ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Diplomová práce

Zpracování dat z vláknově-optických senzorů průhybu

Bc. Matěj Halouska

Květen 2018
Vedoucí práce: Ing. Radek Mařík, CSc.

Poděkování / Prohlášení

Na tomto místě bych rád poděkoval svému vedoucímu práce Ing. Radkovi Maříkovi, CSc. za vedení práce, velmi přínosné rady a podporu při práci a celé firmě Safibra s.r.o. za vstřícnost a příjemné prostředí.

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Abstrakt / Abstract

Společnost Safibra s.r.o zadala požadavek na analýzu a implementaci modulu do software SigProc, který zobrazuje data z vláknově-optických senzorů průhybu. Diplomová práce se zabývá problematikou vizualizace streamovaných dat a implementací modulu, který umožní přehledné zobrazení dat s možností interakce uživatelem.

Safibra company Ltd. commissioned a request for analysis and implementation of a new module into SigProc software, which displays data from fiber optic sensors. This diploma thesis consists of research of visualization of data streaming problems and of the implementation of the module, allowing a clear view of the data with a possibility of interaction by the user.

Obsah /

Tabulky / Obrázky

Kapitola 1
Úvod

Firma Safibra s.r.o. [1] je soukromou inženýrskou společností, která se zabývá vývojem optovláknových technologií a senzorů. Společnost má vlastní výzkumné a vývojové oddělení a účastní se mnoha výzkumných projektů. Jeden z projektů, na kterém se společnost podílí, je měření průhybu vlakových kolejí.

Technologické pokroky umožňují stále rychlejší jízdu po kolejích, což ale způsobuje také rychlejší opotřebování kolejí a vytloukání jejich základů. Jednou z nejdůležitějších prevencí proti katastrofálním nehodám, které mohou rozbité nebo uvolněné koleje způsobit, je kontinuální měření jejich průhybu. K tomuto účelu slouží vláknově-optické senzory, které jsou umístěny na nosnících upevněných na kolejích. Tyto senzory průběžně měří průhyb a posílají výsledná data na zpracování do počítačového software.

Společnost Safibra vlastní program SigProc 4.1, který slouží ke zpracování signálů z vláknově optických senzorů. Data jsou v programu analyzovány a pokud se najde nějaký problém na kolejišti, firma například upozorní, že je potřeba, aby na kolejiště přijel podbíjecí stroj a vadnou kolejnici zpevnil, zvednul či dorovnal a tím se předejde potencionálnímu neštěstí.

Současná verze software SigProc umožňovala zobrazování dat v reálném čase, která přímo přicházejí ze senzorů. Nebylo ale možné zobrazit historická data z databáze. Použití software bylo tedy značně omezené a společnost Safibra proto zadala požadavek na vývoj nového modulu do systémy SigProc, který bude umožňovat zobrazení dat, která jsou uložena v databázi. Další požadavek byl, aby nový modul umožňoval úpravy zobrazení dat uživatelem jako je například filtrování, přibližování, oddalování nebo posouvání pro bližší analýzu detailů signálu.

Jelikož jsem studuji obor Interakce člověka s počítačem, který se zaměřuje z velké části na vývoj uživatelského rozhraní, zaměřil jsem se také na uživatelskou přívětivost a jednoduchost systému. Součástí práce je také rešerše technik vizualizace streamovaných dat a vybrání vhodné technologie.

Při vytváření rešerše jsem pracoval převážně s materiály v anglickém jazyce. Snažil jsem se přeložit vše, co bylo možné, ale některé technické názvy nemají český ekvivalent nebo ho mají, ale nikdo to pod českým názvem nezná, proto jsem se rozhodl některé výrazy nepřekládat. Zde je jejich seznam:

- clutter – neboli „nepořádek“ v grafu. Clutter znamená, že v grafu je tolik dat, že je velice špatně čitelný pro koncového uživatele.
- force-directed graph – neboli graf, který zobrazuje uzly tzv. technikou „odpudivé síly“,
- boxplot – neboli „krabicový graf“,
- treemap – neboli technika „stromové mapy“,
- scatterplot – neboli technika „rozptýleného grafu“,
- horizon graph – neboli „graf horizontu“,
- bendline – nosník senzorů,
- scroll – posun v horizontálním směru doprava či doleva,
- pan – posun ve vertikálním směru nahoru či dolů,

- zoom – přiblížení či oddálení,
- usability test – test použitelnosti,
- model–view–controller – softwarová architektura, která rozděluje aplikaci do tří různých nezávislých komponent,
- Cross-site scripting a Cross-site request forgery – techniky průniku se do aplikace, kterým potřeba předcházet při řešení bezpečnosti.

Diplomová práce je rozdělena na 6 hlavních kapitol.

V kapitole Matematický aparát 2 je popsána teorie optických vláken, braggovských mřížek a nosníků senzorů. Dále je zde vysvětlen algoritmus pro přepočet dat ze senzorů pro uložení do databáze v systému SigProc.

V kapitole Rešerše 3 je vytvořena rešerše technik pro vizualizaci streamovaných dat, problémů a nejčastějších chyb, které se při zobrazování streamovaných dat dělají.

V kapitole Analýza 4 je vytvořena kompletní analýza, potřebná pro vývoj nového modulu do aplikace SigProc.

V kapitole Implementace 5 je podrobně popsána výsledná implementace nového modulu.

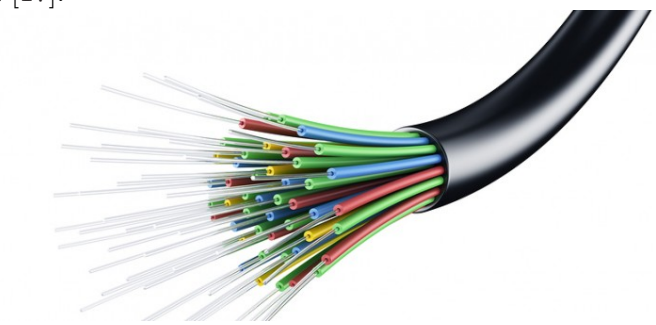
V kapitole Testování 6 je zdokumentováno průběžné i finální testování naimplementovaného modulu.

Kapitola 2
Matematický aparát

Cílem této kapitoly je popsat matematický aparát pro výpočet průhybové křivky senzorů na základě relativního prodloužení vláknových braggovských mřížek.

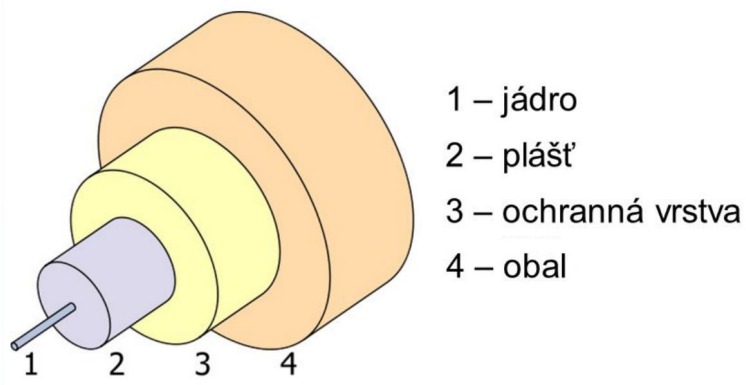
2.1 Optické vlákno

Optické vlákno je plastové nebo skleněné vlákno, které přenáší signály prostřednictvím světla. Jde o válečkový dielektrický vlnovod, kterým se šíří elektromagnetické vlny ve směru podélné osy s využitím totálního odrazu na rozhraní dvou prostředí s rozdílným indexem lomu [17].



Obrázek 2.1. Optické vlákno

Optické vlákno se skládá ze čtyř vrstev 2.2. Nejvnitřnější část se nazývá jádro, kolem něj je plášť a ochranná vrstva a vše je obaleno obalem.



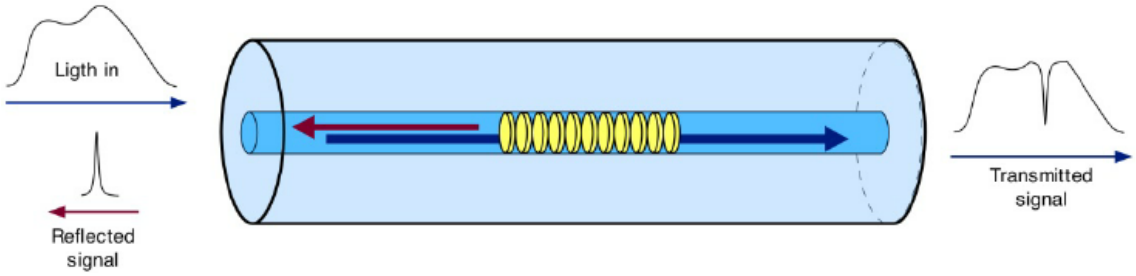
Obrázek 2.2. Průřez optickým vláknem

2.2 Optický senzor

Existují dva druhy optických vláken, jeden typ je určený na přenos dat a druhý jako optický senzor. Optický senzor je optické vlákno, která se vyrábí tak, aby mělo větší citlivost na snímání některé z veličin (tlak, tah apod.).

2.3 Braggova mřížka

Braggova mřížka je struktura vzniklá periodickými nebo kvazi-periodickými změnami indexu lomu v jádře optického vlákna podél jeho osy [23]. Na nosnících jsou umístěné vždy dvě mřížky naproti sobě.



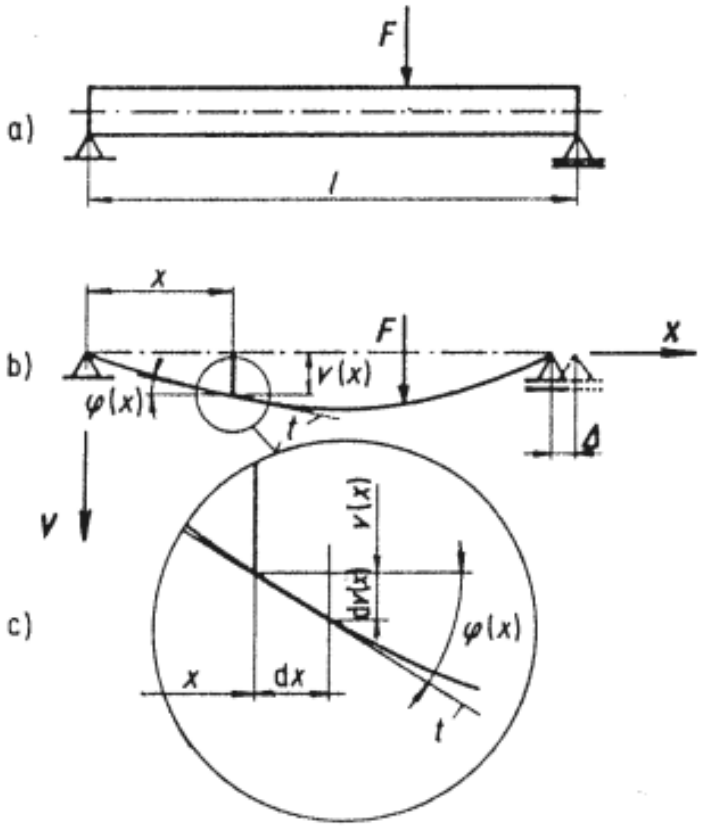
Obrázek 2.3. Braggova mřížka

2.4 Deformace nosníků

Pro popis teorie deformace nosníků a odvození vzorců jsem využil skripta ČVUT Pružnost a pevnost 1 [11].

Deformace nosníku je změna tvaru jeho střednice v důsledku působení vnějších silových účinků. Na obrázku 2.4 b) je naznačena deformovaná střednice nosníku, který je zobrazen na obrázku 2.4 a).

Deformace v libovolném místě nosníku je složena ze dvou složek, **průhyb v(x)** a **úhel natočení střednice nosníku** φ**(x)** (2.4 b),c)).



Obrázek 2.4. Průhyb a úhel natočení střednice nosníku

Nejprve se budeme zabývat deformací přímých nosníků, zatížených příčnými silami působícími kolmo k podélné ose prutu v rovině určené jednou z hlavních os průřezu a podélnou osou. Střednice prutu leží v neutrální rovině a předpokládáme, že při prohnutí prutu se její délka nemění.

Pro směrnici tečny **t** podle obrázku 2.4 c) platí:

$$tg\varphi = \frac{dv(x)}{dx} = v'(x) \tag{1}$$

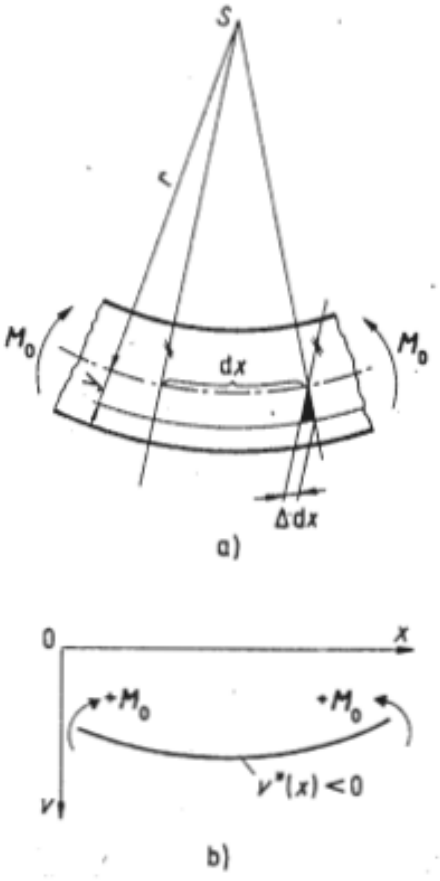
Deformace střednice v(x) a v'(x) uvažujeme velmi malé vůči rozměrům nosníku. Pro úhel natočení pak plyne:

$$tg\varphi = \varphi(x) = v'(x) \tag{2}$$

2.4.1 Diferenciální rovnice průhybové čáry

V kapitole (9) ve skriptech Pružnost a pevnost 1 [11] je odvozen základní vztah pro rozdělení poměrné deformace v průřezu při ohybu ve tvaru:

$$\varepsilon_x = \frac{M_o}{E.J_z}.y \tag{3}$$



Obrázek 2.5. Křivost rovinné křivky

Velice důležitou charakteristikou průhybové křivky nosníku je její křivost K. Z obrázku 2.5 je možné vyjádřit geometrický poměr:

$$\varepsilon_x = \frac{\Delta dx}{dx} = \frac{y}{r} \tag{4}$$

Odkud s použitím rovnice (3) dostaneme pro křivost tento vztah:

$$K = \frac{1}{r} = \frac{\varepsilon_x}{y} = \frac{M_o}{E.J_z} \tag{5}$$

Rovnice (4) je odvozena za předpokladu, že ohybový moment **Mo** je podél prutu konstantní. Platí ale také pro **Mo = Mo(X)**. V průřezu nosníku vznikají ještě smyková napětí od posouvající síly. Vliv těchto smykových napětí na celkovou deformaci nosníku je ale nepatrný. Z rovnice (5) odvodíme diferenciální rovnici průhybové čáry dosazením známého vztahu z analytické geometrie, který vyjadřuje křivost K rovinné křivky v(x) 2.5:

$$K = \frac{1}{r} = \pm\frac{v''(x)}{[1 + (v'(x))^2]^{(3/2)}} \tag{6}$$

Prohnutí uvedenému na obrázku 2.5 b), vyvolanému kladnými ohybovými momenty, odpovídá v rovnici (6) záporné znaménko. Dále použitím rovnice (5), získáme zjednodušenou diferenciální rovnici průhybové čáry pro malé deformace nosníku:

$$\frac{1}{r} = -v''(x) = \frac{M_o(x)}{E.J_z(x)} \tag{7}$$

Při řešení úlohy stačí stanovit pro rovnici (7) funkce Mo= Mo(X) a Jz = Jz(X) pro pravou stranu výrazu. Po prvním integrování získáme rovnici úhlu natočení střednice v'(x) = ε(x) a po druhém integrování rovnici průhybové čáry v(x) v závislosti na odlehlosti X.

$$v''(x) = \frac{M_o(x)}{E.J_z(x)} \tag{8}$$

Diferenciální rovnici elastické průhybové čáry (8) definoval v roce 1694 švýcarský matematik J. Bernoulli. Její aplikaci rozšířil později také Leonard Euler. Je-li zadána funkce spojitého zatížení, je někdy vhodné vycházet ze Schwedlerovy věty:

$$\frac{dM_o(x)}{dx} = T(x)a\frac{dT(x)}{dx} = -q(x) \tag{9}$$

Dvojitou derivací rovnice (8) plyne tento vztah:

$$\frac{d^2}{dx^2}[E.J_z.v''(x)] = \frac{d^2M_o(x)}{dx^2} = q(x) \tag{10}$$

Čtyřnásobným integrováním rovnice (10) určíme vektor průhybové čáry v(x). Tento vektor se pak použije jako vstupní vektor do programu SigProc, kde se pomocí matematického aparátu vypočtou hodnoty, které se uloží do databáze (viz. 2.5).

2.5 Výpočet v programu SigProc

Program SigProc vezme na vstup vektor průhybové čáry v(x), který byl vypočítán v předchozí kapitole 2.4. Z tohoto vektoru systém vypočte data, která se uloží do databáze. Výpočet se skládá z těchto částí:

1. Nejdříve se kontroluje, jestli souřadnice senzorů, které jsou nastavené v parametrech, odpovídají počtu položek vstupního vektoru. Pokud ne, nastaví se pro dané

X souřadnice stejná hodnota na Y souřadnice, čímž vznikne úhlopříčná čára, která indikuje, že je něco v nepořádku.

2. Ve druhé fázi se udělá se kalibrace nuly. Jednotlivé mřížky na začátku dostanou určitou hodnotu. Tato hodnota by měla být ideálně nulová, což značí, že je nosník rovně. Vstupní vektor se při kalibraci uloží do souboru a potom se od tohoto vektoru odčítá, dokud se neuloží vektor nový, čímž se definuje nulová poloha – nosník je rovně.
3. V další fázi se eliminují se data, která jsou označena jako neplatná. Například když se mřížka utrhne. V tomto případě se celá dvojice mřížek vyřadí způsobem, že se hodnoty z nich nahradí nulou, což značí, že v daném místě je nosník rovný.
4. Poté se oddělí ze vstupního vektoru vrchní a spodní část mřížek, aby byly pod sebou. Aby spodní části a horní části mřížek, které patří k sobě, byly ve vektoru spodních a vektoru horních na stejném indexu.
5. Dále se na začátek a na konec vloží virtuální nula. Nosník je rovný.
6. Vypočítá se rozdíl mezi hodnot horní a dolní mřížkou.
7. Jednotky mikrometry na metry se převedou na metry na metry.
8. Spočítá se zakřivení vydělením výškou nosníku (vzdáleností mezi spodní a horní mřížkou).
9. Potom se proloží vstupní vektor pomocí křivostí, dvakrát se zintegruje dle vztahu (10) a spočítá se referenční konstanta, která zajistí otočení profilu podle prvního bodu, aby výsledný profil byl rovně.
10. Na závěr se dosadí do výsledného spočítaného vektoru X souřadnice podle toho, jak je žádáno, aby vypadal vektor na výstupu. V parametrech se dá nastavit, jak moc má být výsledný profil jemný.
11. Na úplný závěr se ještě převede na jednotky, které jsou nastavené v konfiguraci.

Pro účely nového modelu by bylo ale zbytečně komplikované zobrazovat celý vektor průhybu. Pro reálné využití jsou podstatné stejně jenom maximální výchylky. Proto do pro ukládání do databáze vezmeme nejnižší a nejvyšší hodnotu průhybu a vypočítáme středovou hodnotu, kterou pak zobrazujeme v grafu. Pro praxi jsou zajímavé hlavně trendy vývoje extrémů v čase, což tato střední hodnota zobrazuje.

Minima a maxima zobrazují dynamiku čili například při průjezdu vlaku zobrazí větší výchylky od normálu. Střední hodnota zobrazí, jak se nosník chová kontinuálně a nezohlední tolik tyto dynamické události.

Kapitola 3
Rešerše – techniky vizualizace streamovaných dat

3.1 Úvod

Vizualizace streamovaných dat má v dnešní době obrovských dat důležitou roli. Streamovaná data jsou vytvářena všude – od osobních streamů, kterými lidé sdílejí své zážitky z cestování, až po rozsáhlé sítě finančních transakcí či sociálních médií. Navíc se tento trend bude v nejbližší době stále rozvíjet. Velké úsilí je vkládáno do výzkumu, jak efektivně přenášet, uchovávat a zobrazovat obrovské množství dat. Metodám, jak správně vizualizovat, prozkoumat a pochopit všechny tyto obrovské sady dat, se dává čím dál větší význam.

Vizualizace streamovaných dat je silně spojena s časovým kontextem a velmi často se používají metody, které mapují čas na horizontální osu. V této rešerši jsem se zaměřil na problémy, se kterými se nejvíce potýkáme při vizualizaci streamovaných dat, jaké chyby se při vizualizaci dělají a jaké nejčastější metody se používají.

3.2 Problémy vizualizace streamovaných dat

V této kapitole se budeme věnovat nejčastějším problémům, které se objevují při vizualizaci streamovaných dat. Čerpal jsem především ze zdrojů:

- 4 Potential Problems With Data Visualization [5],
- Top 10 unsolved information visualization problems [9],
- Top scientific visualization research problems [14],
- Visualization of streaming data [16],
- několik dalších zdrojů.

3.2.1 Příliš velké zjednodušení dat

Jedním z největších problémů velkých dat je jejich špatná čitelnost a jedním z nejdůležitějších úkolů vizualizace je proto zjednodušení dat, aby byla dobře čitelná. Je ale potřeba najít dobrý kompromis. Často se stává, že technika data zjednoduší natolik, že se ztratí důležité informace a vizualizace je proto nepoužitelná či zavádějící.

3.2.2 Vizualizační techniky nevysvětlují

Častým problémem je, že vizualizační technika data zobrazí, ale dostatečně nevysvětlí. Uživatel, který data čte, musí být expert v oboru, aby zobrazená data vůbec pochopil. Proces analýzy dat se za posledních 30 let téměř nezměnil – analytici se podívají na data a napíšou reporty. Tento proces je velmi zdlouhavý a finančně náročný.

3.2.3 Zpracování nových dat

Jak by se měla vizualizace změnit, když přidáme nová data? Musí být celé uspořádání přepočítáno, když přidáme pouze jeden prvek, jako například v tzv. force-directed grafech, nebo lze prvek snadno přidat jako ve scatterplotu?

3.2.4 Správné měřítko grafu

Často se stává, že nově příchozí data nespadají mezi současné rozpětí minima a maxima v grafu. Je potřeba tedy vždy měřítko upravit podle nových dat. Největší problém nastává v případě extrémních hodnot. Je lepší celé měřítko posunout kvůli jedné extrémní hodnotě, což způsobí špatnou čitelnost ostatních dat? Nebo měřítko zachovat a extrémní hodnotu zobrazit nějakým jiným způsobem?

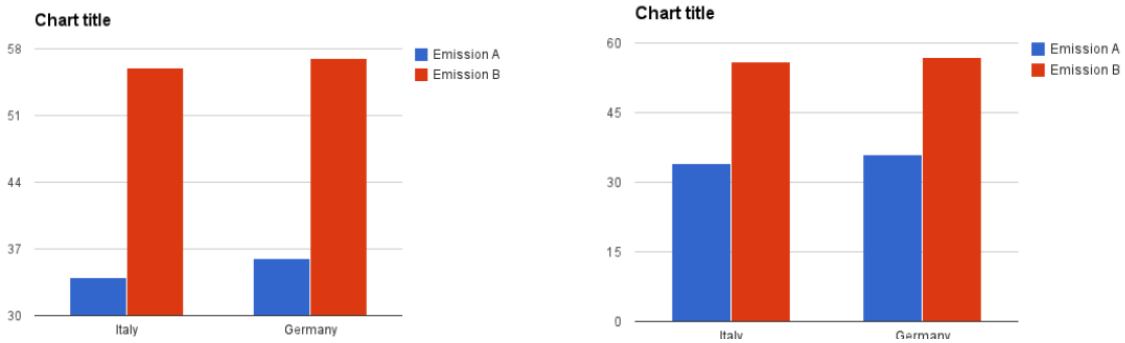
3.3 Časté chyby při vizualizaci dat

V této kapitole se budeme zabývat nejčastějším chybám, které se objevují při vizualizaci dat. Čerpal jsem především z těchto zdrojů:

- 77 most common data visualization mistakes [20],
- Chart dos and don'ts [2],
- Visualization of streaming data [16],
- několik dalších zdrojů.

3.3.1 Neúplná škála

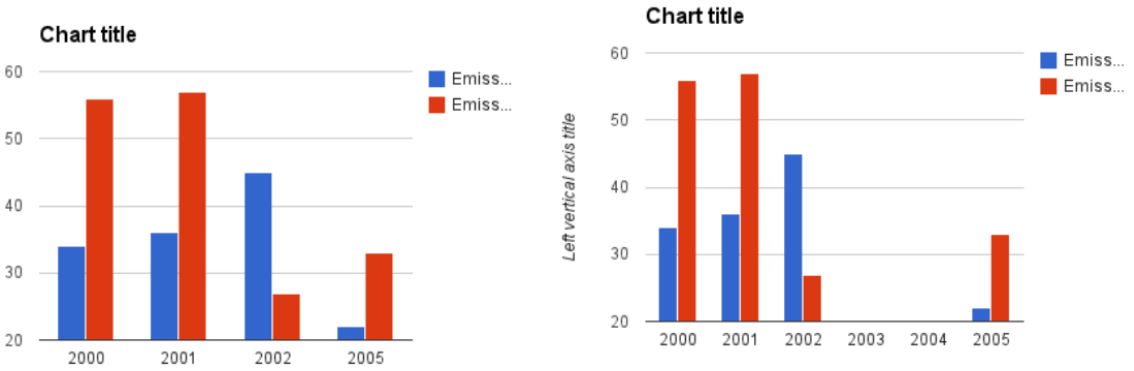
Velice častou chybou vizualizace dat je použití neúplné škály. Graf pak přináší zavádějící dojem. Podle prvního obrázku 3.1 by se mohlo zdát, že množství emisí B je zhruba 4x tolik než emisí A, přitom reálně je podle druhého obrázku patrné, že rozdíl je zhruba pouze o polovinu.



Obrázek 3.1. Použití neúplné vs. úplné škály

3.3.2 Nekonzistentní intervaly

Použití nekonzistentních intervalů, například vynechání některých zdánlivě nedůležitých roků, může vést k zmatení koncového uživatele. Podle prvního obrázku 3.2 by se mohlo zdát, že za poslední rok nastala největší změna. Přitom se nejedná o změnu za poslední rok, ale za poslední 3 roky, jak je patrné podle druhého obrázku.



Obrázek 3.2. Použití neúplného vs. úplného intervalu

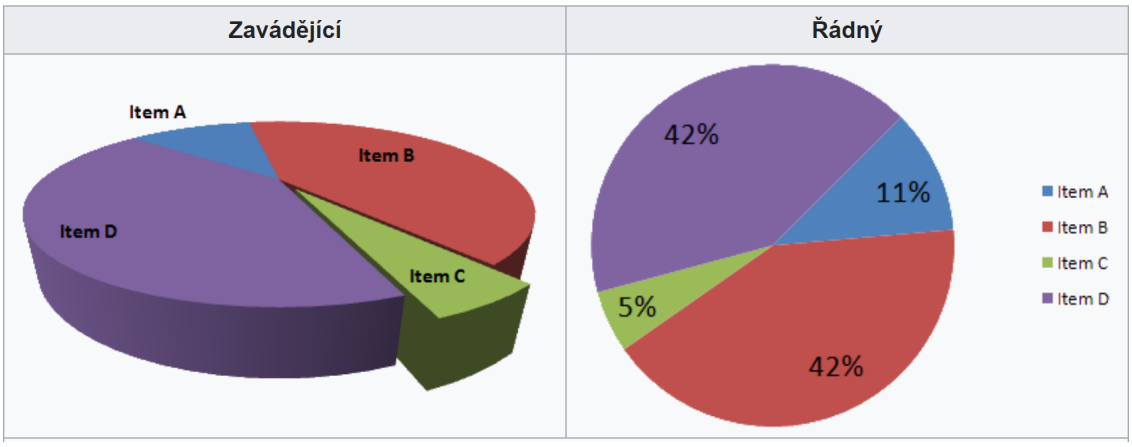
3.3.3 Nesprávný poměr měřítek os

Další častou chybou je použití špatného poměru měřítek obou os. Křivky pak mají příliš strmý či příliš pozvolný sklon, což může vést k zmatení cílového uživatele. Viz. obrázek 3.3.

Obrázek 3.3. Použití různých měřítek os

3.3.4 Nesprávné použití 3 D grafu

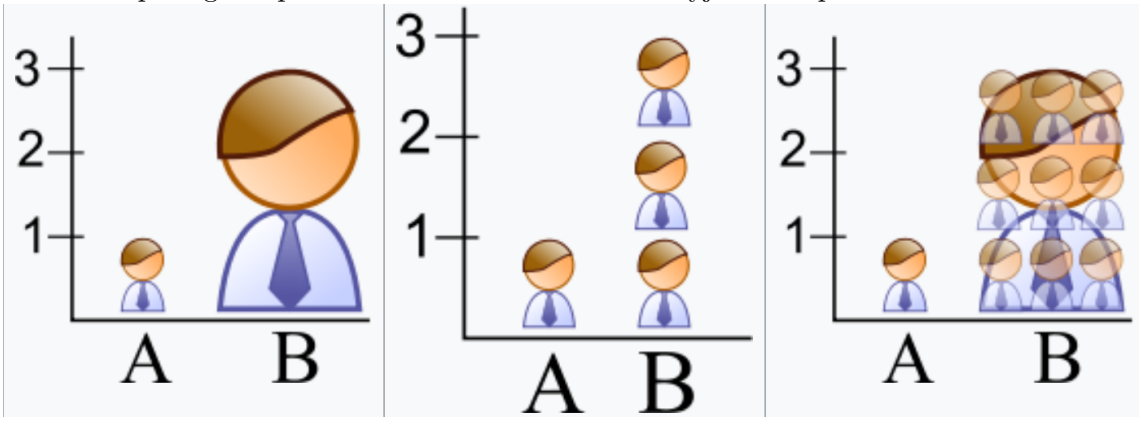
Nevhodné použití 3 D verze grafu může vést k zavádějící vizualizaci. Například na prvním obrázku 3.3 se zdá položka C větší než položka A, přitom v reálu je podle druhého obrázku jasné, že položka A je více než 2x větší než položka C. Viz. obrázek 3.4.



Obrázek 3.4. Zavádějící 3 D graf

3.3.5 Nevhodná změna měřítka 2 D piktogramu

Další chybou může být nevhodná změna měřítka, kdy pouze poměrově zvětšíme 2 D piktogram. Na třetím obrázku 3.5 je patrné, že zvětšený piktogram zabírá plochu 9x větší než piktogram původní. Ve vizualizaci má ale vyjadřovat pouze 3x větší hodnotu.



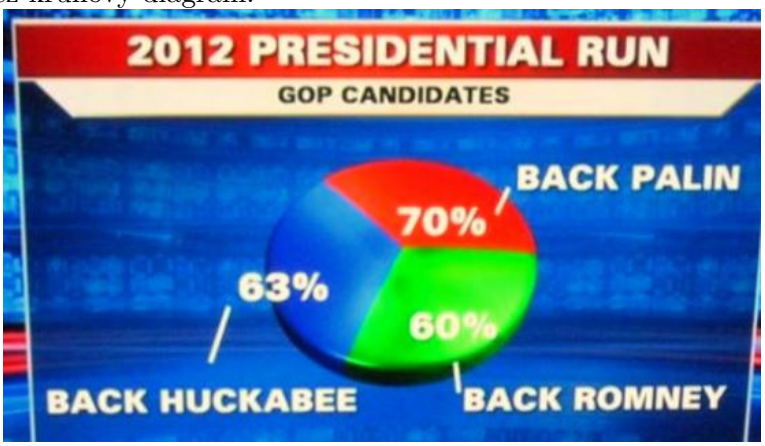
Obrázek 3.5. Nevhodná změna měřítka

3.3.6 Nedostatečně rozdílný kontrast barev

Zvolení dvou kontrastně podobných barev pro zobrazení dvou rozdílných parametrů může vést k mylnému zaměnění cílovým uživatelem. Problém může nastat například při promítání projektorem, zobrazení na starším monitoru s horším rozsahem kontrastu nebo v případě, že má cílový uživatel zrakovou vadu (např. barvoslepost).

3.3.7 Součet procent nesedí

Při zobrazení dat pomocí kruhového diagramu musí součet procent dat dávat dohromady 100. Na obrázku 3.6 je součet 193 procent. Je pravděpodobné, že výzkum dovoloval více možných odpovědí. V tomto případě je tedy nutné vybrat jinou techniku vizualizace než kruhový diagram.



Obrázek 3.6. Součet procent nesedí

3.3.8 Nekompletní data

Je možné stanovit z obrázku 3.7, která firma ovládá větší část trhu? Je jasné, že modrá firma ovládá trh ve více státech USA. Nicméně už není jasné, jak velká je kupní síla v

jednotlivých státech, a proto z tohoto obrázku není možné stanovit, která firma ovládá větší část trhu.

Obrázek 3.7. Nekompletní data

3.4 Typy dat

3.4.1 Spojitá data

Spojitá data jsou data, která obecně nemají předem stanovený počet hodnot. Hodnot může být neomezené množství. Řadí se mezi ně veškeré měřitelné hodnoty jako například výška, váha, věk apod.

3.4.2 Kategorická data

Kategorická data jsou data, která mají předem jasně určené kategorie. Patří mezi ně například pohlaví, rasa, vzdělání, zaměstnání nebo rodinný stav.

3.5 Techniky vizualizace dat

Při rozboru vizualizačních technik jsem se zaměřil na dva důležité faktory, které ovlivňuje příchod nových dat:

- Jaké vizuální proměnné se ve vizualizaci mění.
- Jestli a kde může docházet ke ztrátě kontextu.

Čerpal jsem především z těchto zdrojů:

- Visualization of streaming data [16],
- Streaming a komunikace [12],
- Large-scale data visualization using parallel data streaming [4]
- Stacked graphs – geometry & aesthetics [8],
- Methods for presenting statistical information: The box plot. [18],
- The box-plot: an exploratory analysis graph for biomedical publications [22],
- The Development of the Horizon Graph [19],
- Ordered treemap layouts [21],
- Designing pixel-oriented visualization techniques: Theory and applications [15],

- A New Pixel-Oriented Visualization Technique Through Color Image [7],
- Visual data mining with pixel-oriented visualization techniques [6],
- Context Preserving Dynamic Word Cloud Visualization [10],
- Word cloud explorer: Text analytics based on word clouds [13],
- několik dalších zdrojů.

3.5.1 Streamovací graf

Streamovací graf je někdy také nazýván tematická řeka. Jde o techniku zobrazení několika toků proměnných, které mění své hodnoty v průběhu času a jsou navrstveny symetricky podél časové osy.

Streamovací graf se používá na spojitá data.

Optimální uspořádání jednotlivých proudů je velice důležité pro dobrou čitelnost vizualizace. Aby dobré čitelnosti dosáhlo, vizuální objekty (vrstvy) s nejmenšími změnami velikostí jsou umístěny ve středu, zatímco vrstvy s největšími změnami jsou na horní a spodní straně grafu.

V případě příchozích dat, která nespadají do současného rozsahu, je potřeba tento rozsah grafu upravit. Naštěstí tyto změny rozsahu představují pouze malé, téměř zanedbatelné ztráty kontextu. Je potřeba ale pokaždé upravit uspořádání jednotlivých proudů, což s sebou nese větší výpočetní náročnost a také může způsobit zmatení uživatele.

Streamovací graf má hodně nežádoucích vlastností a vážných omezení čili je použitelný pouze pokud jsou splněna tato kritéria [16]:

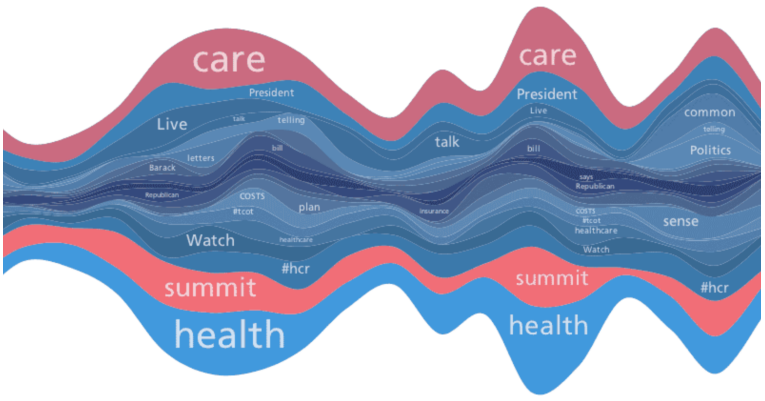
- Náhlé změny v každé vrstvě nejsou příliš extrémní a příliš časté.
- Počet vrstev je malý.
- Historická data jsou vyřazena nebo aproximována.
- Doba mezi dvěma zobrazeními různých aktualizací uživatelem je relativně dlouhá.
- Je dostatek času na informování uživatele o novém přeskupení (například pomocí animace).

Změna vizuálních proměnných:

- rozsahy os,
- změna pořadí datových toků,
- přidání, odebrání toků z obrazovky.

Ztráta kontextu:

- Velice velká, z důvodu změn pořadí datových toků.



Obrázek 3.8. Streamovací graf

3.5.2 Liniový graf

Liniový graf je jedním z nejjednodušších a nejpoužívanějších typů grafů. Jednotlivá data jsou spojena čarou (linií). Používá se nejčastěji k zobrazení časových trendů v datech.

Nejvhodnější použití je pro spojitá data, viz. obrázek 3.9.

Liniový graf je v hodně parametrech podobný streamovacímu grafu. Je potřeba upravit měřítko grafu pokaždé, když příchozí data nespadají do současného rozmezí. V případě absence či špatně zvoleného měřítka může dojít k zavádějící vizualizaci. Obdobně obrácené měřítko osy může být matoucí. U liniového grafu je také potřeba si dávat pozor na případný grafové přeplnění a clutter, které můžou způsobit špatnou čitelnost.

Mezi výhody liniového grafu patří jednoduchost na vytvoření i čtení, dobré vizuální zobrazení trendů a změn, možnost zobrazení kladných i záporných hodnot či přehledné porovnání vztahů mezi dvěma a více proměnnými.

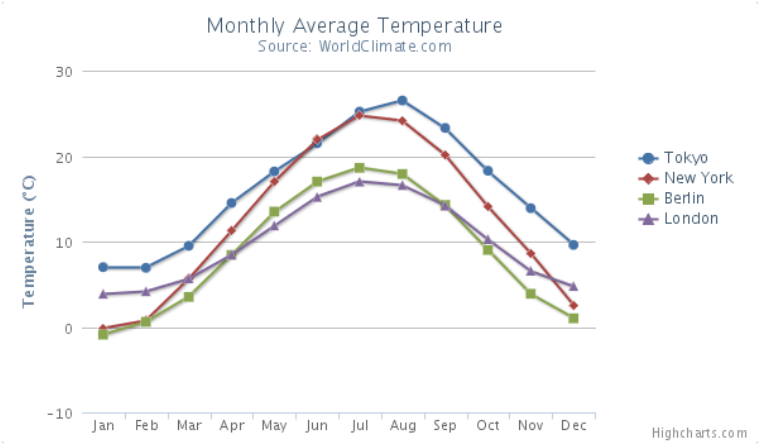
Nevýhodami liniového grafu je špatná čitelnost v případě zobrazení moc velkého množství proměnných a špatná čitelnost v případě moc širokého datového rozmezí.

Změna vizuálních proměnných:

- rozsahy os,
- přidání, odebrání datových linií.

Ztráta kontextu:

- Pouze, pokud jsou data mimo rozsah osy Y.
- Další závažné problémy - grafového přeplnění, clutter, nízké rozlišení.



Obrázek 3.9. Liniový graf

3.5.3 Sloupcový graf

Sloupcový graf je další s často používaných typů grafů. Tento diagram znázorňuje proporciální poměr jednotlivých hodnot pomocí obdélníkových pruhů. Pruhy mohou být zobrazeny vodorovně i svisle.

Nejčastější použití sloupcového grafu je pro kategorická data, viz. obrázek 3.10.

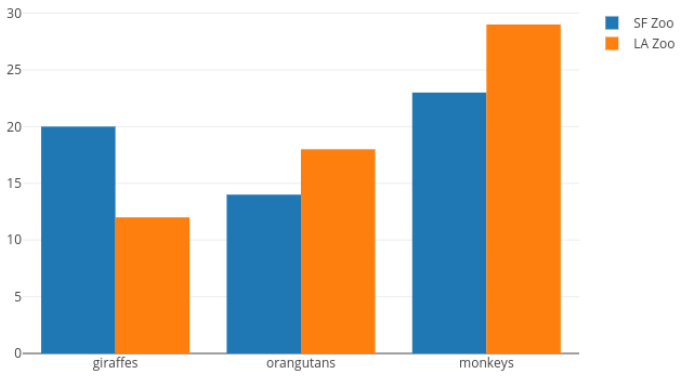
Podobně jako u liniového grafu je potřeba upravit měřítko grafu pokaždé, když příchozí data nespadají do současného rozmezí. V případě absence či špatně zvoleného měřítka může opět dojít k velice zavádějící vizualizaci. Sloupcový graf je proto graf nejvíce zmanipulovatelný.

Změna vizuálních proměnných:

- rozsahy os,
- přidání, odebrání jednotlivých hodnot.

Ztráta kontextu:

- Pouze, pokud jsou data mimo rozsah osy Y v případě svislého zobrazení a osy X v případě vodorovného zobrazení.



Obrázek 3.10. Sloupcový graf

3.5.4 Kruhový diagram

Kruhový diagram je také velice oblíbený graf. Znázorňuje poměr mezi jednotlivými hodnotami pomocí proporcionálního poměru kruhových výsečí.

Kruhový diagram může být použit pouze pro kategorická data, viz. obrázek 3.11.

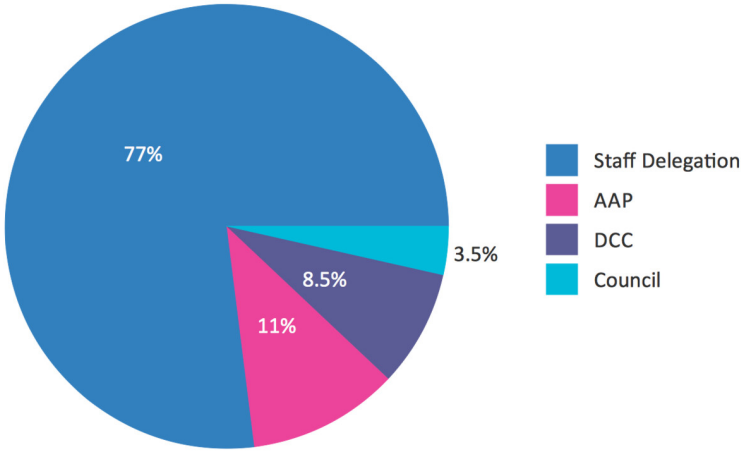
Výhodou je jednoduché zkonstruování a čitelnost. Velkou nevýhodou je možnost použití pouze pro malé množství hodnot. Při větším množství rapidně klesá čitelnost grafu. V případě použití 3 D verze grafu může jednoduše dojít k zavádějící vizualizaci, kdy se bližší hodnota zdá větší než vzdálenější a reálně to může být naopak.

Změna vizuálních proměnných:

- změna barev,
- přidání, odebrání jednotlivých hodnot.

Ztráta kontextu:

- Při změně barev může dojít ke zmatení.
- V případě přidání velkého množství hodnot výrazně klesá čitelnost.



Obrázek 3.11. Kruhový diagram

3.5.5 Boxplot

Boxplot je diagram, který zobrazuje numerická data pomocí jejich kvartilů (tři kvartily rozdělují data na čtvrtiny). Střední "krabicová" část diagramu je zespodu ohraničena 1. kvartilem, shora 3. kvartilem, a mezi nimi je linie, která označuje medián.

Boxplot jde použít pro kategorická i spojitá data, ale pouze pro data numerická, viz. obrázek 3.12.

Velká výhoda boxplotu je, že je jím možné efektivně zobrazit rozsáhlá data. Další výhodou je, že boxplot jako jedna z mála metod zobrazí i odlehlé hodnoty. Nevýhodou je, že při zobrazení rozsáhlých dat, graf zobrazí pouze základní sumář dat a nikoliv přesné konkrétní hodnoty. Proto je nejlepší využít boxplot v kombinaci s jinými metodami vizualizace.

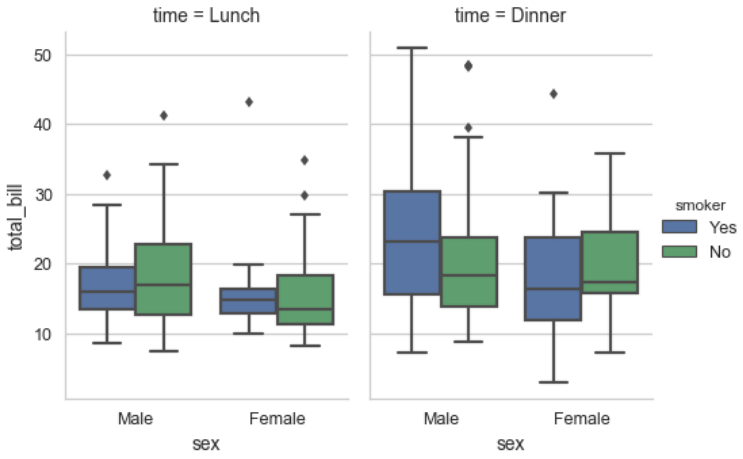
Základní verze boxplotu nezobrazuje hustotu dat podél osy Y. Existují ale modifikované verze, které hustotu dat zobrazují. Jsou dobře popsány v práci, kterou sepsal Kristin Potter [18].

Změna vizuálních proměnných:

- rozsahy os,
- přidání, odebrání datových linií.

Ztráta kontextu:

- Je velice minimální.



Obrázek 3.12. Boxplot

3.5.6 Histogram

Histogram je diagram, který je velice podobný sloupcovému grafu, ale používá se pro spojitá data. Znázorňuje distribuci dat pomocí sloupců stejné šířky, viz. obrázek 3.13. Výška sloupce vyjadřuje četnost zobrazované veličiny v daném intervalu. Nesprávně zvolená šířka intervalu může výrazně snížit informační hodnotu zobrazení.

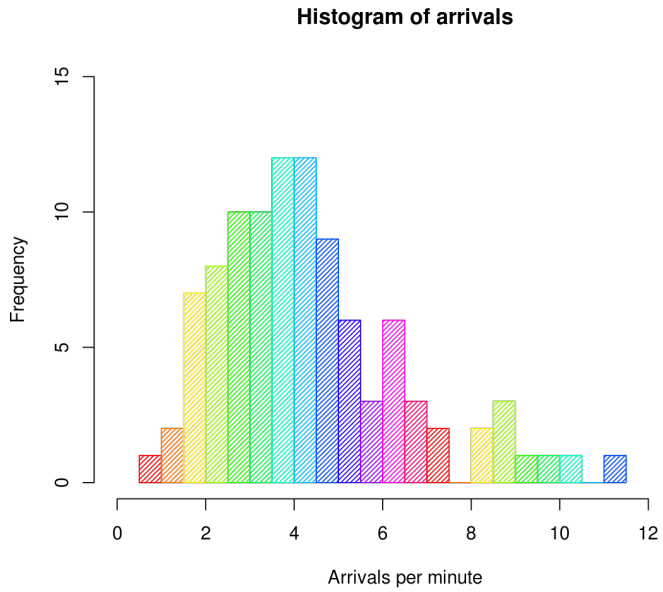
Výhodou histogramu je jeho použitelnost na rozsáhlá data a na data s velkým rozsahem. Nevýhody jsou, že z diagramu je velice těžké poznat konkrétní hodnoty a že histogram není vhodný pro porovnávání více různých kategorií.

Změna vizuálních proměnných:

- rozsahy os,
- změna šířky intervalů.

Ztráta kontextu:

- Je velice minimální.



Obrázek 3.13. Histogram

3.5.7 Treemap

Treemap je technika plnění prostoru daty. Každý uzel je reprezentován obdélníkem, jehož plocha odpovídá jednomu z jeho atributů. Barvou výplně obdélníku lze zobrazit jiný atribut dat. Tato technika může zobrazovat jak hodnoty, tak hierarchii uzlů. Existuje několik různých algoritmů rozložení, jako jsou například [16]:

- Metoda použití pivotu
- Metoda řezu
- Metoda spirály
- Metoda seřazené treemapy

Metoda treemap je použitelná pouze pro kategorická data, viz. obrázek 3.14.

Výhoda metody treemap je dobrá viditelnost malých uzlů. Nevýhodou je okamžitá ztráta kontextu při rychlých změnách dat.

Jde o relativní techniku, ve které vizuální objekty reprezentující položky závisí na ostatních položkách. Metoda treemap může být použita také pro zobrazení nových dat v kontextu s historickými daty, ale není možné zrekonstruovat historii / vývoj dat.

Změna vizuálních proměnných:

- změna atributů obdélníků (velikost, barva, průhlednost),
- posuny obdélníků v hierarchii,
- přidání nebo odebrání obdélníků z obrazovky.

Ztráta kontextu:

- Je významná v případě velkých změn velikostí obdélníků.
- Je významná v případě posunů obdélníků v hierarchii.
- Je potřeba změna barev v případě dat mimo rozsah.

Obrázek 3.14. Treemap

3.5.8 Scatterplot

Scatterplot se obvykle používá k vizualizaci vícerozměrných dat za účelem nalezení korelace mezi dimenzemi a pro detekci klastrů. Dva atributy dat jsou mapovány na kartézské souřadnice, vytvářející bod. Vizuální parametry bodů (velikost, barva, tvar ...) je možné namapovat na další datové atributy.

Scatterplot patří do třídy absolutních vizualizačních technik, kde je každý objekt umístěn ve vizualizaci nezávisle na ostatních objektech.

Scatterplot může být použit na spojitá i kategorická data, viz. obrázek 3.15.

Velkou výhodou této metody je zachování kontextu v případě nově příchozích dat (pokud jsou datové atributy v daném rozmezí). V případě, že atributy v daném rozmezí nejsou, dochází pouze k malé ztrátě kontextu.

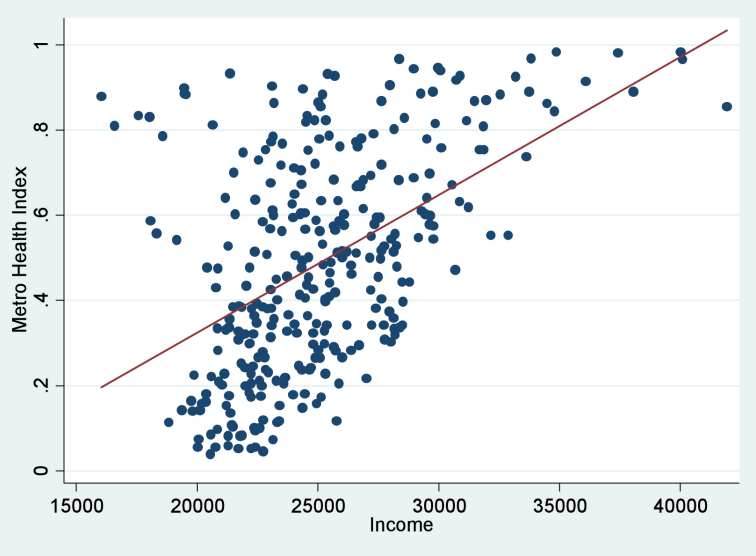
Nevýhodou této techniky je problém grafového přeplnění, kdy je v grafu moc objektů, že se data dají velice špatně číst.

Změna vizuálních proměnných:

- rozsahy os,
- body přidané a odebrané z obrazovky,
- již existující body mohou měnit své parametry (velikost, barvu, průhlednost, pozici).

Ztráta kontextu:

- Nastává pouze, pokud jsou data mimo rozsah.
- Je minimální, kvůli udržení pořadí bodů.
- Další závažné problémy jsou grafové přeplnění a clutter.



Obrázek 3.15. Scatterplot

3.5.9 Horizon graph

Horizon graf je graf tzv. dvou-tónového zbarvení. Velice efektivně využívá prostor a barvy, proto je často využíván pro zobrazování velkého množství časových řad, viz. obrázek 3.16.

Horizon graf se používá pro detekci odlehlých hodnot, extrémních případů či převládajících schémat a pro porovnávání hodnot ve více různých datových řadách.

Tato metoda může být použita na spojitá i kategorická data.

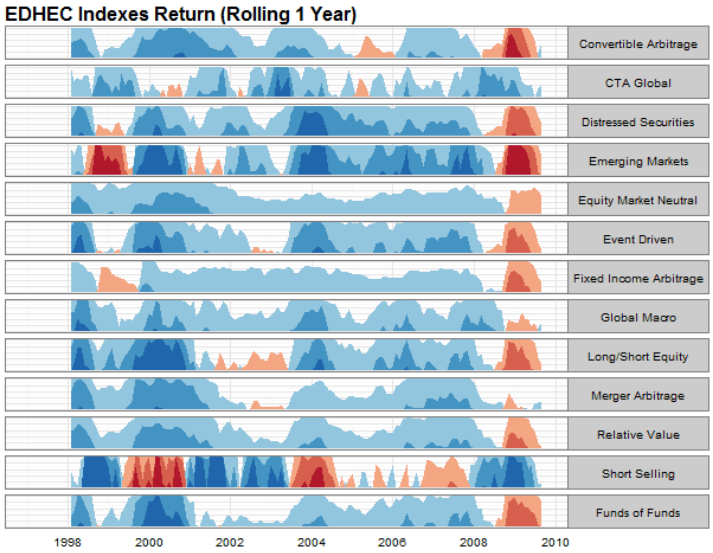
Mezi jeho nevýhody patří ztráta kontextu v případě, že jsou nově přidané hodnoty mimo původní rozsah. Pokud toto nastane, je potřeba opět aplikovat celý proces barevného mapování. Ztráta kontextu může nastat také v případě změn pořadí jednotlivých datových řad [19].

Změna vizuálních proměnných:

- rozsahy os,
- rozsahy barev pro nové maximum a minimum,
- množství jednotlivých grafů,
- změna pořadí jednotlivých grafů.

Ztráta kontextu:

- Je potřeba změna barev v případě dat mimo rozsah.
- Je potřeba změna pořadí jednotlivých grafů.



Obrázek 3.16. Horizon graph

3.5.10 Pixelově orientované vizualizace

Pixelově orientované vizualizace jsou techniky pro vizualizaci vícerozměrných dat. Tyto techniky jsou postavené na přiřazení barev pixelu na jednu konkrétní hodnotu atributu. Existuje několik různých algoritmů na řazení pixelů a nejčastěji používaná je rekurzivní metoda, viz. obrázek 3.17.

Tato technika může být použita pouze na kategorická data.

Nevýhodou je potřeba překreslení všech pixelů v případě nově příchozích dat, která jsou mimo rozsah barvy. Změny pořadí dimenzí mohou také způsobit ztráty kontextu. Množství dat, které je možné zobrazit touto technikou je limitováno počtem pixelů na zobrazují obrazovce.

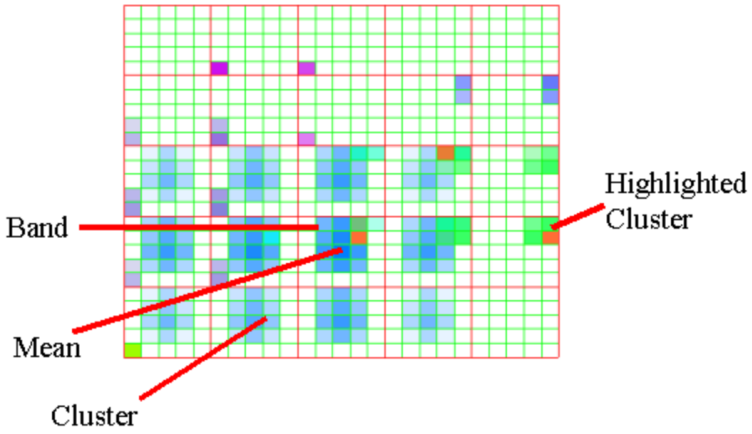
Existují dva hlavní algoritmy, podle kterých se pixely uspořádají: Naturally Ordered Arrangement a Query Dependent Arrangement [7].

Změna vizuálních proměnných:

- pixely přidané a odebrané z obrazovky,
- změny barev,
- přeskupení dimenzí,
- změna rekurzivních stupňů.

Ztráta kontextu:

- Je potřeba změna barev v případě dat mimo rozsah.
- Nastává, pokud je potřeba přeskupení dimenzí.



Obrázek 3.17. Pixelově orientované vizualizace

3.5.11 Word cloud

Word cloud je další populární technika zobrazování dat. Velikost slov a barva značí důležitost konkrétních slov, viz. obrázek 3.18.

Metoda word cloud může být použita pouze na kategorická data.

Používají se dva hlavní typy algoritmů pro řazení slov: force-directed a spirálový. V případě algoritmu force-directed může docházet k velké změně kontextu při jakýchkoliv nově příchozích datech. Je potřeba tedy pokaždé graf přepočítat. V případě spirálového algoritmu dochází ke ztrátě kontextu pouze pokud je změna mezi dvěma iteracemi velká, protože tento algoritmus umisťuje slovo s největší váhou do středu.

Změna vizuálních proměnných:

- velikost a pozice slov,
- přidání nebo odebrání slov.

Ztráta kontextu:

- U spirálového algoritmu nastává jen v případě velké změny.
- U force-directed algoritmu je významná.

Obrázek 3.18. Word cloud

3.6 Použitá technika

Diplomová práce je nový modul do již fungujícího software, který vizualizuje streamovaná data pomocí techniky **liniového grafu**. Po prozkoumání různých možností jsem usoudil, že i tento nový modul bude nejlepší udělat stejnou technikou.

Modul zobrazuje spojitá data, proto by nebylo moc vhodné použít metody sloupcový graf, kruhový diagram, treemap, word cloud nebo pixelově orientovanou metodu, protože všechny tyto techniky se používají převážně na kategorická data. Metody horizon graph a streamovací graf se zase používají spíše pro porovnání více různých časových os, což se v tomto modulu dělat nebude. Scatterplot je výhodné použít v případě vizualizace vícerozměrných dat a pro data zobrazovaná tímto modulem nemá žádnou výhodu oproti liniovému grafu. Metody histogram a boxplot není vhodné použít, protože je z nich velice těžké rozpoznat konkrétní hodnoty, což je v tomto modulu žádoucí.

Největší výhodou metody **liniový graf** pro streamovaná data je minimální ztráta kontextu při příchodu nových dat.

Zobrazení dat jinou technikou určitě je ale také možné, proto je důležitá rozšiřitelnost a modifikovatelnost 4.3.4 nového modulu, aby bylo možné v budoucnu v případě potřeby naimplementovat i zobrazení jinou technikou.

Nový modul tedy bude zobrazovat data stejnou technikou **liniového grafu**, rozdíl bude ve zdroji dat. Všechny ostatní moduly systému SigProc zobrazují pouze reálná data a neumí komunikovat s databází. Nový modul bude data číst právě z databáze. Všechny ostatní moduly také zobrazují data jen podle na pevno nastavené konfigurace a není možné zobrazení dat v grafu upravovat nebo filtrovat. Nový modul bude uživateli umožňovat různé operace s grafem v uživatelském rozhraní podle požadavků 4.2.

Kapitola 4
Analýza a návrh

4.1 Současná funkcionalita systému SigProc

Firma Safibra s.r.o vlastní webový software SigProc, který je určen pro vizualizaci dat z optických senzorů průhybu u založených na technologii braggovských mřížek.

Současná verze programu umí zobrazit pouze data v reálném čase, která přímo přicházejí ze senzorů, ale neumí načíst data uložená v databázi. Použití tohoto software je proto značně omezený. Není možné se podívat na starší data či v případě výpadku systému se data ztratí a není možné se na ně opět podívat. Dalším velkým omezením je zobrazení dat v grafu pouze podle na pevno nastavených parametrů a není možné, jakkoliv měnit zobrazení grafu v uživatelském rozhraní.

Proto vznikl požadavek od společnosti Safibra s.r.o. na nový modul do aplikace SigProc, který bude sloužit právě k zobrazení dat, která jsou uložena v databázi. Nový modul bude také umožňovat uživateli právě měnit zobrazení grafu přímo v uživatelském rozhraní podle zadaných požadavků 4.2. Cílem této diplomové práce je zaměřit se na uživatelskou přívětivost a pohodlnost uživatelského rozhraní nově vytvořeného software.

Nový modul bude mít název `Database Graph` neboli databázový graf.

Obrázek 4.1. Software SigProc

4.2 Funkční požadavky

Model funkčních požadavků vznikal postupně během vývoje. Následující popis je seznam požadavků, které byly kladeny na nový modul.

4.2.1 Konfigurace

Modul bude mít několik konfiguračních parametrů (viz 4.5). Tyto parametry se nastaví při vytvoření modulu, jako je to u všech ostatních modulů v software SigProc.

4.2.2 Načítání dat z databáze

Modul bude umět načítat data, uložená v databázi. Přístupové údaje k databázi budou zadané v konfiguračním souboru. Jedná se o data streamovaná, tzn. do databáze se ukládají kontinuálně nová a nová data a modul musí umět zobrazovat i nová data v reálném čase.

4.2.3 Vybírání části dat a konkrétního senzoru

Data ze senzorů jsou velice obsáhlá a nebylo by vhodné je ukládat do jednoho jediného souboru. Proto se data rozdělují do několika různých částí podle období. Je potřeba, aby bylo možné v modulu vybrat, která část dat se má zobrazit a také ze kterého konkrétního senzoru to má být.

4.2.4 Dva zobrazovací módy

Modul bude mít dva módy zobrazení – `Start now` a `Fixed`. Mód `Start now` bude zobrazovat v reálném čase všechna data včetně nově příchozích. Mód `Fixed` zastaví zobrazování nově příchozích dat a bude v něm možné data filtrovat. Počáteční mód se nastaví v konfiguraci. Z módu `Start now` do módu `Fixed` se přejde při stisknutí jakéhokoliv tlačítka, které filtruje data nebo pracuje s úpravou zobrazení v grafu (zoom, posun, rozšíření). Z módu `Fixed` do módu `Start now` se přejde stisknutím tlačítka `Return to Start now`.

4.2.5 Filtrování podle zadaného časového intervalu

Uživatel bude mít možnost zobrazit data jen podle jím zvoleného časového intervalu. Filtrovat bude možné třemi způsoby:

1) Podle přesně zadaného časového intervalu od – do.
2) Zadáním počátečního času a délky intervalu.
3) Zadáním délky intervalu posledních dat od současnosti.

4.2.6 Posouvání časového intervalu

Uživatel bude mít možnost jednoduše posouvat zvolený časový interval dopředu či dozadu v čase. Velikost posouvacího intervalu se nastavuje v konfiguračních parametrech.

4.2.7 Rozšíření či zkrácení časového intervalu

Uživatel bude mít možnost jednoduše rozšířit či zkrátit zvolený časový interval. Pro rozšíření budou dvě možnosti – rozšířit doprava a rozšířit doleva. Stejní dvě možnosti budou i pro zkrácení. Procentuální poměr rozšiřování a zkracování intervalu se nastavuje v konfiguračních parametrech.

4.2.8 Přibližování a oddalování

Uživatel bude mít možnost data v grafu přibližovat či oddalovat. Procentuální poměr přiblížení a oddálení se nastavuje v konfiguračních parametrech.

4.2.9 Posun nahoru a posun dolů

Uživatel bude mít možnost zobrazená data v grafu posunout nahoru či dolů. Procentuální poměr posunu se nastavuje v konfiguračních parametrech.

4.2.10 Přepínání módu automatického měřítka

V módu `Start now` při příchodu nových dat je potřeba řešit problém úpravy měřítka grafu. Někdy je vhodné měřítko automaticky upravovat a někdy je naopak vhodné ponechat původní měřítko.

Uživatel bude mít možnost přepínat mezi třemi módy:

1) Povolit – automatické měřítko zapnuté. Graf automaticky upraví měřítko podle nově příchozích dat.

2) Zakázat – automatické měřítko vypnuté. Graf ponechá současné měřítko.
3) Držet min a max – graf ponechá maximum a minimum, které je nastavené v konfiguraci.

4.2.11 Export zobrazených dat

Uživatel bude mít možnost exportovat zobrazená data do formátu CSV, který je pak dále možné importovat do běžně dostupných tabulkových procesorů. Při kliknutí na tlačítko `Export to CSV` se otevře dialog, kde uživatel zadá jméno souboru, kam chce data uložit. Do souboru se vždy uloží data, která jsou aktuálně zobrazená v grafu (tzn. data, která si uživatel sám vyfiltroval).

4.2.12 Pravidelná kontrola databáze na nová data

Modul bude v pravidelném časovém intervalu kontrolovat databázi pro nová data a tato data okamžitě zobrazovat.

4.3 Nefunkční požadavky

4.3.1 Bezpečnost

Do aplikace ani do konfigurace modulu nebude možné se dostat bez znalosti přistupovacích údajů (jména a hesla). Z modulu nebude možné, jakkoliv zasahovat do databáze dat, pouze data zobrazovat.

4.3.2 Funkčnost ve všech webových prohlížečích

Software bude fungovat na všech v současnosti nejrozšířenějších webových prohlížečích [3] (tzn. Google Chrome, Microsoft Edge/Internet Explorer, Mozilla Firefox, Safari a Opera).

4.3.3 Intuitivní uživatelské rozhraní

Všechna funkčnost modulu bude vytvořena s důrazem na jednoduché uživatelské rozhraní. Všechny akce se budou provádět pouze jedním jednoduchým stisknutím tlačítka. Tlačítka budou dobře popsána, aby i uživatel, který vidí aplikaci poprvé, jasně věděl, k čemu které tlačítko slouží.

4.3.4 Rozšiřitelnost a modifikovatelnost

Modul bude naprogramován s důrazem na budoucí rozšiřitelnost a modifikovatelnost. Musí být možné v budoucnu upravit či přidělat další funkcionalitu. Modul tedy bude mít co nejmenší provázanost a co největší zapouzdření komponent.

4.3.5 Dokumentace

Funkčnost modulu bude dostatečně zdokumentována pomocí komentářů ve zdrojovém kódu. Dokumentace musí být dostatečně pochopitelná pro budoucí vývojáře.

4.4 Data z databáze

Data jsou uložena v databázi PostgreSQL a jsou rozdělena do jednotlivých tabulek na několik různých částí podle časových úseků. V uživatelském rozhraní je žádoucí zobrazení vždy jen dat z jednoho konkrétního senzoru z jedné konkrétní části dat. Je žádoucí také, aby se nezobrazovala data, která byla naměřena během nějaké události, která mohla měření ovlivnit (například průjezd vlaku). Data mají atributy uvedené v tabulce 4.1:

Parametr	Datový typ	Popis
ID	integer	Primární klíč ID
UnitID	character	ID jednotky
SenzorID	character	ID senzoru
DateTime	timestamp	Datum a čas záznamu
Value	real	průměrná hodnota
MinValue	real	minimální hodnota v měřeném úseku
MaxValue	real	maximální hodnota v měřeném úseku
ValueStatus	smallint	status
IsEventData	integer	Parametr určující, zda jde o hodnotu naměřenou během události
ReferenceSenzorID	character	ID referenčního senzoru

Tabulka 4.1. Data z databáze

4.5 Konfigurační parametry

V tabulce 4.2 je uveden seznam parametrů, které je možné nastavit v konfiguraci před spuštěním modulu. U každého parametru je uveden jeho datový typ a krátký popis.

Parametr	Datový typ	Popis
partition_id	string	Implicitně vybraná část dat
unit_id	string	Implicitně vybraná senzorová jednotka
sensor_id	string	Implicitně vybraný senzor
min_y	double	Dolní mez osy y
max_y	double	Horní mez osy y
autoscale_y	config_item_option_list	Módy disabled / enabled / keep_min_max
x_axis_mode	config_item_option_list	Fixed / Start now
x_scroll_step	double	Interval posunu zobrazení
x_zoom_ratio	double	Procentuální poměr přiblížení a oddálení u osy x
y_zoom_ratio	double	Procentuální poměr přiblížení a oddálení u osy y
y_pan_step	double	Procentuální poměr posunu u osy y

Tabulka 4.2. Konfigurační parametry modulu Database Graph

4.6 Popis konfiguračních parametrů

4.6.1 partition id

Implicitně vybraná část dat. Tímto parametrem je možné nastavit, která část dat se bude zobrazovat implicitně po zapnutí modulu.

4.6.2 unit id

Implicitně vybraná senzorová jednotka. Tímto parametrem je možné nastavit ze které senzorové jednotky se budou zobrazovat data implicitně po zapnutí modulu.

4.6.3 sensor id

Implicitně vybraný senzor. Tímto parametrem je možné nastavit ze kterého senzoru se budou zobrazovat data implicitně po zapnutí modulu.

4.6.4 min y

Dolní mez osy y. Tento parametr určuje minimum na ose y při módu automatického měřítka 4.2.10 `Držet maximum a minimum`.

4.6.5 max y

Horní mez osy y. Tento parametr určuje maximum na ose y při módu automatického měřítka 4.2.10 `Držet maximum a minimum`.

4.6.6 autoscale y

Módy automatického měřítka 4.2.10
Tento parametr má tři možnosti:

1) Povolit – automatické měřítko zapnuté. Graf automaticky upraví měřítko podle nově příchozích dat.
2) Zakázat – automatické měřítko vypnuté. Graf ponechá současné měřítko.
3) Držet min a max - graf ponechá maximum a minimum, které je nastavené v konfiguraci.

4.6.7 x axis mode

Módy zobrazení dat v grafu 4.2.4
Tento parametr má dvě možnosti:

1) fixed – pevný časový interval,
2) start now – daný časový interval od současnosti zpět.

4.6.8 x length

Délka časového intervalu zobrazených dat. Jde o údaj v sekundách.

4.6.9 x start

Počáteční čas zobrazení dat pro mód `Fixed`. Parametr je ve formátu String, který se skládá z datumu a času. Přesný formát: `RRRR-MM-DD hh:mm:ss.s`.

4.6.10 x scroll step

Interval, o který se posouvá zobrazení dat na ose x. Jde o údaj v sekundách.

4.6.11 x zoom ratio

Procentuální poměr přiblížení a oddálení u osy x v procentech. Např. 80 znamená, že při přiblížení se zvětší měřítko o 80 procent původního rozsahu.

4.6.12 y zoom ratio

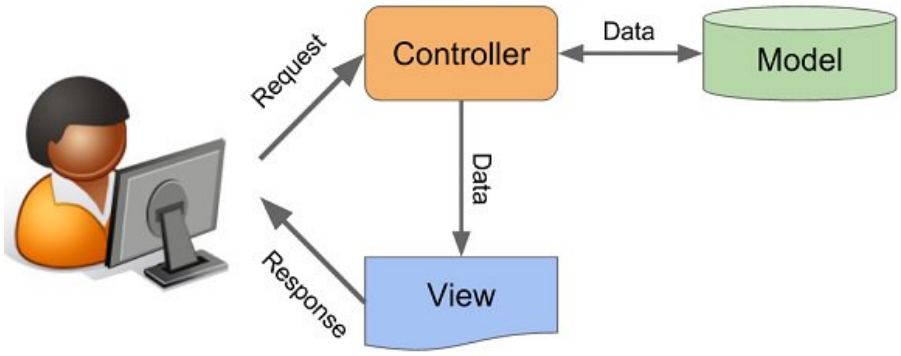
Procentuální poměr přiblížení a oddálení u osy y v procentech. Např. 80 znamená, že při přiblížení se zvětší měřítko o 80 procent původního rozsahu.

4.6.13 y pan step

Interval, o který se posouvá zobrazení dat na ose y. Jde o konkrétní hodnotu.

4.7 Architektura

Modul je navržen softwarovou architekturou Model-view-controller, která rozděluje datový model aplikace, uživatelské rozhraní a řídící logiku do tří nezávislých komponent. Největší výhodou této architektury je právě nezávislost komponent. Modifikace jakékoliv komponenty má jen minimální vliv na ostatní. Matematický aparát je naimplementován samostatně ve své vlastní třídě.



Obrázek 4.2. Model-view-controller

4.7.1 Model

Model je reprezentován databází, ve které jsou uložena všechna data. Modul je napojen na databázi používanou firmou Safibra, která je tvořena open source databázovým systémem PostgreSQL.

Obrázek 4.3. PostgreSQL

4.7.2 View

View je tvořen třídou `wt_graph_view_database.cpp`. Jde o uživatelské rozhraní, se kterým pracuje uživatel. Tato vrstva převádí data reprezentovaná modelem do podoby vhodné k zobrazení uživateli. V této komponentě jsou naprogramovány všechny operace spojené s uživatelským rozhraním.

4.7.3 Controller

Controller je tvořen třídou `graph_database.cpp`. Jde o mezistupeň, který spojuje Model a View. Jsou zde naprogramovány všechny operace, která nejsou přímo spojena s uživatelským rozhraním. Jde o řadič, který reaguje na události pocházející od uživatele a zajišťuje změny v modelu či uživatelském rozhraní.

4.7.4 Matematický aparát

Matematický aparát je naimplementován ve třídě `bendline.cpp`. Je zde naimplementován proces výpočtu dat ze senzorů průhybu.

4.8 Progamovací jazyk C++

Celý systém SigProc je implementován v jazyce C++. Tento jazyk je jeden z nejrozšířenějších programovacích jazyků na světě. Jedná se o multi-paradigmatický programovací jazyk, což znamená, že podporuje více různých programovacích paradigmat – objektově orientované, procedurální i funkcionální, podle toho, co se na danou úlohu nejvíce hodí.

Mezi největší výhody C++ patří:

- C++ je výkonný, efektivní a rychlý programovací jazyk, dá se použít pro vývoj uživatelského rozhraní, 3D grafické hry nebo matematické simulace v reálném čase.
- Velká portabilita - jedná se o programovací jazyk použitelný na všechna možná prostředí. Často se používá na vývoj multiplatformních aplikací.
- Objektově orientovaný - C++ umožňuje vytvářet třídy, dědičnost, polymorfismus nebo abstrakci.
- C++ má velmi bohatou knihovnu funkcí.
- Oproti jazyku C umožňuje ošetřování výjimek nebo přetěžování tříd.

4.9 WebToolkit

Wt WebToolkit je open source webový framework pro programovací jazyk C++. Logo viz. obrázek 4.4 Tento framework je orientovaný na webové widgety a je převážně určen pro implementaci uživatelského rozhraní webových aplikací.

Velkou výhodou frameworku WebToolkit jsou zabudované bezpečnostní prvky, které zabraňují nebezpečným technikám Cross-site scripting nebo Cross-site request forgery.

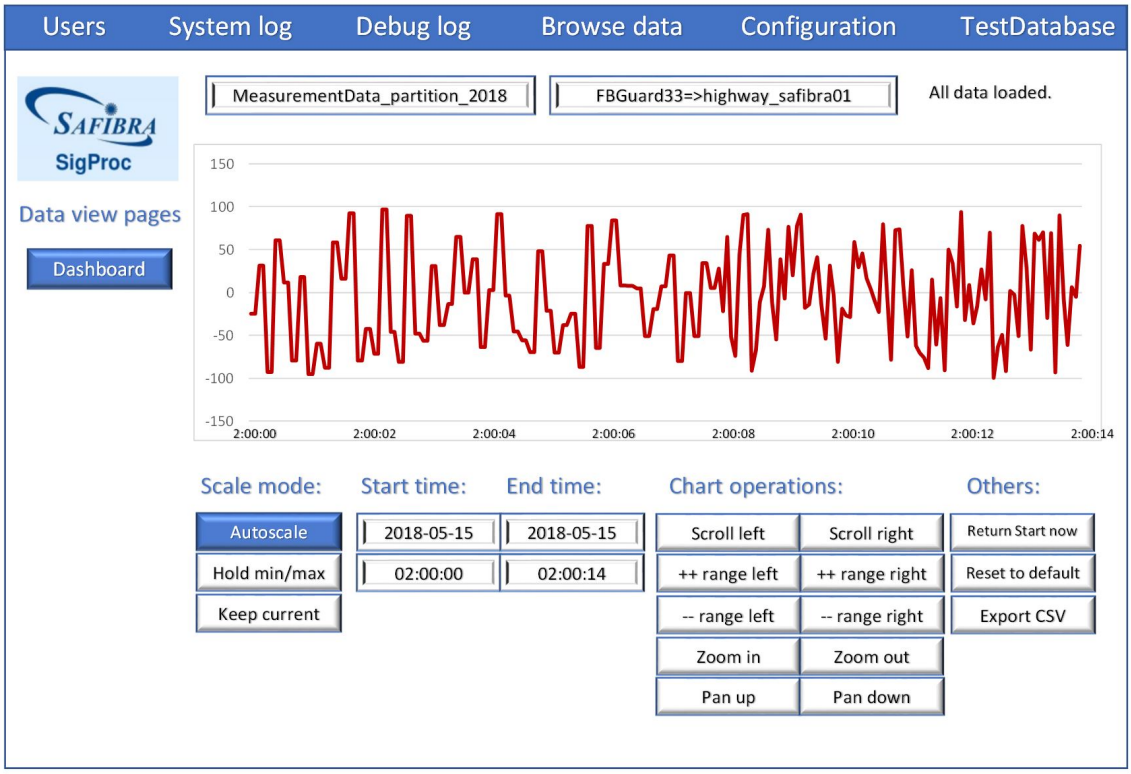
WebToolkit má také dobře řešené poslouchání a zpracování událostí ze vstupních zařízení (myš, klávesnice atd.).

Obrázek 4.4. WebToolkit

4.10 Ovládání v uživatelském rozhraní

Uživatelské rozhraní modulu je tvořeno oblastí ve které je zobrazen graf s daty, oblastí nad grafem, kde jsou umístěny dialogy pro výběr části dat a konkrétního senzoru a oblastí pod grafem, kde jsou umístěna tlačítka a kontejnery pro interakci uživatele s grafem, viz obrázek 4.5.

V případě stisknutí jakéhokoliv tlačítka pošle uživatelské rozhraní informaci o stisknutí řídící metodě, která pošle nový SQL dotaz do databáze a přepošle nová data do uživatelského rozhraní.



Obrázek 4.5. Uživatelské rozhraní

4.10.1 Dialogy pro vybrání dat

Sekce nad grafem obsahuje dva rozbalovací dialogy pro vybrání konkrétních dat, které chce uživatel zobrazit. Jeden dialog obsahuje seznam všech částí dat a druhý dialog obsahuje seznam konkrétních senzorů ve formátu `jednotka => senzor`. Po stisknutí dialogu se seznam rozbalí. Po vybrání jiné hodnoty pošle uživatelské rozhraní tuto informaci řídící vrstvě, která spustí načtení nových dat.

4.10.2 Kontejner pro načítací hlášku

V tomto kontejneru se zobrazuje hláška informující o stavu načítání. Existuje pět možných hlášek, které se můžou zobrazit:

- `Počkejte, data se načítají!` - je hláška, která je zobrazena v průběhu načítání dat.
- `Všechna data jsou načtena` - je hláška, která se zobrazí po načtení všech dat.
- `Žádná data` - je hláška, která se zobrazí, pokud senzor neobsahuje žádná data.
- `Více než 10000 dat, zobrazeno pouze 10000.` - je hláška, která se zobrazí v případě, že senzor obsahuje více něž 10000 datových záznamů.
- `Mód start-now, data se načítají kontinuálně.` - je hláška, která se zobrazí při módu start-now.

4.10.3 Tlačítka automatického měřítka

Módy automatického měřítka jsou přepínány třemi spojenými tlačítky:

- automatická škála,
- držet minimum a maximum,
- ponechat současné měřítko.

Tlačítko reprezentující právě vybraný mód je zvýrazněné modře.

4.10.4 Kontejnery pro časový filtr

Časový filtr je reprezentován čtyřmi kontejnery, kam uživatel zadává čas, podle kterého chce data vyfiltrovat. Filtruje se podle počátečního a koncového času. Oba časy jsou reprezentovány dvěma kontejnery - jeden pro datum ve formátu `RRRR-MM-DD` a druhý pro čas ve formátu `hh:mm:ss`.

Kontejnery automaticky reagují na změnu datumu nebo času, kterou uživatel udělá a pošle změnu řídící metodě, která pošle nový SQL dotaz do databáze a přepošle nová data do uživatelského rozhraní.

4.10.5 Tlačítka pro posun časového intervalu

Uživatelské rozhraní obsahuje 2 tlačítka, která posouvají časový interval o počet sekund nastavených v konfiguraci (podle x_scroll_step 4.6.10):

- Posun o časový interval vlevo
- Posun o časový interval vpravo

4.10.6 Tlačítka změnu časového intervalu

Uživatelské rozhraní obsahuje 4 tlačítka, která mění časový interval o procentuální poměr nastavený v konfiguraci (x_zoom_ratio 4.6.11):

- zvětšení časového intervalu vlevo,
- zvětšení časového intervalu vpravo,
- zmenšení časového intervalu vlevo,
- zmenšení časového intervalu vpravo.

4.10.7 Tlačítka pro přiblížení a oddálení dat

Uživatelské rozhraní obsahuje 2 tlačítka, která mění přiblížení dat v grafu o procentuální poměr nastavený v konfiguraci (y_zoom_ratio 4.6.12):

- přiblížit data,
- oddálit data.

4.10.8 Tlačítka pro posun dat

Uživatelské rozhraní obsahuje 2 tlačítka, která posouvají data o hodnotu nastavenou v konfiguraci (y_pan_step 4.6.13):

- posun dat nahoru,
- posun dat dolu.

4.10.9 Kontejnery pro úpravu parametrů

Uživatelské rozhraní obsahuje 4 kontejnery pro úpravu parametrů pro posouvání dat v grafu, které jsou implicitně nastavené v konfiguraci:

- úprava přibližovacího poměru,
- úprava úseku posunu,
- úprava posouvacího úseku,
- úprava časového rozmezí.

Do kontejnerů je možné novou hodnotu zapsat přímo manuálně nebo hodnotu posouvat pomocí šipek nahoru a dolů.

4.10.10 Tlačítko pro návrat do režimu Start now

Toto tlačítko při stisknutí vrátí graf do režimu Start now .

4.10.11 Tlačítko reset

Tlačítko, které obnoví implicitní zobrazení dat podle konfiguračních parametrů. Při stisknutí tlačítka se ještě zobrazí potvrzující dialog, který si vyžádá od uživatele potvrzení, jestli opravdu chce zobrazení vrátit do původního stavu.

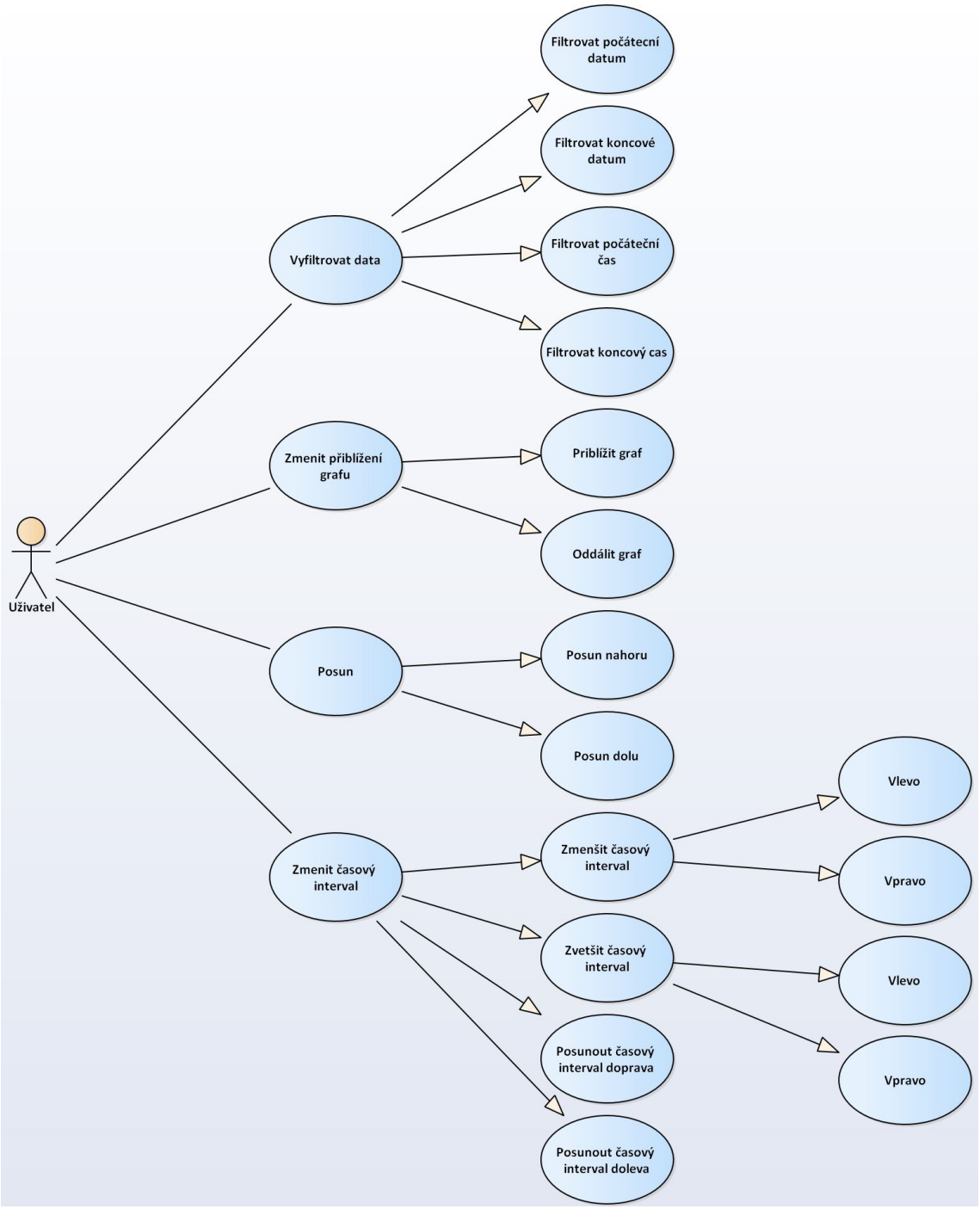
4.10.12 Tlačítko export

Tlačítko, které spustí export zobrazených dat do formátu CSV. Při stisknutí tlačítka se zobrazí dialog, který vyzve uživatele pro zadání jména souboru, do kterého se mají data uložit.

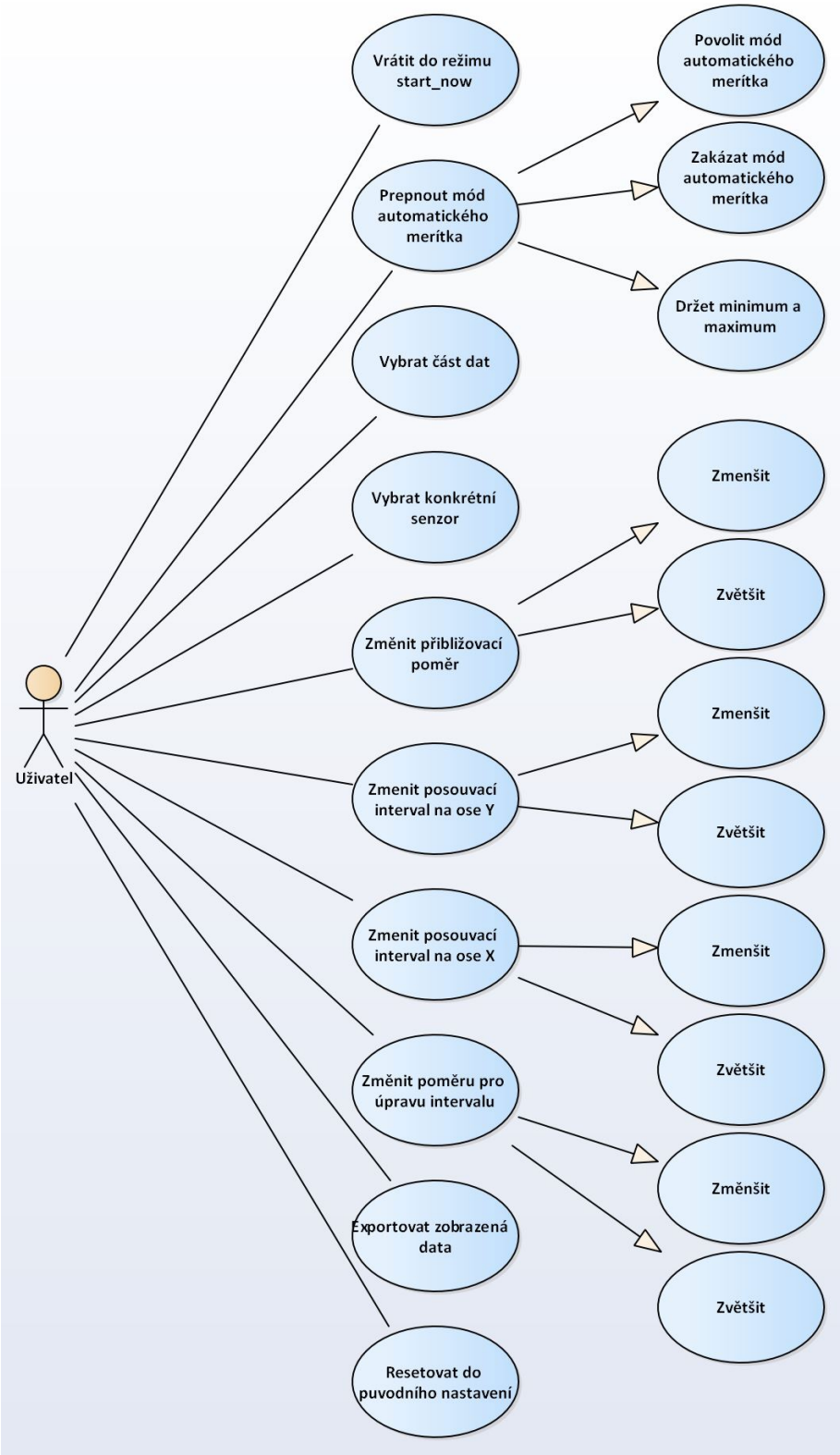
4.11 Případy užití

Na dvou diagramech níže 4.6 a 4.7 jsou znázorněny všechny případy užití, které uživatel aplikace může provést. První diagram obsahuje případy užití spojené přímo úpravou zobrazení grafu a druhý diagram obsahuje všechny ostatní případy užití. [1]

[1] Při vytváření diagramů jsem použil program Enterpsise Architect od firmy Spark Systems, který je pro vytváření podobných diagramů skvělý. Program má ale bohužel problém s českou diakritikou. Neustále nedeterministicky maže z písmen háčky a čárky. Nepodařilo se mi vytvořit diagram bez chyby. Proto prosím omluvte chybějící háčky a čárky v diagramech.



Obrázek 4.6. Diagram případů užití



Obrázek 4.7. Diagram případů užití 2

Kapitola 5
Implementace

Celý modul je naimplementován ve dvou třídách. Veškeré operace spojené s uživatelským rozhraním jsou v souboru `wt_graph_view_database.cpp` a veškerá řídící logika v souboru `graph_database.cpp`. Matematické vypočítání dat ze senzorů je zvlášť ve třídě bendline.cpp.

V implementaci je využit C++ framework WebToolkit a další nativní knihovny programovacího jazyka C++.

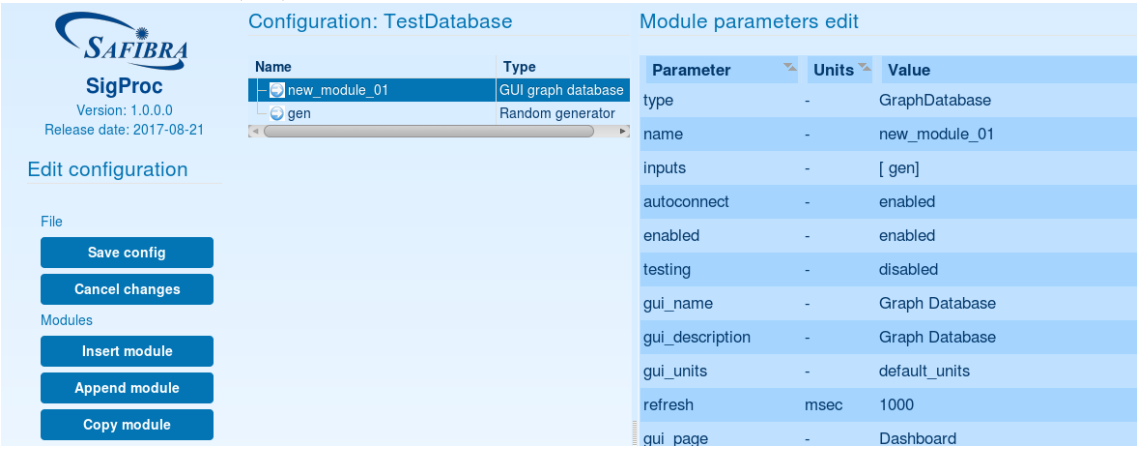
5.1 Spouštění

Nový modul se spouští ve webové aplikaci SigProc v sekci konfigurace. Viz obrázek 5.1.

Obrázek 5.1. Spouštění modulu

5.1.1 Konfigurace

V nastavení konfigurace je možné před spuštěním modulu přenastavit všechny konfigurační parametry (4.5).



Obrázek 5.2. Nastavení modulu

5.2 Řídící proměnné

V řídící logice je naimplementováno pět řídících proměnných typu boolean:

- `firstQuery` – tato proměnná určuje, zda jde o první dotaz do databáze, čili se mají načíst všechna data bez jakéhokoliv filtrování nebo jde o již dotaz s filtrováním.
- `loadFromDatabase` – tato proměnná určuje, zda se mají znovu načíst data z databáze.
- `isLoading` – tato proměnná uchovává informaci, zda zrovna probíhá proces načítání data z databáze.
- `noData` – tato proměnná uchovává informaci, zda ve vybraném senzoru nejsou žádná data.
- `tooManyData` – tato proměnná uchovává informaci, zda ve vybraném senzoru je více než 10000 datových záznamů.

V uživatelském rozhraní jsou implementovány tyto čtyři řídící proměnné:

- `isValid` – jedná se o pole čtyř proměnných typu boolean, které určují, zda jsou zadané vstupy ve formulářích pro datumy a časy validní či ne.
- `minValue` – proměnná, která uchovává informaci o současné minimální hodnotě v grafu.
- `maxValue` – proměnná, která uchovává informaci o současné maximální hodnotě v grafu.
- `dateTimeLoaded` – proměnná, která uchovává informaci, zda byl načten datum a čas
- `zoomInterval` – proměnná, která uchovává hodnotu intervalu, o který se má graf přibližovat či oddalovat.

5.3 Konstanty

V řídící logice jsou naimplementovány všechny konstanty, které uchovávají data z konfigurace (viz. 4.5).

V uživatelském rozhraní jsou naimplementovány konstanty typu double `dateFormat` a `timeFormat`, které uchovávají informaci o formátech datumu a času.

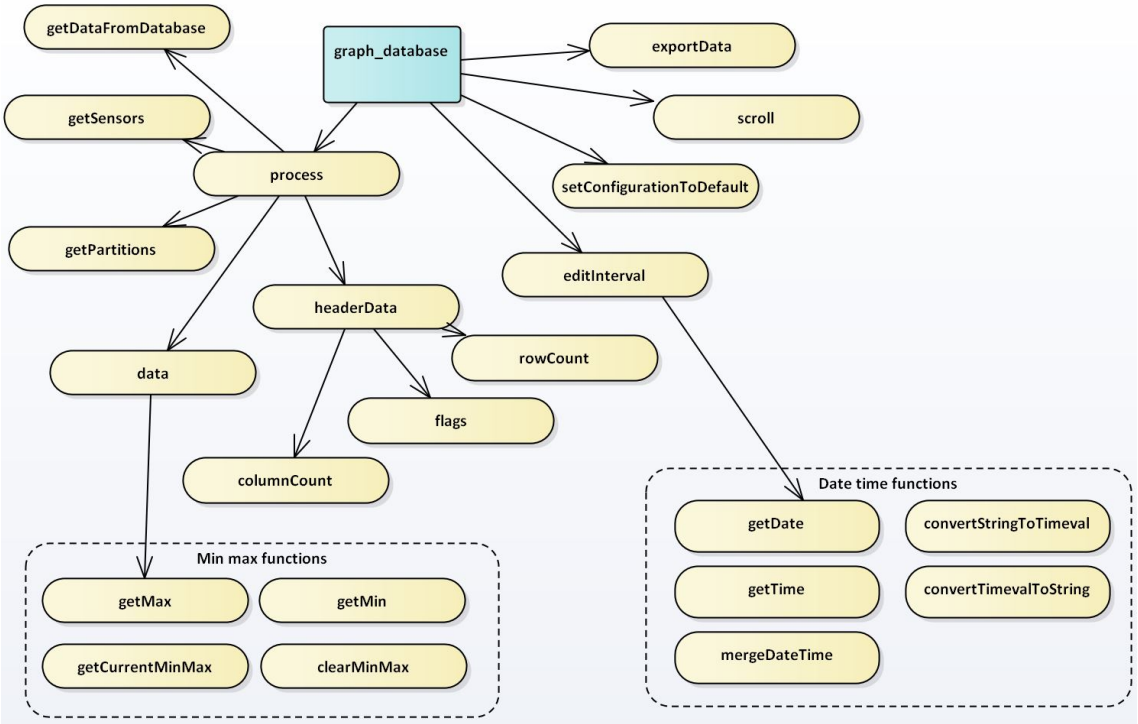
5.4 Struktura Sensor

V řídící logice je naimplementována struktura (struct) Sensor, která uchovává informace o senzorech. Struktura má dva parametry typu string:

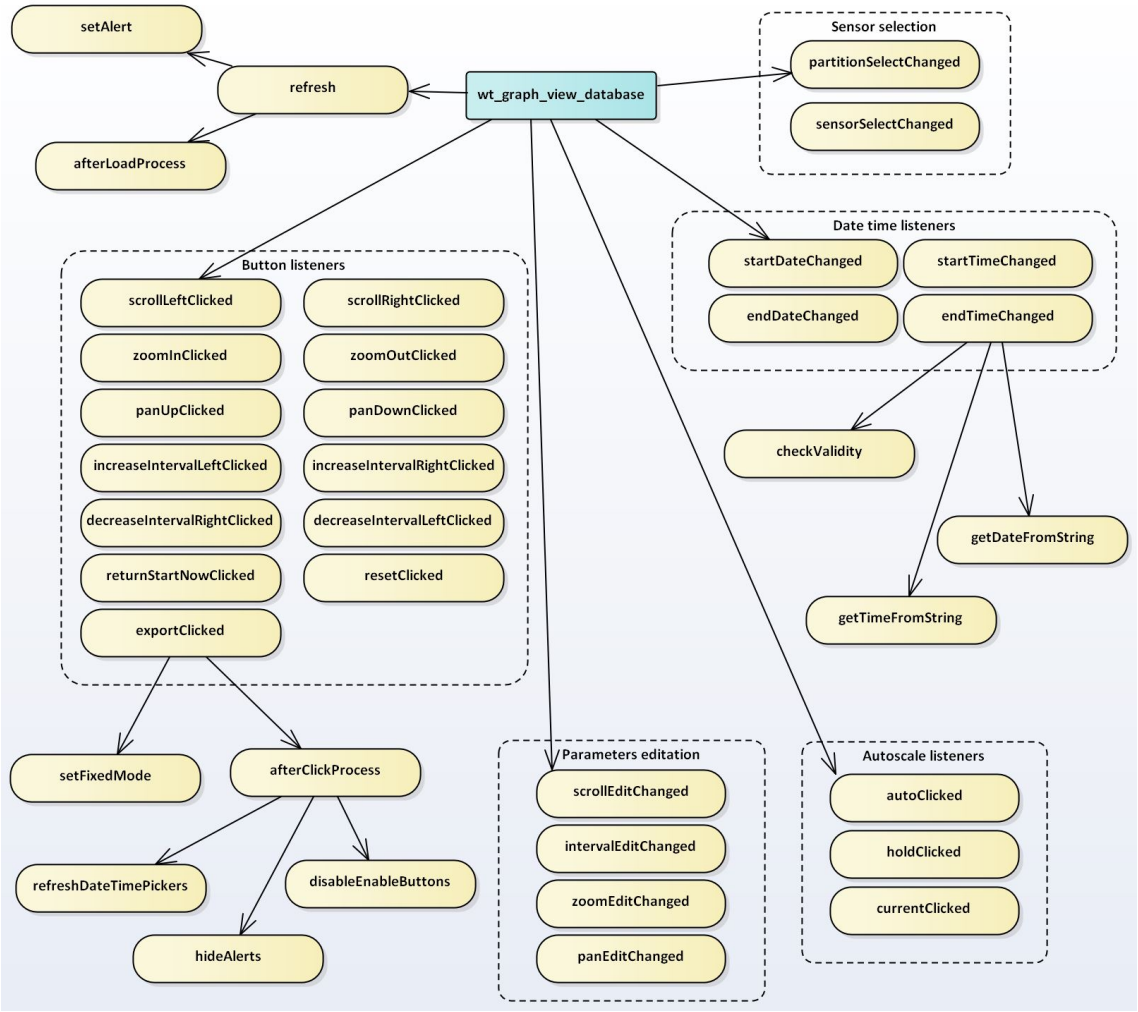
- `unitId` – jedná se o název jednoho, ve které se senzor nachází.
- `sensorId` – jedná se o název daného senzoru.

5.5 Diagramy metod

Na dvou obrázcích níže (5.3 a 5.4) je schematicky zobrazená hierarchie metod, neboli strom volání (call strom):



Obrázek 5.3. Diagram metod



Obrázek 5.4. Diagram metod 2

5.6 Řídící logika

Řídící logika v souboru `graph_database.cpp` je složena z těchto částí:

- načtení dat z databáze,
- metody pro filtrování dat,
- práce s minimem a maximem,
- export dat,
- práce s datumy a časy.

5.6.1 Načtení dat z databáze

Pro načtení dat z databáze slouží metody `getPartitions` a `getSensors`. Tyto metody mají za úkol načíst seznam částí dat a názvy konkrétních senzorů. Dělají to posláním dotazu Select do databáze. A také metoda `getDataFromDatabase`, která načte data z konkrétního vybraného senzoru. Tato metoda pošle do databáze dotaz `Select` s příslušnými filtrovanými počátečními a koncovými časy (v případě prvního dotazu bez filtrování). Poté systém data načte do datové struktury, ze které si pak data vezme uživatelské rozhraní na zobrazení do grafu. Nakonec metoda z dat vyčte počáteční a koncový čas a předá je pro zobrazení do formulářů k tomu určených.

Kód dotazu do databáze:

```
if(firstQuery) {
q1 = app->getDbPostgresClient()->push_sql_query_select("SELECT * FROM
\"" + selectedPartition + "\" WHERE \"IsEventData\"=0 AND \"UnitID\"=\'"
 + selectedUnit + "\' AND \"SensorID\"=\'" + selectedSensor + "\'
 ORDER BY \"DateTime\";");
} else {
q1 = app->getDbPostgresClient()->push_sql_query_select("SELECT * FROM
\"" + selectedPartition + "\" WHERE \"IsEventData\"=0 AND \"UnitID\"=\'"
 + selectedUnit + "\' AND \"SensorID\"=\'" + selectedSensor + "\' AND
 \"DateTime\" BETWEEN '" + start + "' AND '" + end + "' ORDER BY
 \"DateTime\";");
}
```

5.6.2 Metody pro filtrování dat

V řídící logice jsou dvě metody pro filtrování dat – `scroll` a `editInterval`. Metoda `scroll` má vstupní parametr, který určitě, jestli se bude graf posunovat doprava nebo doleva. Metoda `editInterval` má parametr pro určení změny vpravo nebo vlevo a také parametr pro určení, zda jde o zvětšení či zmenšení intervalu.

U implementace funkce `editInterval` bylo potřeba zakázat zmenšení intervalu na nulu, protože poté by již nebylo možné interval opět zvětšit vzhledem k procentuální povaze zvětšování.

Metoda `editInterval`:

```
void GraphDatabase::editInterval(bool isLeft, bool isDecrease) {
std::string startDateTime = mergeDateTime(startDate, startTime);
std::string endDateTime = mergeDateTime(endDate, endTime);
timeval startTimeval = convertTimeStringToTimeval(startDateTime);
timeval endTimeval = convertTimeStringToTimeval(endDateTime);

int interval = endTimeval.tv_sec - startTimeval.tv_sec;
int decreaseInterval = interval*_x_zoom_ratio/100;
int increaseInterval = interval*(200-_x_zoom_ratio)/100 + 1;
if(increaseInterval == interval) {
increaseInterval++;
}

if (isDecrease) {
if (isLeft) {
startTimeval.tv_sec = endTimeval.tv_sec - decreaseInterval;
if(startTimeval.tv_sec == endTimeval.tv_sec) {
startTimeval.tv_sec--;
}
} else {
endTimeval.tv_sec = startTimeval.tv_sec + decreaseInterval;
if(startTimeval.tv_sec == endTimeval.tv_sec) {
endTimeval.tv_sec++;
}
}
} else {
if (isLeft) {
startTimeval.tv_sec = endTimeval.tv_sec - increaseInterval;
```

```
} else {
endTimeval.tv_sec = startTimeval.tv_sec + increaseInterval;
}
}
}
```

5.6.3 Práce s minimem a maximem

Metoda `getCurrentMinMax` slouží k určení minima a maxima z dat pro správné nastavení rozmezí grafy. Metoda `clearMinMax` naopak tyto maxima a minima vynuluje.

5.6.4 Export dat

Metoda `exportData` slouží pro export aktuálně zobrazených dat v grafu do souboru ve formátu CSV. Metoda má vstupní parametr, který určí jméno souboru pro uložení dat.

5.6.5 Práce s datumy a časy

Datum a čas každého záznamu je v databázi uložen ve formátu `string` a formuláře pro výběr datumu a času v uživatelském rozhraní pracují s datovým typem `timeval`. Proto bylo nutné naimplementovat metody `convertStringToTimeval` a `convertTimevalToString`, které převádí jeden formát do druhého a naopak.

V databázi je uložena informace o datumu i času v jednom řetězci. Ve formulářích se ale pracuje s každým údajem zvlášť, proto bylo potřeba naimplementovat metody `getTime` a `getDate`, které ze společného řetězce dostanou jednotlivé údaje. A metodu `mergeDateTime`, která naopak jednotlivé údaje zase spojí dohromady.

Metody `convertStringToTimeval` a `convertTimevalToString`:

```
timeval GraphDatabase::convertStringToTimeval(std::string timeString) {
struct tm timeStruct;
strptime(timeString.c_str(), "%Y-%m-%d %H:%M:%S", &tm);
time_t time = mktime(&timeStruct);
timeval result;
result.tv_sec = time;

if(timeString.length() > 20) {
std::string secondFractions = timeString.substr(20,1);
result.tv_usec = std::stod(secondFractions)*100000;
} else {
result.tv_usec = 0;
}
return result;
}
```

```
std::string GraphDatabase::convertTimevalToString(timeval inputTime) {
time_t time = inputTime.tv_sec ;
struct tm *tempStruct ;
char newTime[30];
tempStruct = localtime( &tttime);

strftime(newTime, sizeof(newTime), "%Y-%m-%d %H:%M:%S", tmp);

std::string usecString = std::to_string(inputTime.tv_usec);
std::string result;
result.assign(newTime);
```

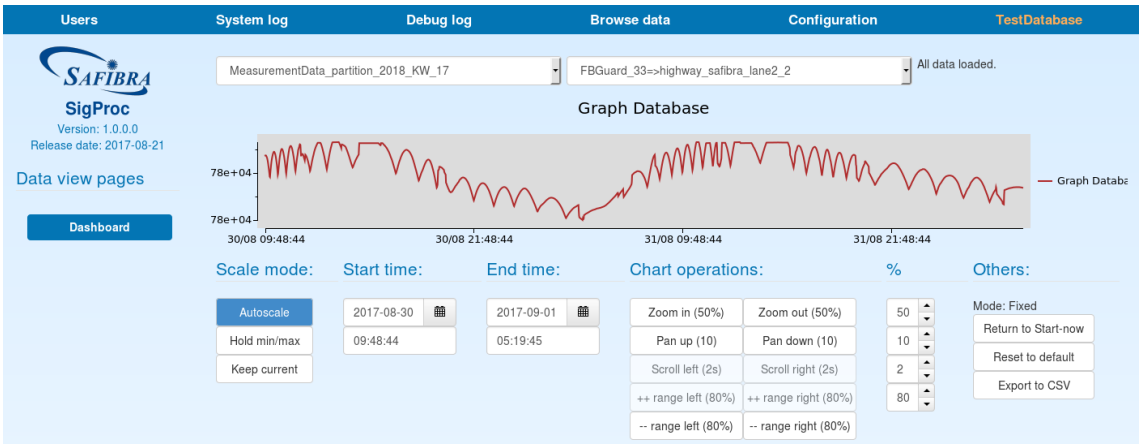
```
result = result + "."+ usecString.substr(0,1);

return result;
}
```

Zajímavostí také je nutnost konverze mezi formátem z knihovny WebToolkit WString, která uchovává řetězec do běžného formátu std::string:

```
std::wstring wString = sensorSelect->valueText();
std::string str(wString.begin(), wString.end());
}
```

5.7 Uživatelské rozhraní



Obrázek 5.5. Uživatelské rozhraní

Uživatelské rozhraní v souboru `wt_graph_view_database.cpp` je složeno z těchto částí:

- sestavení grafických komponent,
- obnovení grafu a proces po načtení dat
- posluchače vstupu,
- posluchače tlačítek,
- proces po stisku tlačítek
- validace
- blokování a povolení tlačítek

5.7.1 Typy grafických komponent

Modul je složen z těchto grafických komponent:

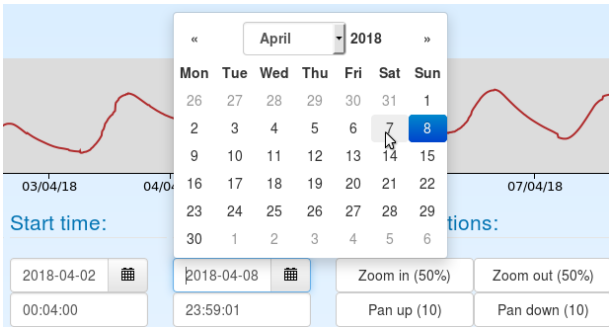
- Wt::WComboBox - je rozbalovací dialog pro vybírání části dat a konkrétního senzoru.
- Wt::WCartesianChart - je liniový graf s kartézskými souřadnicemi.
- Wt::WGroupBox - je kontejner, do kterého se přidá skupina elementů, které patří k sobě. Je jich celkem pět (viz. 5.7.2).
- Wt::WPushButton - je základní klikací tlačítko. V modulu jich je celkem 16 a každé má svou určitou funkci (4.10).
- Wt::WSpinBox - je kontejner pro úpravu parametrů pro posouvání dat v grafu. Je možné přímo napsat hodnotu nebo posouvat hodnotu šipkami nahoru a dolů.

- Wt::WDateEdit - je kontejner pro zvolení data. Při kliknutí zobrazí kalendář, kde si uživatel vybere datum. Je možné datum zadat i přímým napsáním hodnot.
- Wt::WTimeEdit - je kontejner pro zvolení času. Je potřeba manuálně napsat hodnoty.
- Wt::WLabel - je element, který zobrazuje popisky k ostatním elementům.
- Wt::WText - je element, který zobrazuje chybové hlášky.
- Wt::WDialog - je vyskakovací dialog, který slouží k potvrzení resetu nastavení nebo k vybrání jména souboru pro export dat.

5.7.2 Sestavení grafických komponent

Grafické komponenty modulu jsou rozděleny do osmi různých sekcí. V sekci nad grafem jsou rozbalovací dialogy pro výběr části dat a konkrétního senzoru a dialog pro zobrazení načítací hlášky. Pod touto sekcí je samotný graf. Sekce pod grafem je kvůli přehlednosti a uživatelské přívětivosti rozdělena na šet sekcí:

- Automatické měřítko – je sekce, která obsahuje tlačítka pro přepínání módů automatického měřítka
- Počáteční čas – je sekce pro výběr počátečního data a času. Datum se dá vybrat buď manuálním napsáním nebo vybráním z rozbalovacího formuláře. Viz. obrázek 5.6.
- Koncový čas – je sekce pro výběr koncového data a času
- Grafové operace – je sekce, která obsahuje všechna tlačítka pro operace s grafem.
- Úprava parametrů – je sekce pro úpravu parametrů pro posouvání dat v grafu.
- Ostatní operace – sekce, která obsahuje všechna ostatní tlačítka.



Obrázek 5.6. Výběr datumu

5.7.3 Obnovení grafu a proces po načtení dat

Metoda obnovení grafu se stará o pravidelnou obnovu grafu v časovém intervalu nastaveném v konfiguraci. Pro správné fungování obnovování dat v módu `start-now` je potřeba nastavit dostatečně velký interval (viz. sekce omezení 5.8). Na konci tato metoda spustí `proces po načtení dat`, který se skládá z opětovného přenastavení minima a maxima a pro vypočtení přibližovacího intervalu.

5.7.4 Posluchače vstupu

Metody, které detekují změny ve vstupních kontejnerech. Posluchače jsou tři různé - pro detekci změny senzoru, pro detekci změny parametrů a pro detekci změny data a času v kontejnerech pro filtrování data a času. V případě jakékoliv změny v některém z kontejnerů, pošle příslušná metoda nejprve nový vstup k validaci a v případě, že je vstup validní, pošle data a informaci ke změně řídící vrstvě. V případě nevalidních dat zobrazí varovnou hlášku.

5.7.5 Posluchače tlačítek

Metody, které detekují stisk jednotlivých tlačítek. V případě stisknutí některého tlačítka pošle příslušná metoda informaci o stisknutí řídící vrstvě. Poté se spustí `proces po stisku tlačítka` 5.7.6.

5.7.6 Proces po stisku tlačítek

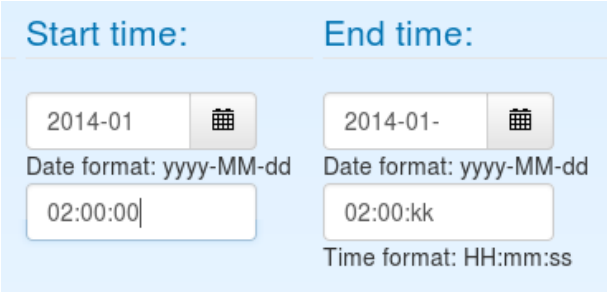
Proces po stisku tlačítka je proces, který se spustí po stisku jakéhokoliv tlačítka. Skládá se ze čtyř operací:

- nastavení načítací hlášky na hlášku - `Počkejte, data se načítají!`,
- obnovení data a času v kontejnerech,
- schování varovných hlášek,
- povolení či zakázání tlačítek.

5.7.7 Validace

Uživatel zadává při filtrování dat manuálně datum a čas, proto je třeba ošetřit možnost zadání nesprávných údajů či údajů ve špatném formátu. V tomu slouží naimplementovaná validace, která při každé úpravě filtru kontroluje, zda jsou vstupní data odpovídají formátu uloženého v konstantách k tomu určených 5.3. V případě, že je některý ze čtyř vstupů nevalidní, systém tuto informaci uloží do proměnné isValid 5.1. Systém pak před každým dotazem do databáze kontroluje, zda jsou všechny proměnné v poli isValid ve stavu true (validní) a stačí když je jedno vstupní pole nevalidní a systém dotaz nespustí.

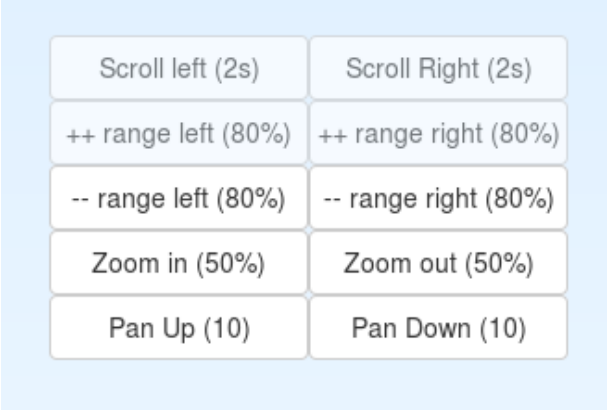
V případě, že je některý ze vstupních formulářů nevalidní, zobrazí se pod tímto formulářem hláška, která uživatele upozorní, že formulář není validní (viz. obrázek 5.7).



Obrázek 5.7. Validace

5.7.8 Blokování a povolení tlačítek

V případě, že se rozmezí grafu dostane na samý okraj dat a posun si zvětšení měřítka tím směrem by už šel mimo rozsah dat, daná tlačítka pro tyto operace jsou automaticky blokována (viz. obr8zek 5.8). Stejně tak pokud se interval rozmezí minima a maxima v grafu dostane na hodnotu pouhé jedné sekundy, zablokují se tlačítka pro zmenšení časového intervalu. V případě, že se graf opět dostane do stavu, kdy některá ze zablokovaných operací opět začne dávat smysl, systém dané tlačítko opět povolí.



Obrázek 5.8. Blokovaná tlačítka

5.8 Omezení

Obnovovací interval pro překreslení grafu nesmí být kratší než jedna sekunda, protože pak systém v módu start-now nestíhá načítat data z databáze a v grafu nic nezobrazí. Jedna sekunda je nicméně dostačující interval pro zobrazení reálných dat v grafu.

Maximální množství dat, které se v grafu zobrazí je omezen na 10000. Pokud senzor obsahuje větší množství dat, zobrazí se pouze posledních 10000 záznamů, nicméně stále je možné pomocí vyfiltrování zobrazit i starší data.

5.9 Matematický aparát

Matematický aparát přepočtu dat ze senzorů do databáze je naimplementován ve třídě `bendline.cpp`. Je zde naimplementován postup popsaný v kapitole 2.5.

5.10 Nasazení

Modul byl po důkladném otestování nasazen přímo do funkční verze webové aplikace SigProc.

5.11 Statistiky kódu

Výsledný modul je naimplementován ve třech různých třídách, které obsahují 52 různých metod. Celkový kód zabírá 1902 řádků kódu.

Kapitola 6
Testování

6.1 Testování autorem

Prototypy modulu byly testovány průběžně v průběhu implementace. V případě nalezení jakéhokoliv problému bylo cílem problém okamžitě odstranit, aby se chyby postupně nenabalovaly. Po naimplementování všech požadavků proběhlo řádné testování autorem celého modulu, rozdělené na čtyři části:

- Testování UI
- Testování na testovací databázi
- Testování na reálné databázi
- Testování rychlosti

6.1.1 Testování UI

Cílem bylo prověřit, zda všechny ovládací prvky uživatelského rozhraní správně fungují. Při testování jsem přišel na dvě chyby, které bylo potřeba opravit. Interval je možné zmenšit až na nulu 6.3.1 a špatné zaokrouhlování mikrosekund 6.3.2. Napadlo mě také nové vylepšení – začít blokovat tlačítka, pokud nejsou relevantní 6.4.1.

6.1.2 Testování na testovací databázi

Cílem bylo prověřit, zda modul zvládá správně zobrazovat data z lokální testovací databáze a zda fungují všechny operace s daty. Při testování na testovací databázi se objevila jedna chyba, a to chybné porovnávání datumů 6.3.3.

6.1.3 Testování na reálné databázi

Cílem bylo prověřit, zda modul zvládá správně zobrazovat reálná data z reálné databáze a zda fungují všechny operace s těmito daty. Při testování na reálné databázi byly odhaleny tři závažné chyby, které bylo nutné opravit. Systém padal v případě, že vybraný senzor neobsahoval žádná data 6.3.4. A další dvě chyby měly spojitost s pomalejším načítáním velkých datových sad. Zaprvé se nesmí nastavit obnovovací interval na menší čas než jedna sekunda 6.3.5. Zadruhé byl odhalen problém se synchronizací 6.3.6.

6.1.4 Testování rychlosti

Cílem bylo prověřit, jak dlouho trvá malých a velkých datových sad. Otestovat, jestli je načtení dostatečně rychlé pro běžné použití.

- Načtení malé datové sady (do 500 záznamů) trvá méně než jednu sekundu.
- Načtení střední datové sady (2000 záznamů) trvá do tří sekund.
- Načtení velké datové sady (10000 záznamů) trvá kolem 9 sekund.

Po konzultaci s firmou Safibra jsme usoudili, že pro běžné použití je současná implementace dostatečně rychlé.

6.2 Usability test

Po důkladném otestování aplikace autorem práce byl proveden usability test, jehož účelem bylo prověřit funkčnost, a hlavně použitelnost aplikace. Účelem tohoto testu bylo prověřit, zda je aplikace pro uživatele snadno použitelná a jestli je logické uspořádání ovládacích prvků v uživatelském rozhraní dostatečně intuitivní. Testování probíhalo na osobním počítači, na kterém se modul vyvíjel a na reálné databázi.

Ideální by bylo provést testování v testovací laboratoři, ale ta ve firmě Safibra není. Snahou bylo testovací laboratoř alespoň trochu napodobit, a tak bylo počínání testerů natočeno na kameru a poté zpětně vyhodnoceno.

6.2.1 Výběr testerů

Pro tento test byli zvoleni dva cíloví uživatelé, kteří již používají systém SigProc a budou s největší pravděpodobností využívat i nový modul. Na testování byli přizváni ještě dva další uživatelé, kteří běžně používají webové aplikace, ale nepoužívají systém Sigproc. Z důvodu získání pohledu z jiného úhlu.

Při usability testu bylo testery objeveno chybné blokování tlačítek 6.3.7. Byly definovány také tři nové vylepšení, které bylo vhodné naimplementovat. Úprava parametrů přímo v UI 6.4.2, přidání senzoru do konfigurace 6.4.3 a úprava přibližovacího poměru 6.4.4.

6.2.2 Testované scénáře

Každému testerovi byl předložen níže uvedený seznam scénářů, které mají provést. Testerům nebyly specifikovány přesné hodnoty, ty si mohl každý tester zvolit sám (např. jaký datum zvolit, kolikrát změnit interval atd.).

1. Změnit část dat.
2. Změnit vybraný senzor.
3. Změnit počáteční datum.
4. Změnit koncové datum.
5. Přiblížit graf.
6. Oddálit graf.
7. Posunout zobrazení data doprava či doleva.
8. Několikrát zmenšit interval.
9. Několikrát zvětšit interval.
10. Jakkoliv změnit parametry pro posouvání dat.
11. Exportovat data.
12. Změnit mód automatického měřítka.
13. Vrátit vše do původního nastavení.
14. Vrátit do módu start-now.

U scénáře 2. vznikla myšlenka vylepšení přidání implicitního senzoru do konfigurace 6.4.3. U scénáře 7. došlo k odhalení chyby chybného blokování tlačítek 6.3.7. U scénářů 5. a 6. vznikla myšlenka nápadu úpravy přibližovacího poměru 6.4.4 a u scénářů 8. a 9. vznikla myšlenka přidání možnosti úpravy parametrů přímo v UI 6.4.2. Ostatní scénáře žádné chyby ani vylepšení nepřinesly.

Byla testována i uživatelská přívětivost a jednoduchost, která se ukázala jako dobrá. Všichni testeři se v systému rychle a bez problémů orientovali.

6.3 Nalezené problémy

V této kapitole jsou podrobně popsané všechny chyby, které byly nalezeny při testování a následně opraveny.

6.3.1 Zmenšení intervalu na nulu

Při testování byl objeven problém, kdy bylo možné časový interval dat zmenšit až na nulu a poté již nebylo možné opět interval zvětšit. Parametr pro zvětšování intervalu je totiž v procentuálním formátu a procentuální zvětšení nulového intervalu je stále nula. Chyba byla opravena zamezením zmenšení intervalu na nulu. Interval je možné zmenšit na minimální čas rovné jedné sekundě.

6.3.2 Zaokrouhlování mikrosekund

Další chyba objevená testováním bylo chybné zaokrouhlování mikrosekund u horní hranice časového intervalu. Mikrosekundy byly původně zaokrouhlovány dolů, čímž docházelo k nechtěnému oříznutí posledních datových záznamů. Chyba byla opravena zaokrouhlováním nahoru.

6.3.3 Chybné porovnání datumů

Další chybou bylo špatné porovnávání datumů. Datumy byly původně porovnávány ve formátu string, což je seřadí abecedně, a nikoliv podle času, což je samozřejmě nežádoucí. Tato chyba byla jednoduše opravena převedením datumů do formátu timeval a následným porovnáním těchto formátů.

6.3.4 Ošetření vstupu s žádnými daty

Při testování na reálné databázi systém padal v případě, že vybraný senzor neobsahoval žádná data. Chyba byla opravena, aby systém nespadnul, ale pouze zobrazil prázdný graf a vypsal hlášku: `Žádná data!`.

6.3.5 Omezení obnovovacího intervalu

Další chybou, která nastávala u rozsáhlejších datových sad, bylo nastavení obnovovacího intervalu na interval menší než jedna sekunda. V módu start-now se pak nové načtení dat spustilo dříve, než vůbec došlo k dokončení načítání dat předchozích a proto se žádná data nikdy nenačetla. Tato chyba byla opravena nastavením obnovovacího intervalu na hodnotu vyšší než jedna sekunda.

6.3.6 Problém se synchronizací

Při práci s rozsáhlejšími datovými sadami nastával problém se synchronizací. Pokud uživatel spustil nějaký další proces v průběhu stále běžícího načítání dat, systém buď spadnul nebo nastavil některé parametry špatně. Tento problém byl vyřešen přidáním úseku čekání na dokončení některých procesů a řídící proměnnou `isLoading`, která informuje ostatní procesy, že stále probíhá načítání dat.

6.3.7 Chyba blokování tlačítek

Při usability testu byla testery odhaleno chybné blokování tlačítek pro úpravu intervalu. Bylo totiž možné počáteční či koncový čas intervalu posunout až mimo rozmezí v datech a tlačítko se v tomto případě nezakázalo a bylo tedy možné interval ještě více posouvat mimo rozmezí. Problém byl vyřešen blokováním tlačítek nejen v případě, že se interval dostane na stejné hodnoty jako rozmezí dat, ale i když se dostane mimo tyto hodnoty.

6.4 Vylepšení

V této kapitole jsou podrobně popsané všechny vylepšení, které byly navrženy při testování a následně naimplementovány.

6.4.1 Blokování tlačítek

Při mém osobním testování mě napadlo přidat blokování tlačítek v případě, že se stanou irelevantní. Například ve chvíli, kdy se dostaneme s časovým intervalem na samý okraj dat a nemá smysl tedy již dále interval posouvat nebo zvětšovat v tomto směru. Daná tlačítka v tom případě budou blokována. 5.7.8

6.4.2 Úprava parametrů přímo v UI

Při usability testu bylo testery navrženo přidání možnosti úpravy parametrů pro posouvání dat v grafu přímo do uživatelského rozhraní. Tato možnost byla naimplementována (viz. 4.10.9).

6.4.3 Primární senzor do konfigurace

Další vylepšení navrhnuté testery je přidání implicitního senzoru do konfigurace. Tato možnost velice urychlí zobrazování dat v případě, že chce uživatel velice často zobrazovat data z jednoho konkrétního senzoru. Do konfigurace byly přidány tři parametry pro definování implicitní části dat, jednotky a konkrétního senzoru (viz. 4.5).

6.4.4 Úprava přibližovacího poměru

Poslední testery navrhnuté vylepšení byla změna způsobu počítání přibližovacího poměru. Prvotně se nový poměr počítal z aktuálního zobrazení a vzhledem k tomu, že je poměr procentuální, docházelo k tomu, že se přibližovalo o jiný poměr než oddalovalo. Přepočítání bylo předěláno tak, aby se počítalo z úvodního rozmezí dat a tím se problém vyřešil a přibližování a oddalování je již konzistentní.

6.5 Závěr z testování

Různé fáze testování odhalily několik závažných chyb, které byly všechny následně opraveny a opět řádně otestovány. Během testování se přišlo také na některá vylepšení, která také byla všechna implementována a otestována. Byla testována i uživatelská přívětivost, která se ukázala jako dobrá. Žádný z testerů neměl s funkčností ani s pochopením problémy.

Kapitola 7
Závěr

Cílem diplomové práce byla implementace modulu do software SigProc 4.1, který vlastní společnost Safibra s.r.o. a který slouží pro zobrazování dat z vláknově optických senzorů, které měří průhyb vlakových kolejí a pomáhají upozorňovat na neobvyklé stavy kolejí, které jsou potřeba opravit, aby se předcházelo potenciálním neštěstím.

Předchozí verze programu SigProc umožňovala zobrazování dat v reálném čase, která přímo přicházejí ze senzorů, ale nebylo možné zobrazit data zpětně z databáze. Nový modul dokáže zobrazit jakákoliv data z databáze a také umožňuje úpravy zobrazení dat v grafu uživatelem jako je například přibližování, oddalování, posouvání nebo filtrování pro analýzu detailů signálu.

V kapitole 2 byla popsána teorie optických vláken, optických senzorů, braggovských mřížek a nosníků senzorů. Dále je zde vysvětlen výpočet průhybové křivky nosníků a algoritmus pro přepočet dat ze senzorů pro uložení do databáze v systému SigProc.

V kapitole 3 byla provedena rešerše technik pro zobrazování streamovaných dat, problémů s tím spjatých a nejčastějších chyb, které se při vizualizaci streamovaných dat dělají. Na základě této rešerše byla zvolena vhodná technika pro vizualizaci dat v navrhovaném modulu aplikace.

V kapitole 4 byla sestavena kompletní analýza, potřebná pro vývoj nového modulu. Byla popsána současná funkčnost systému SigProc, sepsány všechny funkční i nefunkční požadavky, popsány konfigurační parametry. Také byla navržena vhodná architektura, uživatelské rozhraní a definovány všechny případy užití.

V kapitole 5 byla popsána výsledná implementace nového modulu. Byla sepsána struktura aplikace, konstanty, proměnné a řídící logika. Také byly přidány některé screenshoty obrazovky a zajímavé kusy kódu. Dále zde byly vytvořeny diagramy metod a detailně popsáno uživatelské rozhraní. Na závěr byla definována veškerá omezení, které s sebou nový modul nese.

V kapitole 6 bylo popsáno průběžné i závěrečné testování. Proběhlo testování v několika fázích, nejdříve autorem a poté i dalšími testery. Během testování bylo odhaleno několik chyb, které byly všechny následně opraveny. Testování přineslo také nápady na další vylepšení, která byla následně naimplementována.

Implementace byla zaměřena nejen na splnění všech požadavků firmy Safibra s.r.o. na funkčnost, ale i na uživatelskou přívětivost a intuitivnost. Všechny funkční i nefunkční požadavky se povedlo splnit a i uživatelská přívětivost se ukázala jako dobrá.

Nový modul do systému SigProc byl nasazen do reálného použití a předpokládá se jeho aplikační používání v následujících letech a tak pomůže předcházet potenciálním neštěstím na kolejích.

7.1 Použitý software a technologie:

- Eclipse – vývojové prostředí na vývoj aplikace v jazyce C++.
- TeXstudio – software pro psaní prací ve formátu LaTeX.

- Enterprise Architect – software pro vytváření softwarové dokumentace (byl využit pro tvorbu diagramů).
- JabRef – software pro správu citací a referencí.
- FreeMind – software pro vytváření myšlenkových map.
- Git – webový software pro správu verzí.
- GoPro kamera – použitá na natočení usability testu.

Kapitola 8
Literatura

[1] Safibra s.r.o. http://www.safibra.cz/.

[2] Chart dos and don'ts. https://www.eea.europa.eu/data-and-maps/daviz/learn-more/chart-dos-and-donts, 2016.

[3] Web browsers statistics. https://www.w3schools.com/browsers, May 2018.

[4] James Ahrens, Kristi Brislawn, Ken Martin, Berk Geveci, C Charles Law, and Michael Papka. Large-scale data visualization using parallel data streaming. *IEEE Computer graphics and Applications*, 21(4):34–41, 2001.

[5] Larry Alton. 4 potential problems with data visualization.

[6] Mihael Ankerst. Visual data mining with pixel-oriented visualization techniques. In *Proceedings of the ACM SIGKDD Workshop on Visual Data Mining*, 2001.

[7] Frédéric Blanchard, Michel Herbin, and Laurent Lucas. A new pixel-oriented visualization technique through color image. *Information Visualization*, 4(4):257–265, 2005.

[8] Lee Byron and Martin Wattenberg. Stacked graphs–geometry & aesthetics. *IEEE transactions on visualization and computer graphics*, 14(6), 2008.

[9] C. Chen. Top 10 unsolved information visualization problems. *IEEE Computer Graphics and Applications*, 25(4):12–16, July 2005.

[10] Weiwei Cui, Yingcai Wu, Shixia Liu, Furu Wei, Michelle X Zhou, and Huamin Qu. Context preserving dynamic word cloud visualization. In *Visualization Symposium (PacificVis), 2010 IEEE Pacific*, pages 121–128. IEEE, 2010.

[11] CSc. Doc. ing. Jiří Michalec. *Pružnost a pevnost 1*. ČVUT, 2010.

[12] David Gajdušek. Streaming a komunikace. 2013.

[13] Florian Heimerl, Steffen Lohmann, Simon Lange, and Thomas Ertl. Word cloud explorer: Text analytics based on word clouds. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 1833–1842. IEEE, 2014.

[14] C. Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, July 2004.

[15] Daniel A Keim. Designing pixel-oriented visualization techniques: Theory and applications. *IEEE Transactions on visualization and computer graphics*, 6(1):59–78, 2000.

[16] M. Krstajic and D. A. Keim. Visualization of streaming data: Observing change and context in information visualization techniques. pages 41–47, Oct 2013.

[17] Bc. Stanislav Novák. Analyzátor signálu optických vláknových senzorových sítí. Master's thesis, České vysoké učení technické v Praze, 2016.

[18] Kristin Potter, Hans Hagen, Andreas Kerren, and Peter Dannenmann. Methods for presenting statistical information: The box plot. *Visualization of large and unstructured data sets*, 4:97–106, 2006.

[19] Hannes Reijner and Panopticon Software. The development of the horizon graph.

[20] Nisnith Sharma. 77 most common data visualization mistakes. 2015.

[21] Ben Shneiderman and Martin Wattenberg. Ordered treemap layouts. In *Information Visualization, 2001. INFOVIS 2001. IEEE Symposium on*, pages 73–78. IEEE, 2001.

[22] Ross J Simpson, Timothy A Johnson, and Ingrid A Amara. The box-plot: an exploratory analysis graph for biomedical publications. *American heart journal*, 116(6):1663–1665, 1988.

[23] Bc. František Urban. Braggovy mřížky v optických vláknech. Master's thesis, Vysoké učení technické v Brně, 2014.

Kapitola 9
Obsah přiloženého CD

Vzhledem k tomu, že předmětem diplomové práce byla implementace nové modulu do již funkčního software, který je vlastněn firmou Safibra s.r.o., nebylo možné na přiložené CD nahrát celý spustitelný software, ale pouze zdrojové kódy implementovaného modulu.

Přiložené CD obsahuje:

- readme.txt – stručný popis obsahu CD,
- složka `src` – zdrojové kódy implementace,
- složka `thesis` – text práce ve formátu PDF,
- složka `tex` – zdrojová forma práce ve formátu LaTeX.