se sestaví na základě již popsaného "postorder"algoritmu prohledávaní do hloubky. Následně zavolá metodu compute() každého modulu.
- **reset()**: Smaže výsledky výpočtu každého modulu, což může být výhodné v případě iterativního použití objektu, a to tak, že zavolá metodu reset().
- **draw()**: Metoda draw() uloží do .png obrázku a vykreslí výpočetní graf.

3.9.2 Třída výpočetní modul

Třída výpočetních modulů je abstraktní třídou. Kromě metod, které jsou společné pro všechny moduly má i abstraktní metodu computationRoutine(), ve které probíhají výpočty specifické pro každý modul. Také obsahuje definice tříd výjimek (angl. exception), které vyjadřují informace o chybách, které mohou nastat za běhu programu. Jazyk Python nemá nativní podporu abstraktních tříd, proto se pro implementaci použila knihovna ABC.

Výjimky většinou vyjadřují nesprávný počet vstupních modulů.

- **MoreThanOneInputException**: Některé moduly nemohou mít více než jeden vstup, pokud k této situaci dojde, nastane právě tato výjimka.
- **NoInputNodesSpecifiedException**: Analytické a ukládací moduly vždy musí mít alespoň jeden vstup, jinak nebudou mít zdroj dat.
- **HasInputNodes**: Načítací moduly naopak nemohou mít vstupní moduly, protože generují data pro další část programu.

Třída obsahuje následující metody přičemž metoda computationRoutine() je abstraktní.

- **compute()**: Metoda compute() načte vstupní data, zpracuje je pomocí computationRoutine() a předá data dalšímu modulu.
- **reset()**: Smaže vstupní data.
- **computationRoutine()**: Je abstraktní metodou, která se musí implementovat v třídě potomku.

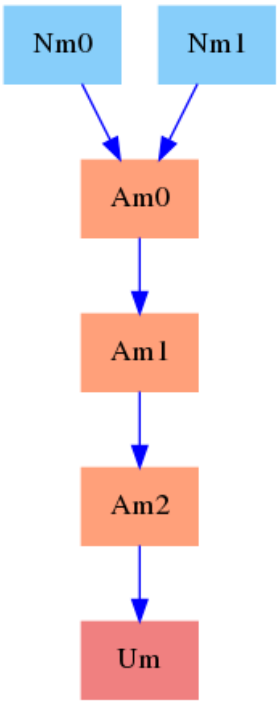
3.9.3 Ostatní třídy

Ostatní třídy jsou potomky třídy ComputeModule. Musí mít implementovanou metodu computationRoutine(), do které se zapouzdří příslušné algoritmy. Lze je rozdělit do tři základních skupin podle činností které vykonávají:

- **Třídy pro načítaní**: Načítají data ze souborů nebo i z jiných zdrojů.
- **Třídy pro analýzu**: Mohou plnit různé funkce, mezi které patří například čištění dat nebo jejich úpravy.
- **Třídy pro ukládání**: Ukládají data do souborů nebo je přesměrují dál ven z frameworku.

3.10 Načítací moduly

Do načítáních modulů nevstupují data z jiných modulů, jejich hlavním účelem je generovat data pro následující analýzu. Mohou toto udělat například načtením dat ze souboru, nebo z proměnné v jiné části programu. Vrcholy grafu, které vyjadřují tyto moduly, budou mít vstupní stupeň rovný nule. Příklad grafu s několika načítacími moduly je znázorněn na obrázku 3.15.



Obrázek 3.15: Příklad grafu s dvěma nečítacími moduly Nm0 a Nm1

3.10.1 Modul pro načtení dat z proměnné ve skriptu

Umožňuje především propojení s jinými částmi programu a zabudovaní datové roury do většího programu. Proměnná se musí objevit jako globální, uživatelský program poté načte data do proměnné a zavolá objekt reprezentující výpočetní graf. Podporované nastavení:

- Název proměnné, ze které se načtou data.

3.10.2 Modul pro načtení CSV a TXT

Pro načítaní .csv a .txt souboru se použila knihovna Pandas. Podle nastavení se načítají soubory s příslušnými oddělovači. Data v souborech musí být uspořádaný do sloupců. Jelikož časová značka je unikátní hodnotou, která jednoznačně definuje každý řádek, časové značky se načtou jako index, toto pak umožní jednodušší a rychlejší zpracování dat. Podporované nastavení:

- Cesta k souboru.
- Typ oddělovače.

- Název sloupce, ve kterém je uložen čas.
- Nové názvy pro sloupce.

3.10.3 Modul pro načtení Dicom

Dicom modul nejprve rozezná typ zařízení, které nahrávalo data, pak podle typu zařízení určí jak jsou data uspořádané. V rámci tohoto projektu, jediné zařízení, které nahrává data ve formátu dicom je EKG s názvem SE-1515 od firmy Edan instruments. Program podporuje načítání dicom souborů, které byly vytvořeny tímto přístrojem. Nicméně SW umožňuje jednoduché rozšíření i pro jiné systémy. Podporované nastavení:

- Cesta k souboru.

3.10.4 Modul pro načtení Hdf5

Modul pro načítání hdf5 využívá knihovny h5py. Uživatel musí přiřadit cestu k hdf5 souboru a poté cestu k příslušné skupině uvnitř hdf5 souboru. Časové značky se načtou jako index. Podporované nastavení:

- Cesta k souboru.
- Cesta k datům uvnitř souboru.
- Název sloupce, ve kterém je uložen čas.

3.11 Analytické moduly

Analytické moduly se používají pro zpracování dat, mohou mít libovolnou funkcionalitu. Příkladem použití je výpočet a přidávání nových dat, nebo modifikace již existujících. Vrcholy výpočetního grafu, které reprezentují tyto moduly, mají nenulové vstupní a výstupní stupně.

3.11.1 Aritmetický modul

Umožňuje použití základních aritmetických operací: sčítaní, odečítaní, násobení, dělení, spočítaní modula. Může se používat např. pro posun všech časových značek o konstantní hodnotu. Podporované nastavení:

- Cesta k souboru.
- Cesta k datům uvnitř souboru.
- Název sloupce, ve kterém je uložen čas.

3.11.2 Modul pro změnu názvů sloupců

Změní názvy sloupců podle nastavení. Podporované nastavení:

- Jaký název se změní za jaký.

3.11.3 Modul pro mazaní sloupců

V některých případech je potřeba smazat určitá data, tento modul umožňuje pravě toto. Podporované nastavení:

- Názvy sloupců, které se musí smazat.

3.11.4 Modul pro mazání řádků, které obsahují Nan hodnoty

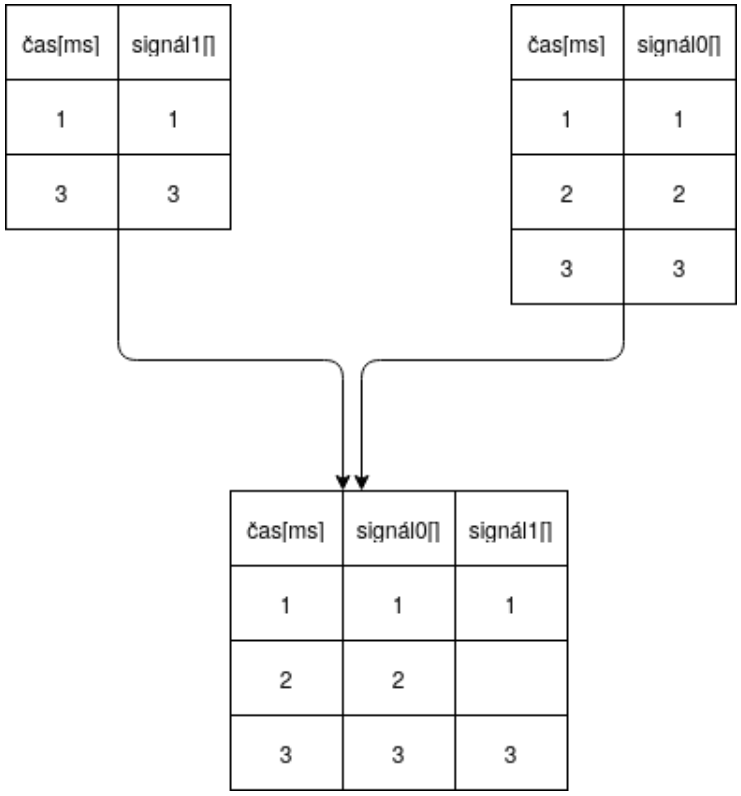
Smaže řádky, které obsahují hodnoty Nan. Podporované nastavení:

- Názvy sloupců, které se musí upravit.

3.11.5 Modul pro spojení dat

Tento modul spojí všechny vstupní data do jediného baličku, názvy sloupců ve vstupních datech se nesmí opakovat. Spojení probíhá podle časových značek. To znamená pro každou časovou značku přítomnou v signálech se najdou hodnoty ostatních signálů, přičemž pokud nebudou existovat, budou místo nich prázdná místa. Princip fungování je podobný

příkazu OUTER JOIN (vnější spojování) v SQL jazyku. Funkcionalita tohoto modulu je znázorněna na obrázku 3.16.



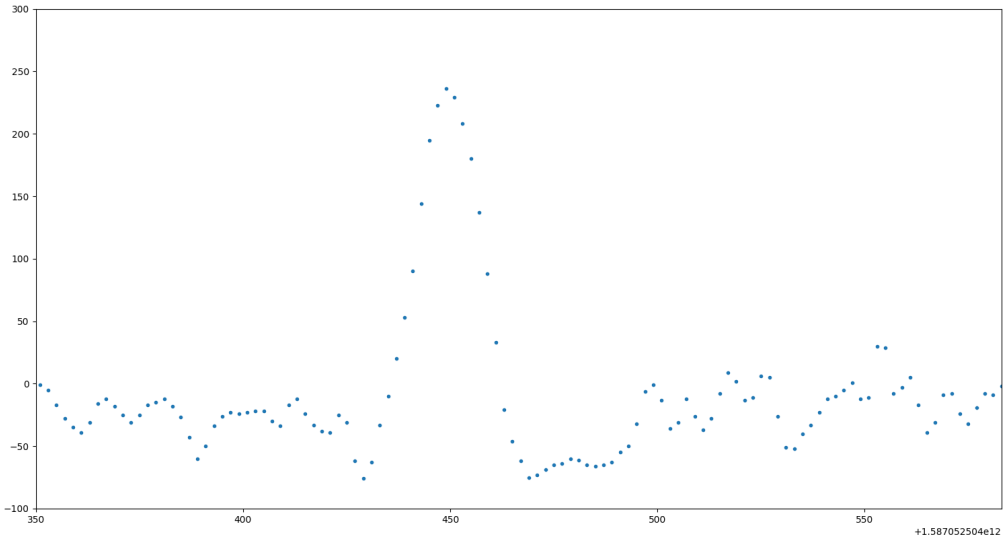
Obrázek 3.16: Spojení signálů

3.11.6 Modul pro interpolaci a převzorkování

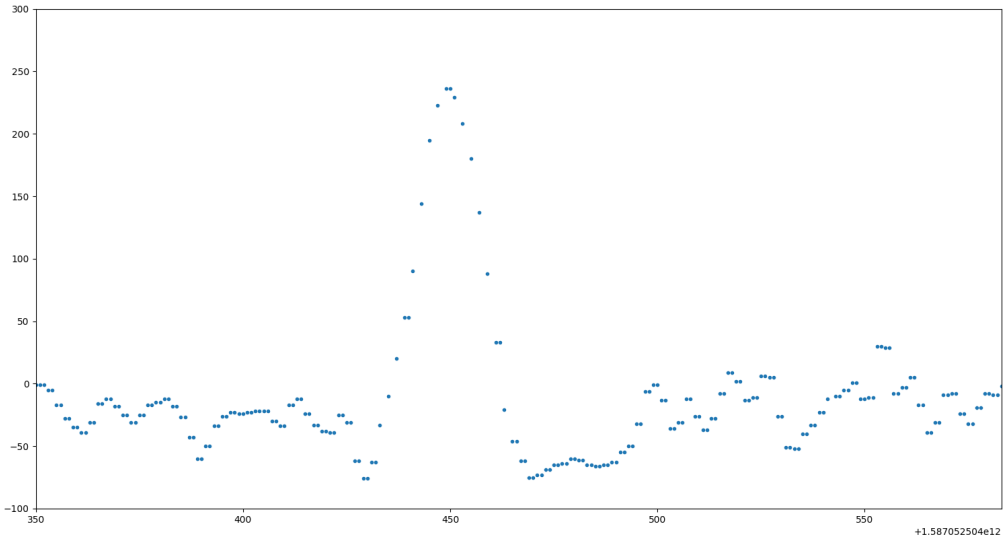
Hlavní funkcionalitou interpolačního modulu je vyplnění chybějících hodnot signálů. Vstupní data mohou být reprezentována událostmi u kterých interpolace není potřeba, proto v nastavení tohoto modulu je možné vybrat jaká data se budou interpolovat. Přičemž je také možné zvolit i interpolační metodu. Implementované jsou následující metody: metoda nejbližšího sousedu (viz. 3.18), lineární interpolace (viz. 3.19), interpolace polynomem n-stupně (viz. 3.20). Interpolované signály se následně převzorkují. Podporované nastavení:

- Metoda.
- Názvy sloupců, na které se musí interpolace aplikovat.

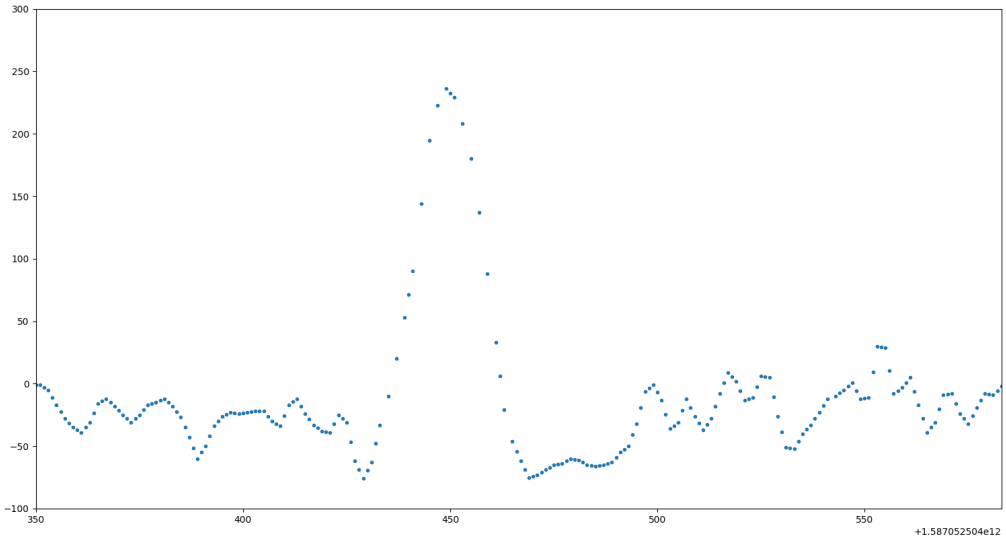
Příklad interpolace a převzorkování signálu pomocí tohoto modulu je znázorněn dále. Vstupní signál lze vidět na obrázku 3.17.



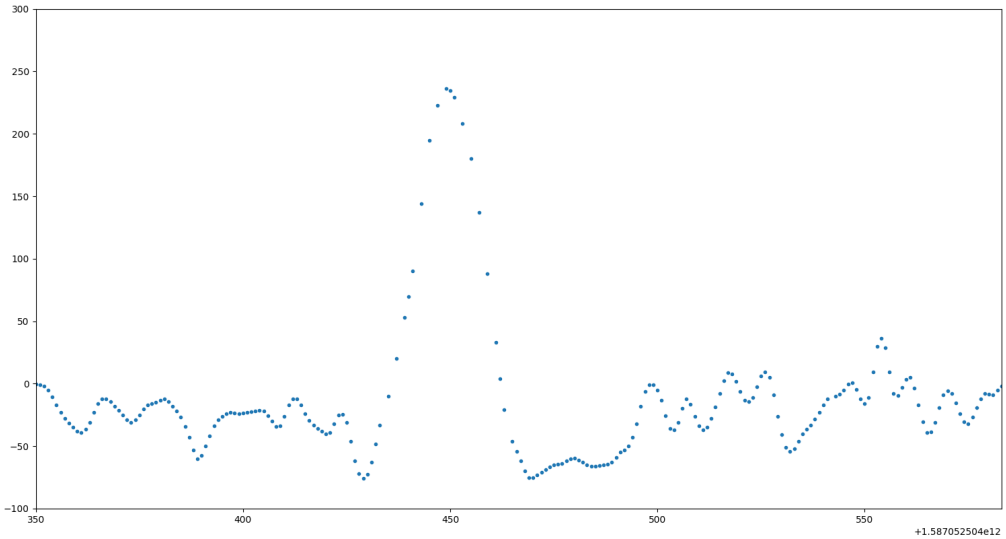
Obrázek 3.17: Neinterpolovaný signál, který se snímal frekvencí 500 Hz.



Obrázek 3.18: Signál po interpolaci metodou nejbližšího souseda a převzorkování.



Obrázek 3.19: Signál po lineární interpolaci a převzorkování.



Obrázek 3.20: Signál po interpolaci polynomem druhého řádu a převzorkování.

3.12 Ukládací moduly

Vrcholy grafu asociované s ukládacími moduly budou mít vstupní stupeň rovný 1, protože v nich bude končit jediná hrana, která začala v kořenovém vrcholu.

3.12.1 Modul pro ukládaní do souboru

Ukládací modul uloží signály ve zvoleném formátu. Možné formáty jsou .h5, .csv a .txt. Podporované nastavení:

- Název výstupního souboru.
- V případě formátu hdf5, vnitřní cesta souboru.

V případě HDF5 každý signál je reprezentován datasetem. Metadata obsahují informace o vzorkovací frekvenci, typu signálu, délce signálu, začátku a konci měření.

3.12.2 Modul pro ukládaní do proměnné ve skriptu

Výstupní data se uloží do proměnné programu s určitým názvem. Jiná část programu může tato data následně využívat. Podporované nastavení:

- Název proměnné.

3.12.3 Modul pro zobrazení dat

Tento modul využívá knihovny matplotlib pro vykreslení dat z vybraných sloupců, umožňuje mimo jiné i vykreslení několika grafů do jednoho obrázku, prohledávání a export grafů. Podporované nastavení:

- Názvy sloupců, které se musí vykreslit.
- Jestli se mají vykreslit do jednoho grafu anebo do několika.

3.12.4 Modul pro zobrazení dat v reálném čase

Má podobnou funkcionalitu jako předchozí modul, jen s tím rozdílem, že v případě dávkového zpracování dat, nevytváří pro každou dávku nové okno. Toto umožňuje iterativní volaní a postupné zobrazovaní dat. Podporované nastavení:

- Názvy sloupců, které se musí vykreslit.
- Jestli se mají vykreslit do jednoho grafu nebo do více grafů.

3.13 Konfigurační soubor

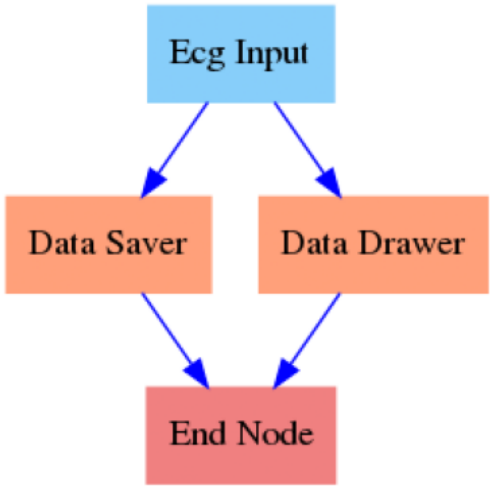
Konfigurační soubor má formát json, a určuje, jak pořadí spuštění jednotlivých modulů, tak i nastavení každého modulu. Příklad jednoduchého konfiguračního souboru je popsán dál.

```
{
    "Ecg Input":
        {
            "inputFile": "004-26062020-093244.DCM",
            "inputNodes": [],
            "type": "dicomReader"
        },
    "Data Drawer":
        {
            "type":"1DDrawer",
            "columnsInput":["Ecg1"],
            "subplot":false,
            "inputNodes":["Ecg Input"]
        },
    "Data Saver":
        {
            "type":"dataSaver",
            "outputFile":"output.csv",
            "append":false,
            "delimiter":";",
            "inputNodes":["Ecg Input"]
        },
```

```
"End Node":
    {
        "type":"endNode",
        "inputNodes":["Data Saver","Data Drawer"]
    },
"outputNode":"End Node"
}
```

Tento program načte ECG data ze souboru ve formátu dicom, zobrazí je, a následně uloží jako soubor ve formátu .csv.

Výpočetní graf, který vznikne na základě tohoto nastavení je znázorněn na obrázku 3.21.



Obrázek 3.21: Výpočetní graf testovacího programu

Struktura s názvem 'ECG Input' slouží pro načítaní souboru ve formátu dicom, jejím vstupem je cesta k souboru.

'Data Drawer' přečte data z modulu 'ECG Input' a vykreslí data ze sloupce 'Ecg1' do grafu.

Modul 'Data Saver' též vezme data z modulu 'ECG Input' a následně uloží je do souboru s názvem 'output.csv'

Poté se program ukončí.

3.14 Spuštění

Navržený program se spouští z příkazového řádku, též ho jde použít jako modul v jiném programu v jazyce Python. Příklad spouštěcího příkazu:

```
$ python3 main.py -c config.json
```

Kde config.json je cesta k konfiguračnímu souboru.

3.15 Ověření funkcionality

V rámci experimentu, ve kterém se sbírala data, se řešila rehabilitace řidičů po mozkové obrně. Cílem projektu bylo navržení řidičské rehabilitace pro osoby po poškození mozku s využitím automobilového trenažéru. Zjišťovalo se jaké jsou jejich aktuální řidičské schopnosti a zda jsou dostatečné pro bezpečné řízení vozidla. Veškerá změřená data byla anonymizována.

3.15.1 Použité zařízení

- EKG Kognitivní zátěž se rapidně zvyšuje během vykonávání složitějších a komplexnějších činností typu řízení automobilu[36]. Některé parametry vypočtené z EKG křivky korelují s kognitivní zátěži, což se potvrdilo v mnoha studiích[37]. Z tohoto důvodu se záznam elektrické aktivity srdce použil i v tomto projektu.
- Eyetracker Zařízení pro sledování očí se použilo pro určení toho, kam se člověk díval během jízdy.
- Armtracker Pomocí tohoto zařízení se nahrávaly pohyby rukou.
- Simulator dopravních prostředků Na simulátoru se zaznamenávala jízda za různých podmínek a v různých prostředích, mezi nahrávanými daty byla například i trajektorie jízdy.

Jistým problémem je, že některé z těchto zařízení již mají ukládací protokoly předdefinované, a nelze je tím pádem upravovat, z čehož plyne, že po experimentu musí tato data projít synchronizací a formátováním.

3.15.2 Popis měření

Všechna vstupní data se nahrávala na stejném počítači.

Simulátor dopravních prostředků zaznamenává jízdu člověka v simulovaném prostředí. Výstupní data jsou ve formátu .txt. Tato data nemají názvy sloupců, časová razítka všech signálů jsou posunutá o dvě hodiny, vzorkovací frekvence je přesně 500 Hz.

Elektrokardiogram se nahrával do dicom souboru, data se měřila pomocí EKG SE-1515 od firmy Edan Instruments, vzorkovací frekvence též byla 500 Hz. Tento přistroj ovšem neumožňuje záznam delší než 30 minut, proto během jednoho měření mohlo vzniknout několik souborů.

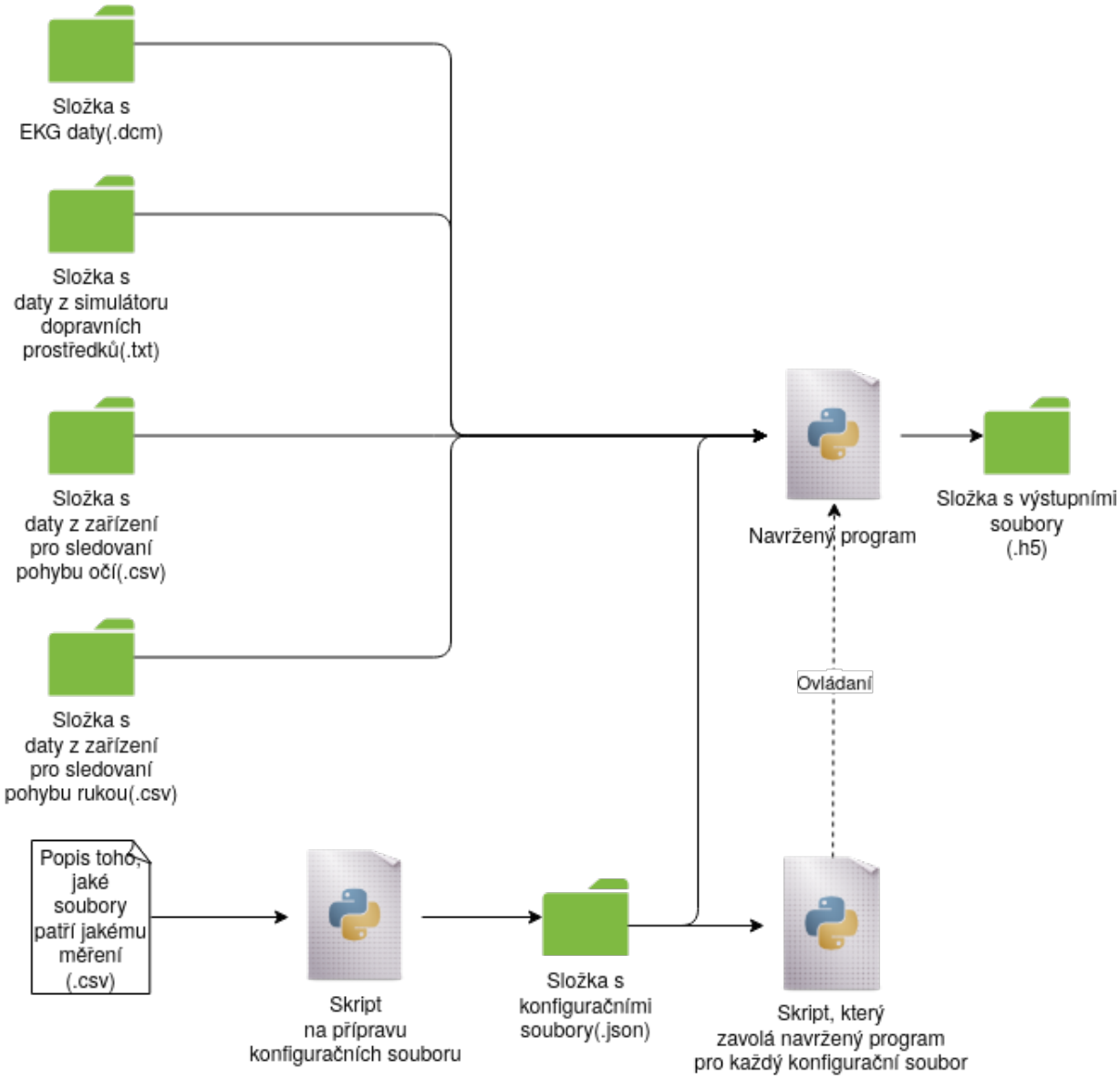
Data ze zařízení pro sledování očí a rukou se nahrávala do .csv souboru, přičemž u některých měření nemusí být obě zařízení zapnuté. Data z těchto zařízení mají podobu událostí, z čehož plyne, že nemají konstantní vzorkovací frekvenci.

Též se v rámci experimentu vytvořil informační soubor, který obsahuje informace o tom který soubor patří kterému měření.

3.15.3 Použití navrženého SW

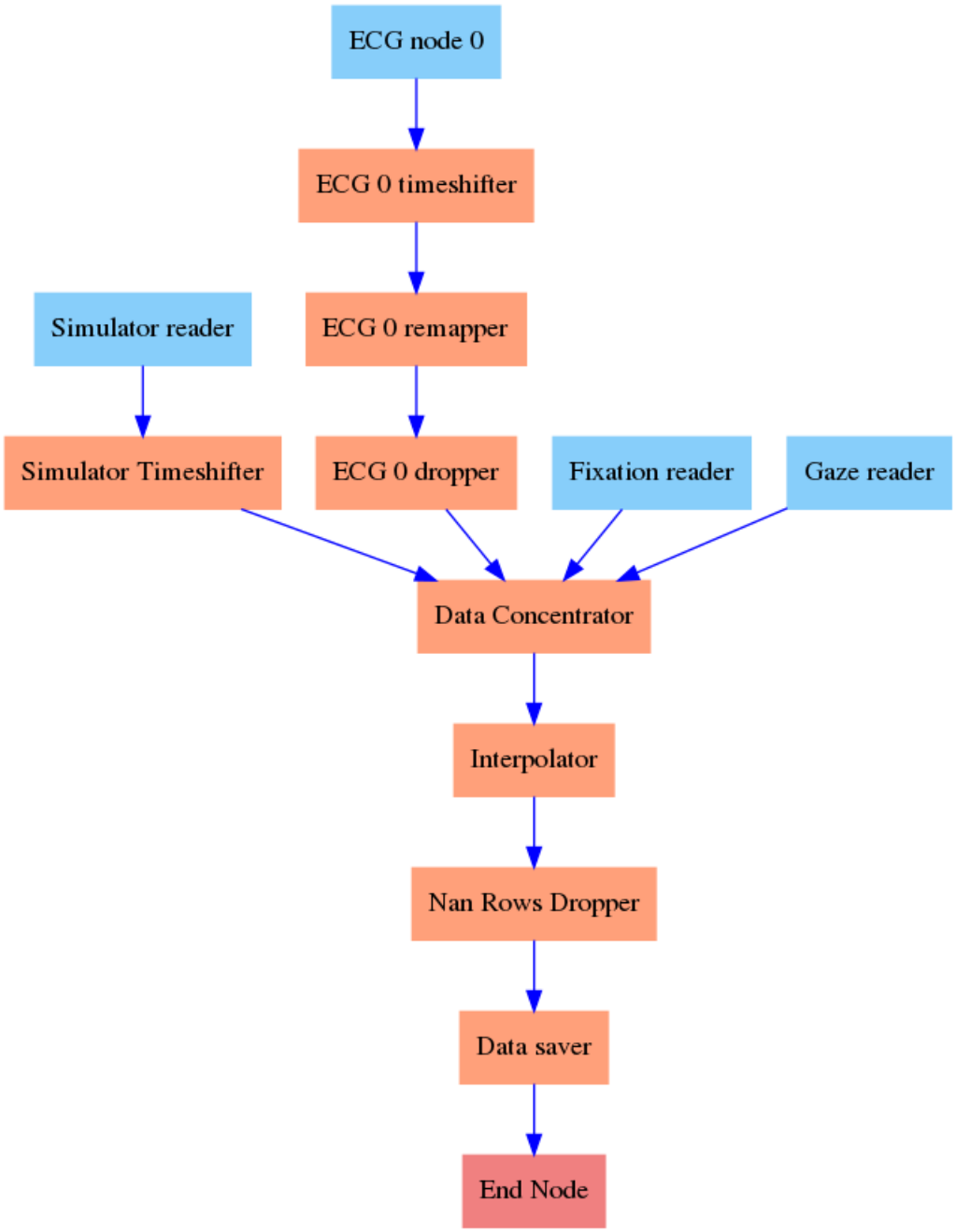
Navržený program přijme na vstup konfigurační soubor, podle kterého načte správné datové soubory, následně provede předzpracování a uloží výstupní data ve zvoleném formátu. Proto je potřeba na základě informačního souboru vytvořit konfiguraci pro každé měření, a následně spustit navržený program s každou z těchto konfigurací. Jako základna, podle které proběhne synchronizace se použije sloupec, který obsahuje časové značky simulátoru dopravních prostředků.

Pro automatické generovaní konfiguračních souborů a pro spuštění programu s každým souborem se napsal skript v jazyku Python. Propojení vnitřních struktur systému během testování lze vidět na obrázku 3.22.



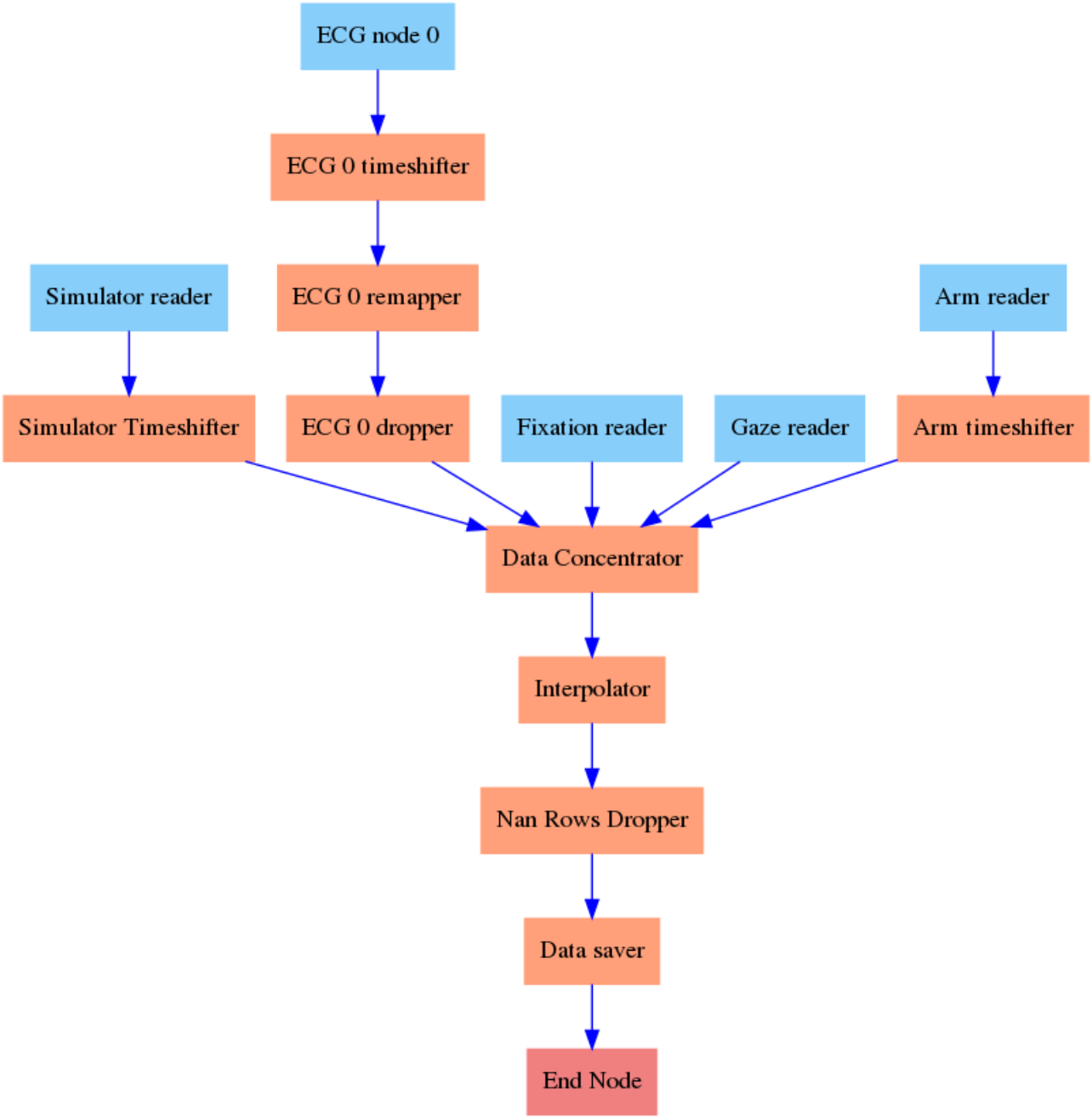
Obrázek 3.22: Diagram testování

Jelikož počet souborů pro každé měření se lišil, grafy vytvořené na základě vstupních dat měly rozdílnou strukturu. V nejjednodušším případě během měření se nahrávala data ze zařízení pro sledování pohybu očí, simulátoru dopravních prostředků a EKG. Model grafu, který vzniknul pro tento typ měření je znázorněn na obrázku 3.23.



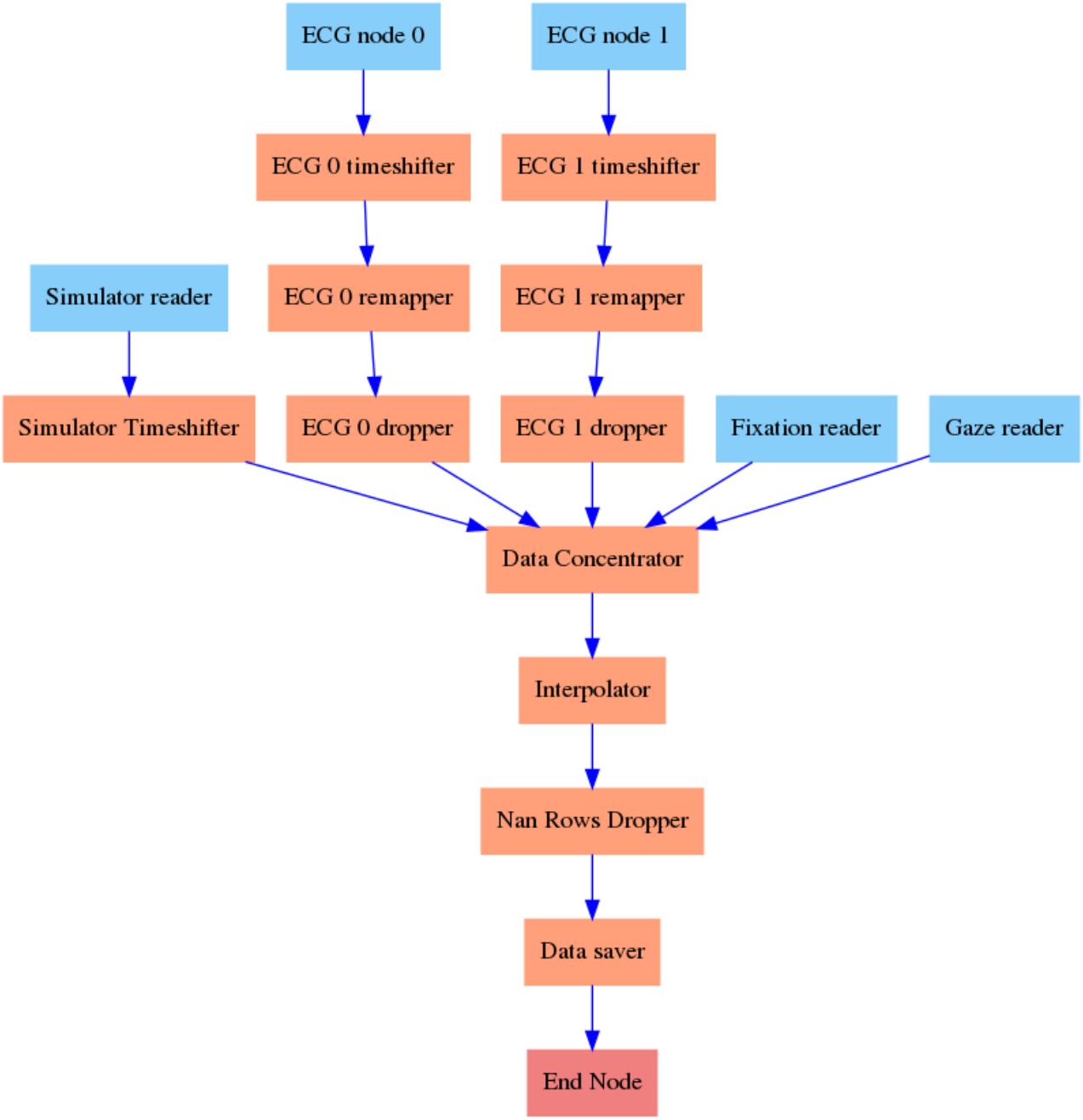
Obrázek 3.23: Struktura programu č. 1

U některých měření se také nahrávala data ze zařízení pro sledování pohybu rukou. Model výpočetního grafu, který se použil pro zpracování dat z tohoto měření lze vidět na obrázku 3.24 nebo na obrázku 3.26.

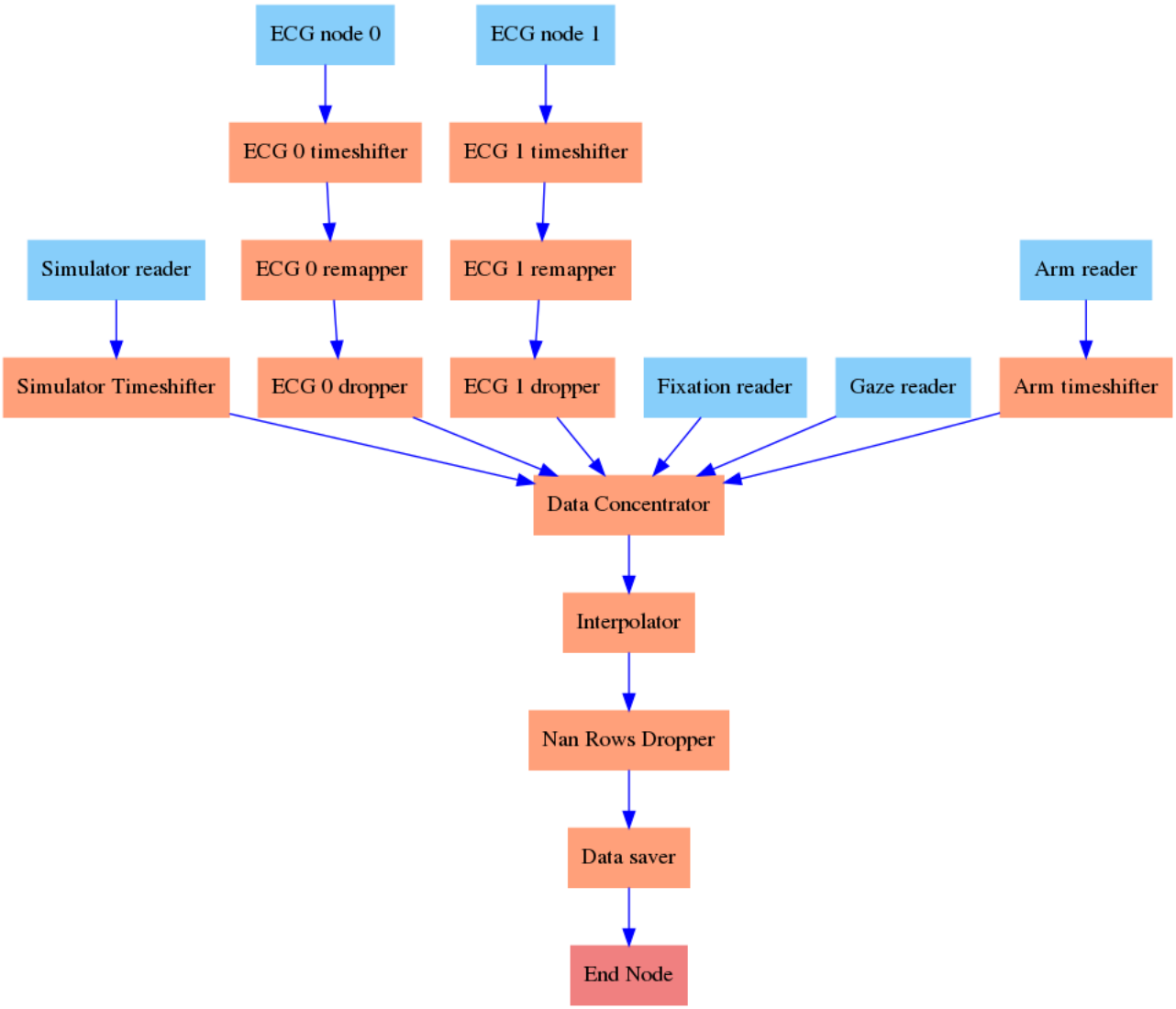


Obrázek 3.24: Struktura programu č. 2

Přičemž mohlo dojít k takové situaci, že několik EKG záznamů patřilo jednomu měření. Důvodem je to, že zařízení, pomocí kterého se provádělo měření neumožňuje nahrávaní záznamu delšího než 30 minut. Strukturu programu v tomto případě lze vidět na obrázku 3.25.



Obrázek 3.25: Struktura programu č. 3



Obrázek 3.26: Struktura programu č. 4

Vzhledem k tomu, že data ze simulátoru dopravních prostředků byla posunuta o dvě hodiny, musela se provést operace která vykompenzuje tuto chybu. Modul s názvem 'Simulator Timeshifter' posune časové značky zpátky a tím eliminuje zmíněnou chybu. Posunutí proběhne pomocí modulu pro vykonaní aritmetických operací. Konkrétně pomocí jeho aplikování na sloupec, který obsahuje časové značky.

U EKG záznamů se vyskytovala stejná chyba, proto se též musely aplikovat moduly pro posunutí časových značek. Kromě toho se u EKG musely změnit názvy sloupců, důvodem je to, že pokud jednomu měření patří dva záznamy, vznikalo několik EKG tabulek, která

měla stejné názvy sloupců. Další transformace, která proběhla bylo odstranění některých sloupců, která neobsahovala žádná data.

U zařízení pro sledování rukou se rovněž musel posunout čas.

Data ze zařízení pro sledování očí se načetla bez jakýkoliv změn.

Všechna data se spojila do jedné tabulky v modulu 'Data Concentrator', spojení proběhlo na základě časových značek, přičemž pokud nějaký ze signálů neměl definovanou hodnotu v určitém čase, místo něj byla hodnota Nan.

Dalším krokem je interpolace a převzorkování, jelikož hodnota vzorkovací frekvence pro EKG je 500 Hz, ale pro simulátor dopravních prostředků je to 1000 Hz. První variantou je interpolovat EKG signál a následně jej převzorkovat, druhou variantou je převzorkovat data ze simulátoru dopravních prostředků. Jelikož není požadavek na zmenšení objemu dat, použila se první varianta. Zvolenou interpolační metodou je lineární interpolace.

Následně se data očistila od chybějících hodnot, pokud takové hodnoty byly, a proběhlo uložení výstupního souboru ve zvoleném formátu.

4 Výsledky

V rámci této práci se navrhnul modulární program pro předzpracování dat. Testování proběhlo na anonymizovaných datech simulátoru dopravních prostředků a zařízení pro záznam fyziologických dat. Celý algoritmus fungování lze rozdělit do tři bloků: načítání dat, jejích zpracování, ukládání. Nastavení frameworku probíhá pomocí konfiguračního souboru ve formátu json.

4.1 Charakteristiky navrženého frameworku

Navržený framework má následující funkcionality:

- Načítání vstupních souboru ve formátech: .csv, .txt, .dcm, .h5.
- Nastavení pomocí konfiguračního souboru ve formátu json.
- Možnost opravy následujících druhů chyb:
  - Posunutí času.
  - Chybějící názvy sloupců.
- Převzorkování a interpolace jednou z následujících metod:
  - Lineární interpolace
  - Metoda nejbližšího sousedu.
  - Interpolace polynomem n-tého řádu.
- Možnost vizualizace dat.
- Možnost zabudování frameworku do většího systému.
- Ukládání výstupu ve formátech: .csv, .h5.

SW též umožňuje doplnění modulů pro načítání a ukládání dat v jiných formátech nebo dalších modulů pro předzpracování nebo vizualizaci dat.

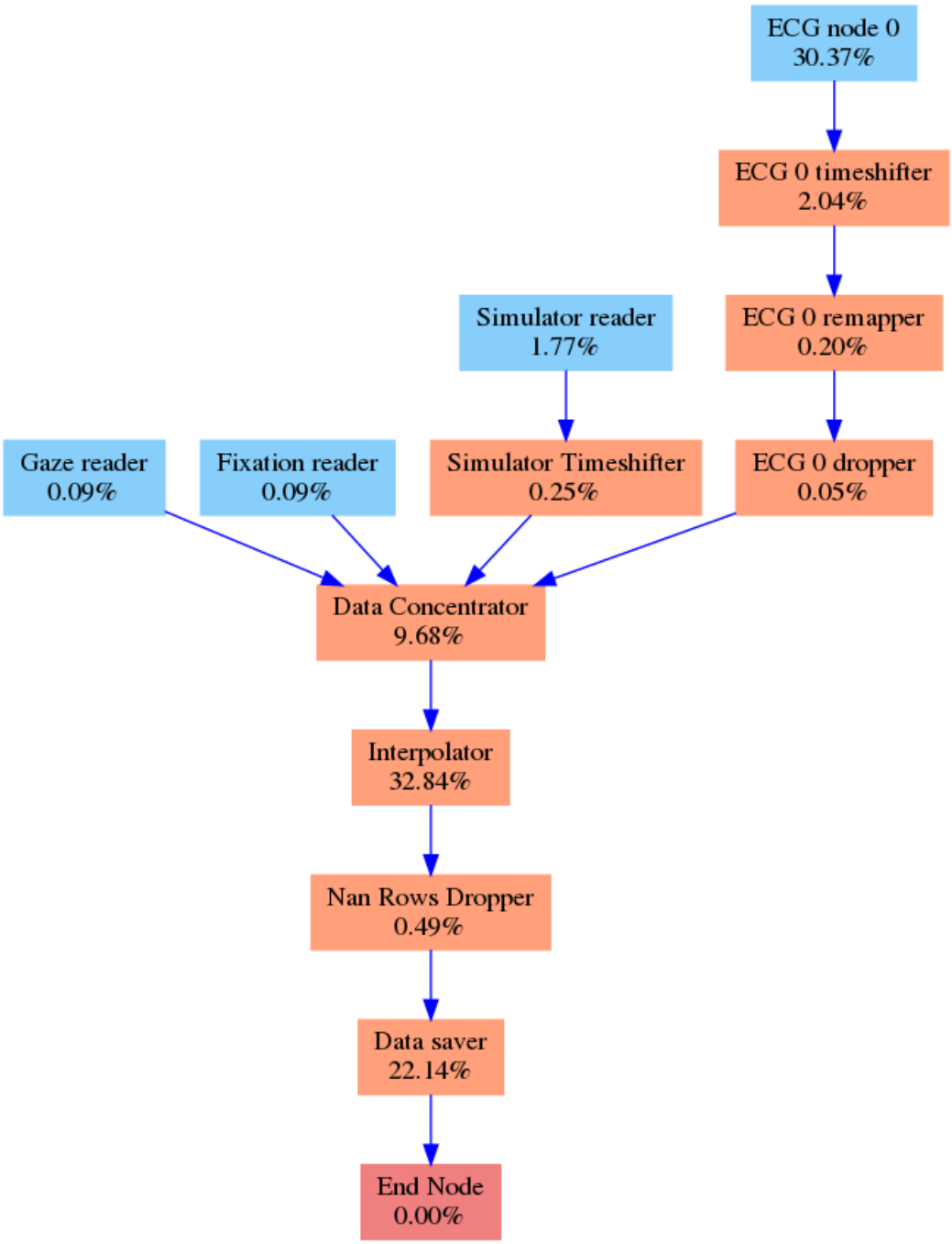
4.2 Splnění požadavku

Splnění funkčních požadavků je patrný z minulé kapitoly. Mezi nefunkční požadavky patřily rozšiřitelnost a jednoduchá konfigurace. Rozšiřitelnost je zajištěná tím, že uživatel může jednoduše přidávat rozšiřující moduly za předpokladu, že se bude používat abstraktní třída ComputeModule. Konfigurace probíhá pomocí souboru ve formátu json, který určí jak tok dat v programu, tak nastavení pro jednotlivé moduly. Framework umožňuje automatické generování tohoto souboru jiným programem, z čehož plyne možnost jednoduchého propojení s jiným softwarem. Další možností integrace je přímé použití frameworku v zdrojovém kódu externího programu.

4.3 Testování

Navržený program se úspěšně otestoval na datech ze simulátoru dopravních prostředků. Vygenerovaná výstupní data jsou ve formátu hdf5, a jsou připravena na další analýzu.

Zpracování dat probíhalo na počítači Dell Precision M6800 s procesorem Intel(R) Core(TM) i7-4810MQ CPU. Po zpracování 87 záznamů střední počet cyklů procesoru zpracování dat z jednoho měření byl $47123007577 \pm 12193025279$, což odpovídalo cca 16.73 ± 4.30 sekundám, přičemž velikost dat na vstupu byla 34.01 ± 11.46 MB. Na obrázku 4.1 je znázorněno rozložení cyklů procesoru podle modulu.



Obrázek 4.1: Příklad grafu datové roury s dvěma ukládacími moduly

5 Diskuze

Navržený framework poskytuje základní nástroje pro předzpracování dat, též podporuje rozšíření a modifikace podle potřeby. Umožňuje jednoduché nastavení a zabudování do většího systému pro automatizované zpracování dat. Vytvořený framework by se mohl úspěšně používat pro předzpracování experimentálních dat, kde se často pracuje s daty, která se nasbírala na různých zařízeních.

Na rozdíl od nástrojů Cleanix, BigDansing, Katara a Eracer které se popisovaly v kapitole 2, navržený systém je určen na práci s relativně malými objemy dat.

Dalším rozdílem je modifikovatelnost navrženého frameworku, možnost jednoduchého rozšířeni a přidávání nových modulů.

Kromě standardních funkcionalit typu odstranění duplikátních záznamů, a náhrady chybějících hodnot, které mají např. systémy Cleanix a BigDansing, navržený framework umožňuje vykreslení záznamů do grafu a interpolaci s převzorkováním.

Vytvořený framework se též od zmíněných nástrojů liší tím, že se zaměřuje spíše na předzpracování časových řad.

5.1 Možné další směřování práce

Jednoduchá struktura programu dovoluje přidání nových modulů pro import, export vizualizaci nebo analýzu dat.

Dicom modul je navržen tak, že pro úspěšné načítání dat z určitého zařízení je vždy potřeba přidávat kód, který určuje způsob načítání dat. V současné době jediným podporovaným přístrojem je EKG SE-1515 od firmy Edan instruments.

Textové příznaky, by se mohly automatický nahrazovat za čísla. Slovník text-číselný identifikátor by mohl být součástí metadat výstupního souboru. Výstupní soubor by poté byl plně připraven na následující analýzu pomoci např. metod strojového učení.

Architektura frameworku podporuje implementaci nadstavby pro grafickou konfiguraci frameworku. Nadstavba by mohla umožňovat vizuální nastavení výpočetního grafu. Vystup by se uložil ve formátu json. Navržený framework by mohl následně použit tento soubor jako konfigurační.

6 Závěr

Byl vytvořen program pro import a synchronizaci dat z různých systémů. Na základě zadání a cílů práce byly určeny požadavky.

Veškeré potřebné funkcionality se shrnuly do funkčních požadavků, další nároky se zapsaly do nefunkčních požadavků.

Na základě těchto dat se vytvořila architektura programu. Jejími základními součástmi v souladu s ETL modelem jsou část pro načítaní dat, část pro analýzu a část pro ukládání. Vytvořená architektura předpokládá, že se výstupní program skládá z modulů, které implementují požadovanou funkcionalitu, tím se zajišťuje rozšiřitelnost a modifikovatelnost frameworku.

Všechny moduly se zapojí do grafové struktury. Nastavení grafové struktury probíhá pomocí konfiguračního souboru ve formátu json. Tento soubor určuje jak pořadí vykonání modulů, tak i nastavení pro jednotlivé moduly.

Podle vytvořené architektury byl implementován framework. Jednotlivé moduly se implementovaly tak, aby se splnily veškeré funkční požadavky.

Program se otestoval na záznamech ze simulátorů dopravních prostředků. Pro export dat se zvolil binární formát HDF5, který umožňuje rychlé a efektivní vyhledávání vzorků v čase. Program dovoluje integraci nových formátů pro import, export, zpracování a vizualizaci.

Použitá literatura

[1] Alfaro-Ponce, M.; Chairez, I.: Continuous and recurrent pattern dynamic neural networks recognition of electrophysiological signals. *Biomedical Signal Processing and Control*, ročník 57, 2020, ISSN 17468094, doi:10.1016/j.bspc.2019.101783. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1746809419303647

[2] Lepot, M.; Aubin, J.-B.; Clemens, F.; aj.: Interpolation in Time Series. *Water*, ročník 9, č. 10, 2017: s. 46–54, ISSN 2073-4441, doi:10.3390/w9100796. Dostupné z: http://www.mdpi.com/2073-4441/9/10/796

[3] Taktak, A.; Long, D.; Ganney, P. S.; aj.: *Clinical Engineering*. Academic Press, třetí vydání, 2020, ISBN 978-0-08-102694-6.

[4] Lee, M. L.; Ling, T. W.; Low, W. L.: IntelliClean. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*, 2000: s. 290–294, doi:10.1145/347090.347154. Dostupné z: http://portal.acm.org/citation.cfm?doid=347090.347154

[5] Hellerstein, J. M.; Raman, V.: Potter's Wheel: An Interactive Data Cleaning System. September 2001: str. 11, doi:https://www.researchgate.net/publication/2380270.

[6] Ridzuan, F.; Zainon, W. M. N. W.: A Review on Data Cleansing Methods for Big Data. *Procedia Computer Science*, ročník 161, 2019: s. 731–738, ISSN 18770509, doi:10.1016/j.procs.2019.11.177. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1877050919318885

[7] Wang, H.; Li, M.; Bu, Y.; aj.: Cleanix. *ACM SIGMOD Record*, ročník 44, č. 4, 2016-05-09: s. 35–40, ISSN 0163-5808, doi:10.1145/2935694.2935702. Dostupné z: https://dl.acm.org/doi/10.1145/2935694.2935702

[8] Khayyat, Z.; Ilyas, I. F.; Jindal, A.; aj.: BigDansing. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015-05-27: s. 1215–1230, doi:10.1145/2723372.2747646. Dostupné z: https://dl.acm.org/doi/10.1145/2723372.2747646

[9] Chu, X.; Morcos, J.; Ilyas, I. F.; aj.: KATARA. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, 2015-05-27: s. 1247–1261, doi:10.1145/2723372.2749431. Dostupné z: https://dl.acm.org/doi/10.1145/2723372.2749431

[10] Yakout, M.; Berti-Équille, L.; Elmagarmid, A. K.: Don't be SCAREd. *Proceedings of the 2013 international conference on Management of data - SIGMOD '13*, 2013: s. 553–, doi:10.1145/2463676.2463706. Dostupné z: http://dl.acm.org/citation.cfm?doid=2463676.2463706

[11] Mayfield, C.; Neville, J.; Prabhakar, S.: ERACER. *Proceedings of the 2010 international conference on Management of data - SIGMOD '10*, 2010: s. 75–, doi:10.1145/1807167.1807178. Dostupné z: http://portal.acm.org/citation.cfm?doid=1807167.1807178

[12] doc. RNDr. Jiří Zháněl; Hellebrandt, D. P. V.; Sebera, P. M.: *Metodologie výzkumné práce*. Brno, první vydání, 2014, ISBN 978-80-210-6857-5.

[13] Kim, E. J.; Brunner, R. J.: Teaching Data Science. ročník 2016, 2016: str. 10, doi:10.1016/j.procs.2016.05.513.

[14] Schröer, C.; Kruse, F.; Gómez, J. M.: A Systematic Literature Review on Applying CRISP-DM Process Model. *Procedia Computer Science*, ročník 181, 2021: s. 526–534, ISSN 18770509, doi:10.1016/j.procs.2021.01.199. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1877050921002416

[15] CRISP-DM 1.0. 2000. Dostupné z: https://www.the-modeling-agency.com/crisp-dm.pdf

[16] Palacios, H. J. G.; Toledo, R. A. J.; Pantoja, G. A. H.; aj.: A comparative between CRISP-DM and SEMMA through the construction of a MODIS repository for studies of land use and cover change. *Advances in Science, Technology and Engineering Systems Journal*, ročník 2, č. 3, 2017: s. 598–604, ISSN 24156698, doi:10.25046/aj020376. Dostupné z: http://astesj.com/v02/i03/p76/

[17] Introduction to SEMMA. Dostupné z: https://documentation.sas.com/doc/en/emref/14.3/n061bzurmej4j3n1jnj8bbjjm1a2.htm

[18] Cai, J.; Luo, J.; Wang, S.; aj.: Feature selection in machine learning. *Neurocomputing*, ročník 300, 2018: s. 70–79, ISSN 09252312, doi:10.1016/j.neucom.2017.11.077. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0925231218302911

[19] Peng, M. H.; Long, M. F.; Ding, C.: Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 27, č. 8, 2005: s. 1226–1238, ISSN 0162-8828, doi:10.1109/TPAMI.2005.159. Dostupné z: http://ieeexplore.ieee.org/document/1453511/

[20] Law, M.; Figueiredo, M.; Jain, A.: Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, ročník 26, č. 9, 2004: s. 1154–1166, ISSN 0162-8828, doi:10.1109/TPAMI.2004.71. Dostupné z: http://ieeexplore.ieee.org/document/1316850/

[21] Subasi, A.: *Practical Machine Learning for Data Analysis Using Python*. Academic Press, první vydání, 2020, ISBN 978-0-12-821379-7.

[22] Turner, C.; Joseph, A.; Aksu, M.; aj.: The Wavelet and Fourier Transforms in Feature Extraction for Text-Dependent, Filterbank-Based Speaker Recognition. *Procedia Computer Science*, ročník 6, 2011: s. 124–129, ISSN 18770509, doi:10.1016/j.procs.2011.08.024. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1877050911004893

[23] Lin, S.; Wu, X.; Martinez, G.; aj.: Filling Missing Values on Wearable-Sensory Time Series Data. *Proceedings of the 2020 SIAM International Conference on Data Mining*, 2020-01-26: s. 46–54, doi:10.1137/1.9781611976236.6. Dostupné z: https://epubs.siam.org/doi/10.1137/1.9781611976236.6

[24] Pianykh, O. S.: *Digital imaging and communications in medicine (DICOM)*. Berlin: Springer, second edition vydání, [2012], ISBN 978-3-642-10849-5.

[25] Hierarchical Data Formats - What is HDF5? Dostupné z: https://www.neonscience.org/resources/learning-hub/tutorials/about-hdf5

[26] Folk, M.; Heber, G.; Koziol, Q.; aj.: An overview of the HDF5 technology suite and its applications. *Proceedings of the EDBT/ICDT 2011 Workshop on Array Databases - AD '11*, 2011: s. 36–47, doi:10.1145/1966895.1966900. Dostupné z: http://portal.acm.org/citation.cfm?doid=1966895.1966900

[27] Sarin; Arvind: Comparing HDF5 and NetCDF for scientific data conversion and storage. 06 2020.

[28] Shah, T.; Patel, S.: A Novel Approach for Specifying Functional and Non-functional Requirements Using RDS (Requirement Description Schema). *Procedia Computer Science*, ročník 79, 2016: s. 852–860, ISSN 18770509, doi:10.1016/j.procs.2016.03.083. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1877050916002143

[29] Mairiza, D.; Zowghi, D.; Nurmuliani, N.: An investigation into the notion of non-functional requirements. *Proceedings of the 2010 ACM Symposium on Applied Computing - SAC '10*, 2010: s. 311–, doi:10.1145/1774088.1774153. Dostupné z: http://portal.acm.org/citation.cfm?doid=1774088.1774153

[30] El-Sappagh, S. H. A.; Hendawi, A. M. A.; Bastawissy, A. H. E.: A proposed model for data warehouse ETL processes. *Journal of King Saud University - Computer and Information Sciences*, ročník 23, č. 2, 2011: s. 91–104, ISSN 13191578, doi:10.1016/j.jksuci.2011.05.005. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S131915781100019X

[31] Babu, P.; Stoica, P.: Spectral analysis of nonuniformly sampled data – a review. *Digital Signal Processing*, ročník 20, č. 2, 2010: s. 359–378, ISSN 10512004, doi:10.1016/j.dsp.2009.06.019. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1051200409001298

[32] Caruso, C.; Quarta, F.: Interpolation methods comparison. ročník 35, č. 12, 1998: s. 109–126, ISSN 08981221, doi:10.1016/S0898-1221(98)00101-1. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0898122198001011

[33] Boulanger, J.-L.: *Certifiable Software Applications 2*. ISTE Press - Elsevier, první vydání, 2017, ISBN 9780081011645.

[34] Sedgewick, R.: *Algorithms in C++: Graph Algorithms*. Pearson Education, třetí vydání, 2002, ISBN 978-0-201-36118-6.

[35] Kalra, N.; Bhatt, P.: Parallel algorithms for tree traversals. *Parallel Computing*, ročník 2, č. 2, 1985: s. 163–171, ISSN 01678191, doi:10.1016/0167-8191(85)90026-2. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/0167819185900262

[36] Engström, J.; Markkula, G.; Victor, T.; aj.: Effects of Cognitive Load on Driving Performance. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, ročník 59, č. 5, 2017-05-26: s. 734–764, ISSN 0018-7208, doi:10.1177/0018720817690639. Dostupné z: http://journals.sagepub.com/doi/10.1177/0018720817690639

[37] Haapalainen, E.; Kim, S.; Forlizzi, J. F.; aj.: Psycho-physiological measures for assessing cognitive load. *Proceedings of the 12th ACM international conference on Ubiquitous computing*, 2010-09-26: s. 301–310, doi:10.1145/1864349.1864395. Dostupné z: https://dl.acm.org/doi/10.1145/1864349.1864395

Obsah přiloženého CD

- klíčová slova v ČJ a v AJ.
- abstrakt práce v ČJ.
- abstrakt práce v AJ.
- naskenované zadání diplomové práce.
- kompletní diplomová práce.
- složka se zdrojovým kódem frameworku.
- skript v jazyce Python, který se použil na testování frameworku.

Seznam použitých symbolů a zkratek

DB ..... Databáze
HDF ..... Hierarchický datový formát
DICOM ..... digital imaging and communications in medicine
SW ..... Software
TDMS ..... Technical Data Management Streaming
EKG ..... Electrokardiogram

Seznam tabulek

Seznam obrázků