Internal Report 2010-12 Augustus 2010

Universiteit Leiden

Opleiding Informatica

SignalBrowser

Derk Mus

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

SignalBrowser

Inleiding

Infrawatch is een groot project, waarbij grote infrastructurele bouwwerken (bijvoorbeeld bruggen en tunnels) worden bestudeerd. De *Hollandse Brug* is zo'n groot infrastructureel bouwwerk. Deze brug is de verbinding tussen Flevoland en Noord-Holland, gelegen op de plek waar het Gooimeer over gaat in het IJmeer. De snelweg A6 loopt over deze brug.

In 2007 bleek uit metingen dat de brug niet meer voldeed aan de kwaliteits- en veiligheidseisen. Tijdens de reparatie is een groot aantal sensoren op verschillende locaties op de brug geinstalleerd, om zo de conditie van de brug te meten en grote hoeveelheden data te kunnen verzamelen. Het doel hiervan was om aan de hand van deze data te kunnen evalueren hoe de brug reageert op bepaalde belastingen.

Figuur 1 De Hollandse Brug

Uitleg probleem

De 145 sensoren op de brug producren grote hoeveelheden data. Deze data moet kunnen worden geanalyseerd. Daarom was het nodig hiervoor een applicatie, met een grafische gebruikersinterface,

te ontwikkelen, die de data van de sensoren kan visualiseren in grafieken. Ook moeten bepaalde operaties en filters toegepast kunnen worden op de data.

Een probleem is dat twee sensoren op verschillende locaties op de brug een vrijwel hetzelfde signaal kunnen produceren, alleen met een bepaald verschil in tijd. Het signaal van de ene sensor loopt bijvoorbeeld 2 seconden achter op het signaal van de andere sensor, omdat deze verder op de brug is geplaatst. De applicatie moet dit tijdsverschil kunnen herkennen en de twee grafieken van de sensoren kunnen *alignen* (één van de twee grafieken zo over de ander schuiven dat er een zo groot mogelijk overlap ontstaat).

Theorie

Convolutie [3] is een manier om twee grafieken te kunnen *alignen*. Convolutie is een wiskundige bewerking op twee functies met een nieuwe functie, de convolutie van beide, als resultaat.

$$(u * v)(t) = \int_{-\infty}^{\infty} u(s)v(t-s)\mathrm{d}s$$

Figuur 2 De formule voor de convolutie van twee meetbare functies *u* en *v* op reële getallen.

Aanpak

De opgestelde requirements voor de applicatie, genaamd *SignalBrowser*, zijn als volgt:

- Sensoren kunnen kiezen in pop up window met tabel (die een 2d weergave brug van de brug vormt) [*prioriteit: 1*]
- “Onbeperkt” aantal sensoren onder elkaar (max 30) [*1*]
- CSV bestanden kunnen inlezen (max 50 MB) [*1*]
- Operaties op de afzonderlijke sensor data:
    - Inzoomen / Uitzoomen [*1*]
    - Slepen [*1*]
    - Filters
        - Down-samplen [*2*]
        - Eenvoudig andere filters toe kunnen voegen [*1*]
- Operaties op meerdere sensoren
    - Auto alignment [*1*]
- Huidige toestand in programma opslaan in eigen formaat (csv) [*1*]
    - Opslaan in xml formaat [*3*]
- Aanpassingen aan sensor data terugschrijven naar bestand [*1*]

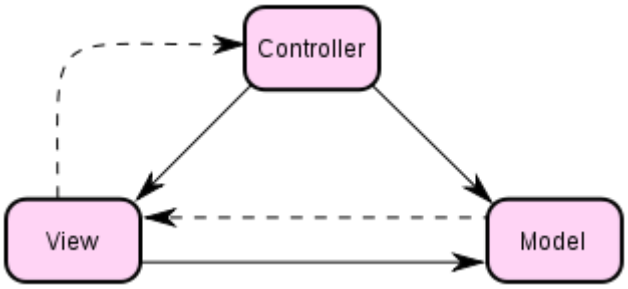
Met deze requirements in de applicatie gemaakt. Belangrijk was, gezien de beperkte tijd, dat de architectuur zo is dat er later gemakkelijk extra functionaliteit (bijvoorbeeld filters) kan worden toegevoegd.

Implementatie

Voor de implementatie is gekozen voor de programmeertaal Java. Voor de grafische gebruikersinterface is gekozen voor de Swing library [1]. Om de grafieken te tekenen is gekozen voor de JFreeChart library [2].

Architectuur

De architectuur van de applicatie volgt het Model-view-controller (MVC) patroon. Hierin wordt het ontwerp van de applicatie opgedeelt in drie eenheden met verschillende verantwoordelijkheden: domein logica (model), presentatie (view) en applicatie logica (controller).



Figuur 3 Model View Controller architectuur, de doorgetrokken lijnen geven een directe associatie aan (de controller kent de view en model, andersom niet). Gestippelde lijnen geven een indirecte associatie aan via het observer pattern [5].

Model

Alle sensor data wordt gerepresenteerd in het model. De classes in de *signalbrowser.model.data* package kunnen data bestanden uitlezen en ook gewijzigde data weer terugschrijven naar een bestand. Op dit moment is er ondersteuning voor de volgende formaten:

- **XML**
- **CSV**
- **SiBr** – Dit formaat kan de huidige toestand in de *SignalBrowser* applicatie opslaan.

Elke sensor met bijbehorende data wordt gerepresenteerd door een *Sensor* object. De *view* kan een sensor object weergeven in een grafiek.

Ook de verschillende filters die op sensor data kunnen worden toegepast zijn onderdeel van de *signalbrowser.model* package. De filters die zijn ontwikkeld:

- **Aligning** – Om twee grafieken te kunnen alignen. Hiervoor wordt de convolutie methode gebruikt. De convolutie array heeft een grootte van $2n - 1$ voor sensoren met n gemeten waardes. Zo kan elke mogelijke overlap worden berekend.
- **Cropping** – Om de data tussen een bepaalde begin- en eindtijd te behouden en de rest weg te gooien.
- **Downsampling** – Om de sample rate (in aantal samples per seconde) te kunnen verlagen. Sample rates kunnen alleen zo worden verlaagd dat het tijdsinterval tussen elke twee opeenvolgende samples hetzelfde is.

View

De view maakt gebruik van de Swing API voor de GUI en de JfreeChart API om grafieken te tekenen en zo het signaal van de sensoren te representeren. In de *signalbrowser.view* package zijn classes te vinden voor de windows, dialogs en panels (onderdeel van een window of dialog). Verschillende objecten van de view bieden andere objecten de kans om events van de GUI (een klik op een button of menu item bijvoorbeeld) te ontvangen door zich te registreren als *listener*. De mogelijkheden hiervoor zijn:

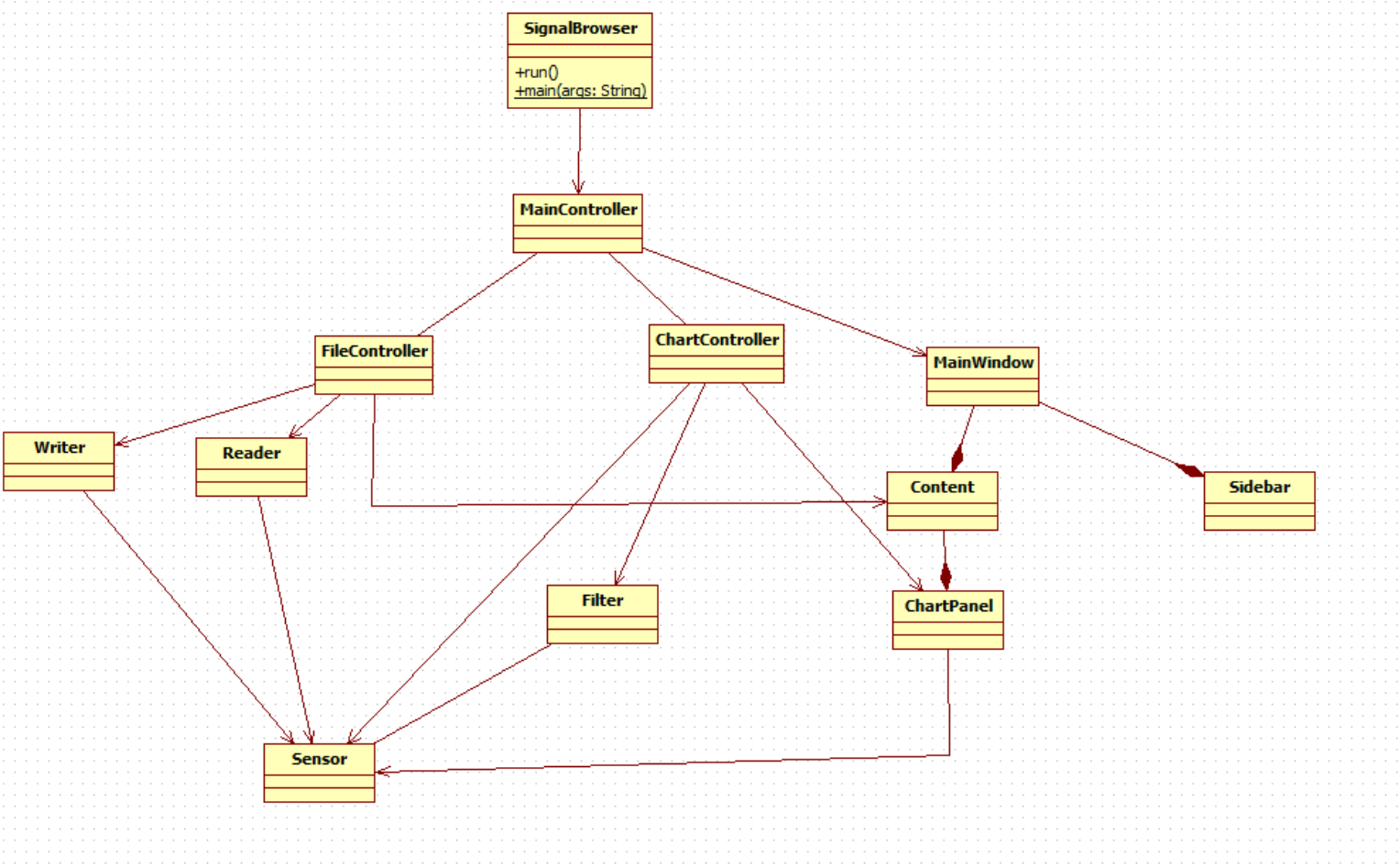
- **ChartPanelListener** – Ontvangt events als het bereik of domein van een grafiek in een panel veranderen of als er op de knop “Sluiten” wordt geklikt.
- **FileListener** – Ontvangt een event als er op een item in het “File” menu is geklikt.
- **FilterListener** – Ontvangt een event als er op een item in het “Filter” menu is geklikt.
- **SensorSelectDialogListener** – Ontvangt een event als er een sensor wordt aangevinkt/afgevinkt in de tabel met alle sensoren.

Controller

De controller handelt bepaalde events af. Een controller kan zichzelf bij een object, dat onderdeel is van de view, registreren om zo events in de GUI te ontvangen. De vier controllers in *Signalbrowser* zijn:

- **MainController** – Deze controller regelt de initialisatie bij de start van de applicatie en regelt het toevoegen en verwijderen van grafieken via een apart window.
- **FileController** – Deze controller regelt acties als bestanden openen, sluiten en opslaan.
- **ChartController** – Deze controller handelt de acties af die uitgevoerd moeten worden als een gebruiker een filter wil toepassen, als de gebruiker de pijltjestoetsen gebruikt om door een grafiek te navigeren en zorgt ervoor dat in alle grafieken in principe hetzelfde domein zichtbaar is.

- **ErrorHandler** – Hier worden errors afgehandeld die niet op een andere plaats konden



Figuur 4 Class diagram met de belangrijkste classes van de SignalBrowser applicatie

worden afgehandeld.

Unit tests

Voor unit tests is de map *tests* beschikbaar. De package structuur hierin is dezelfde als in de map *src.*

Er is één simpele test aanwezig, *signalbrowser.model.filter.AligningTest*, om de theorie van het aligning filter te testen. De test voegt vijf willekeurige metingen aan twee sensoren (*f* en *g*) toe. Vervolgens alignt de test sensor *g* met sensor *f* en andersom en kijkt of het resultaat (aantal millisecondes dat respectievelijk de grafieken van *f* en *g* moeten worden verschoven) is zoals verwacht.

Functionaliteit toevoegen

Filters

Een niew filter moet de interface in *signalbrowser.model.filter.Filter* implementeren. Deze interface ziet er als volgt uit:

```
public interface Filter {
      public void filter(Sensor sensor) throws Exception;
```

```
}
```

In deze functie kunnen operaties op de filter data worden uitgevoerd.

Bestandsformaten

Naast ondersteuning voor de formaten csv, xml en sibr kan ondersteuning worden toegevoegd voor andere formaten. Om een bestand in een ander formaat te kunnen lezen moet de *Reader* interface worden geimplementeerd:

```
public interface Reader {
	public void readAll() throws Exception;
	public Map<Integer, Sensor> getSensors();
}
```

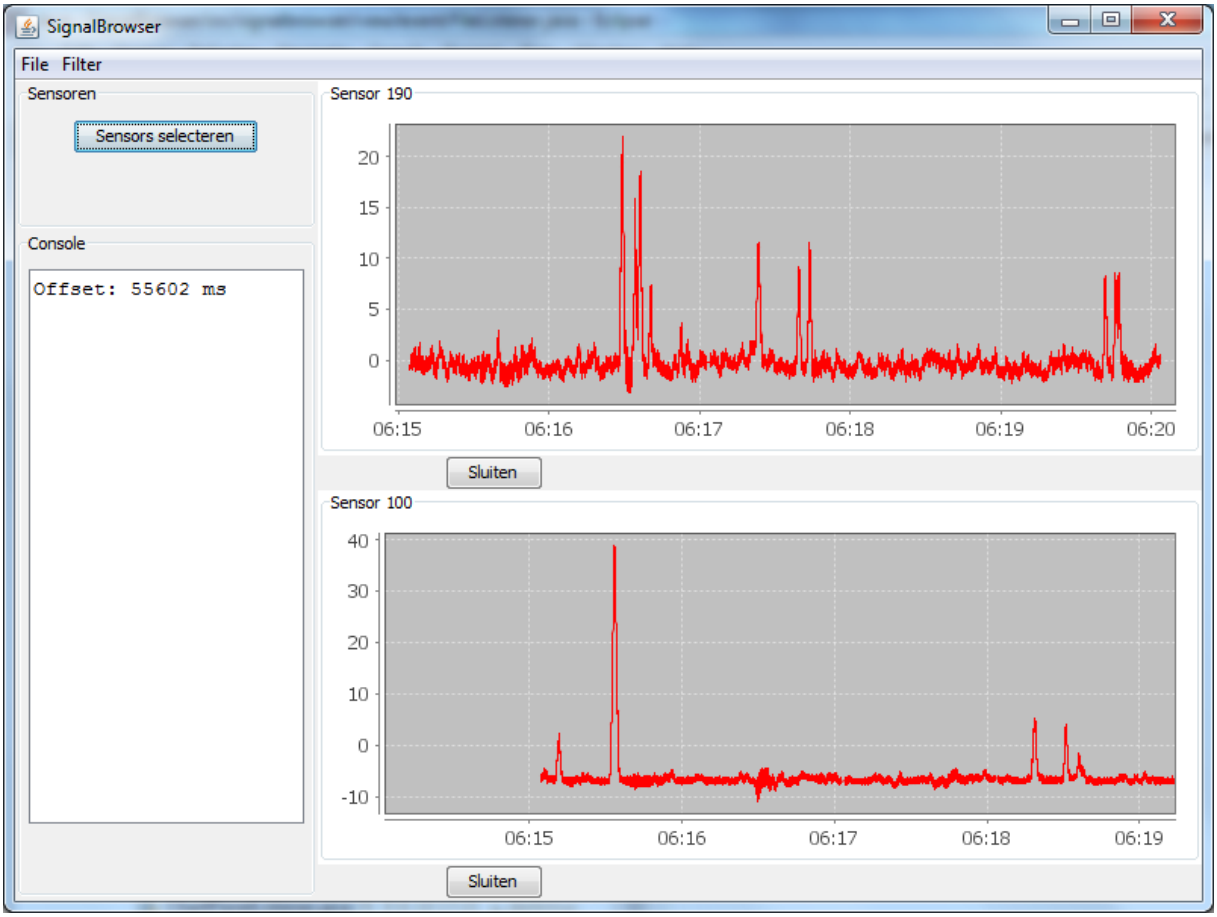
Om de gewijzigde data terug te kunnen schrijven in een bepaald formaat moet de *Writer* interface worden geimplementeerd:

```
public interface Writer {
	public void setSensors(Map<Integer, Sensor> sensors);
	public void write() throws IOException;
	public void close() throws IOException;
}
```

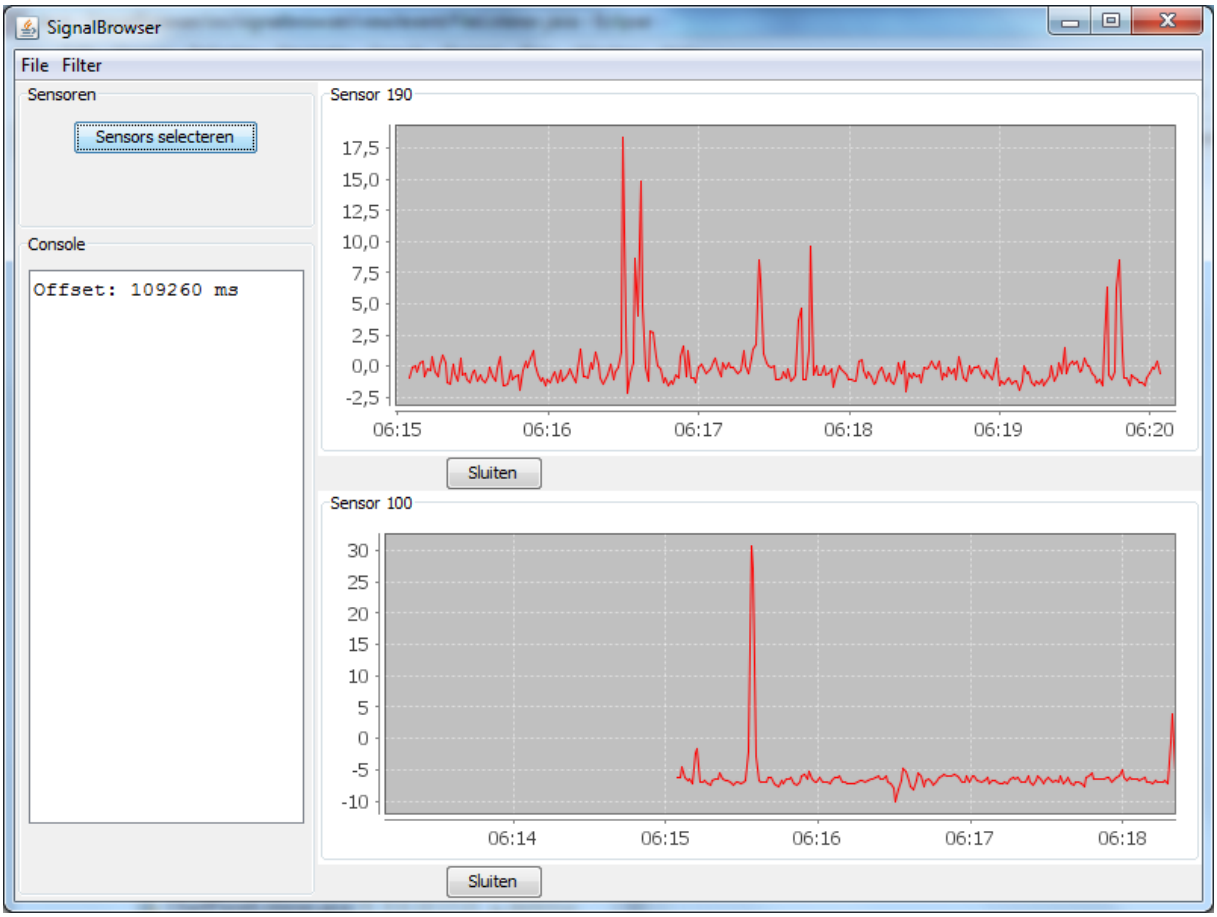
Experimenten

Door de beschikbare tijd zijn weinig experimenten met de sensor data van de *Hollandse Brug* uitgevoerd. Een experiment dat is uitgevoerd is het alignen van de sensors 100 en 190.

Experiment 1



Figuur 5 Resultaat na het alignen van sensor 190 en 100 op data van 5 november 2008. De onderste grafiek is 55.602 ms (55,6 seconden) naar rechts geschoven. De grootste pieken in de twee grafieken staan onder elkaar.



Figuur 6 Zelfde operatie op data van 24 oktober 2008. Hier wordt de onderste grafiek ruim 109 seconden naar rechts verschoven.

Als wordt aangenomen dat de afstand tussen sensor 100 en sensor 190 375 meter bedraagt (de totale lengte van de brug [6]), kan worden uitgerekend wat de snelheid van een voertuig geweest moet zijn geweest als deze trillingen in beide sensoren heeft veroorzaakt.

5 november 2008

375 m / 55,602 s = 6,74 m/s ≈ 24,3 km/h

24 oktober 2008

375 m / 109,260 s = 3,43 m/s ≈ 12,4 km/h

Experiment 2

Als tweede experiment is gekeken naar het alignen van de volgende sensoren: 100-111, 100-137, 100-153, 100-174, 100-185, en 100-190. De sensoren 100, 111, 137, 153, 174, 185 en 190 bevinden zich ieder aan de onderkant van een opeenvolgende steunpilaar van de brug.

Tabel 1 Resultaten experiment (in milliseconden)

	100
111	0
137	-298772
153	301

174 -298772
185 -253020
190 -55601

Conclusie

De *SignalBrowser* applicatie voldoet aan de oorspronkelijke requirements. Grote databestanden in zowel csv als xml formaat kunnen worden ingelezen. Ook is er een eigen formaat (SiBr) ontwikkeld om hierin de huidige toestand van de applicatie te kunnen opslaan. Belangrijk is dat de data goed gevisualiseerd kan worden, dit is mogelijk door het in- en uitzoomen en slepen op de grafieken. Daarnaast is het kiezen van de sensoren overzichtelijk door de tabel, waarin bij elke sensor staat aangegeven wat voor type het is.

Simpele tests met het aligning filter hebben uitgewezen dat deze goed werkt. Ook het downsampling filter en het cropping filter werken goed. Voor toekomstig werk zouden deze filters nog verder kunnen worden geoptimaliseerd om de werking van het programma te versnellen. Voor het aligning filter zou bijvoorbeeld kunnen worden overwogen om een convolutie array met een grootte kleiner dan *2n – 1* te gebruiken. Voor downsampling operaties kan worden overwogen om in plaats van samples die niet meer nodig zijn één voor één weg te gooien, een nieuwe array te maken waar alleen de samples die behouden blijven in worden geplaatst.

Uit experiment 1 is gebleken dat het niet plausibel is dat de gealignde trilling van sensor 100 en sensor 190 door hetzelfde voertuig wordt veroorzaakt. Het voertuig zou daarvoor een snelheid gehad moeten hebben van ongeveer 24 km/h. Dit lijkt niet waarschijnlijk, aangezien er een autosnelweg over de brug loopt. Een mogelijkheid zou kunnen zijn dat er een voertuig dat wegwerkzaamheden uitvoerde met zo'n lage snelheid over de brug reed. Het experiment op de data van 9 november 2008 is minder betrouwbaar, omdat hierbij niet duidelijk twee pieken onder elkaar worden gezet.

Uit experiment twee is gebleken dat achtereenvolgens sensor 111, 137, 153 en 174 alignen met sensor 100 geen lineair resultaat oplevert.

Door de architectuur is het eenvoudig om nieuwe filters aan de applicatie toe te voegen en zo de functionaliteit nog verder uit te breiden.

Referenties

[1] Java Swing. *Oracle, http://download.oracle.com/docs/cd/E17476_01/javase/1.5.0/docs/api/javax/swing/package-summary.html*.

[2] JFreeChart. *JFree, http://www.jfree.org/jfreechart/*.

[3] Convolution – Wikipedia, *http://en.wikipedia.org/wiki/Convolution*

[4] Arno Knobbe, Hendrik Blockeel, Arne Koopman, Toon Calders, Bas Obladen, Carlos Bosma, Hessel Galenkamp, Eddy Koenders, and Joost Kok: InfraWatch: Data Management of Large Systems for Monitoring Infrastructural Performance

[5] Observer pattern – Wikipedia, *http://en.wikipedia.org/wiki/Observer_pattern*

[6] Hollandse Brug – Wikipedia, *http://nl.wikipedia.org/wiki/Hollandse_Brug*