PROJEKTARBEIT

Dynamische Visualisierung von Messdaten mithilfe von aktuellen Webframeworks

Ludwig-Maximilians-Universität München
Fakultät: Informatik
Studiengang: Informatik (Diplom)

Autor: Christian Müller
Betreuer: Christian Hilgers, ZAE Bayern
Prüfer: Prof. François Bry

Ausgabedatum: 01.11.2011
Abgabedatum: 29.02.2012

Abstrakt

Im Rahmen dieser Projektarbeit werden aktuelle Webframeworks auf ihre Tauglichkeit zur dynamischen Visualisierung von Messdaten analysiert. Anschließend erfolgt die Implementierung der dynamischen Messdatenvisualisierung. Großer Wert wird auf die Sondierung der verschiedenen Webframeworks gelegt. Die daraus gewonnen Erkenntnisse bringen ein geeignetes Webframework für die Eingliederung in eine existierende Webapplikation zur Messdatenvisualisierung hervor. Anhand eines Anforderungskataloges wird das ausgewählte Webframework dygraphs optimiert und erweitert. Nach der Ausarbeitung einer passenden Strategie zur Messdatenvorhaltung wird das Webframework in die bestehenden Webapplikation integriert.

Within this project thesis, current web frameworks are found and evaluated on their suitability for the dynamic visualization of measurement data. Afterwards the dynamic measurement data visualization is implemented. Great emphasis is placed on exploring the various web frameworks. The gained knowledge leads to an appropriate web framework dygraphs for the integration into an existing web application to visualize measurement data. On the basis of the requirements the selected web framework is optimized and expanded. After developing an appropriate strategy for data provisioning, the web framework will be integrated in the existing web application.

Inhaltsverzeichnis

Abstrakt I

Inhaltsverzeichnis II

1 Einleitung 1

1.1 Motivation 1

1.2 Beispiel eines Energiesystems 1

1.3 Zielsetzung 3

2 Stand der Technik 3

3 Vorabüberlegungen und Auswahlkriterien 4

3.1 Ist-Zustand des bestehenden Systems 4

3.2 Anforderungskatalog 5

3.2.1 Darstellungs- bzw. Diagrammoptionen 5

3.2.2 Dynamische Optionen 5

3.2.3 Nichtfunktionale Anforderungen 6

3.3 Kriterien zur Auswahl eines geeigneten Webframeworks 6

4 Webframeworks für die Visualisierung 7

4.1 Frameworks mit Grundfunktionalität 7

4.1.1 Ample SDK 8

4.1.2 dhtmlxChart 8

4.1.3 dojox.charting 8

4.1.4 Ext JS Charting 8

4.1.5 YUI Library 9

4.1.6 Google Chart Tools 9

4.1.7 jQuery Visualize Plugin 9

4.2 Frameworks mit erweiterter Funktionalität 9

4.2.1 Highcharts 10

4.2.2 jqPlot 11

4.2.3 Flot 12
4.2.4 Flotr 13
4.2.5 dygraphs 13
4.3 Beurteilung der Webframeworks 15
5 Implementierung und Einbindung in das System 19
5.1 Softwarearchitektur und Konzept 20
5.2 Konfiguration von mitgelieferten Funktionalitäten 20
5.3 Erweiterung um Funktionalitäten 22
5.4 Einbindung in die Server-Client-Struktur 24
6 Fazit und Ausblick 28
Abbildungsverzeichnis VI
Literaturverzeichnis VII
Erklärung IX
Danksagung X

1 Einleitung

1.1 Motivation

Die Abteilung „Technik für Energiesysteme und Erneuerbare Energien" des Bayerischen Zentrums für Angewandte Energieforschung e.V. (ZAE Bayern) in Garching forscht im Bereich der Energiespeicherung sowie Energieoptimierung von Gebäuden und Prozessen. So ist das ZAE Bayern beispielsweise auf dem Gebiet der offenen Sorptionssysteme mit festen und flüssigen Sorbentien, die zum Heizen, Kühlen und Trocknen verwendet werden, sowie der thermischen Nutzung von Solarenergie tätig. In diesem Rahmen arbeitet das ZAE Bayern an der Entwicklung von innovativen Komponenten und an der Planung energietechnischer Referenzanlagen.

Um genaue Kenntnis über physikalische Vorgänge in diesen Referenzanlagen oder Versuchsaufbauten zu erhalten, ist neben der eigentlichen Datenaufzeichnung eine geeignete Visualisierung und Auswertung erforderlich.

Aus diesem Grund wurde am ZAE Bayern eine Visualisierungs- und Auswertesoftware entwickelt, welche speziell auf die Problemstellungen des Instituts zugeschnitten ist. Die Visualisierung der Daten ist als statische Generierung einer Grafikdatei auf dem Datenbankserver realisiert, welche nach der Erstellung an die Weboberfläche des Client-Systems übertragen und angezeigt wird. Die Umgestaltung dieses Prinzips hin zu einer dynamischen Generierung des Graphen rein auf der Client-Seite stellt die Herausforderung dieser Arbeit dar. Die Vorteile einer dynamischen Visualisierung, wie beispielsweise der interaktive Zugriff auf einzelne Messwerte, sollen in einem Anforderungskatalog gesammelt und anhand diesem ein geeignetes Webframework für die Implementierung gefunden werden.

1.2 Beispiel eines Energiesystems[1]

Um einen Einblick über die zu vermessenden Systeme am ZAE Bayern zu vermitteln, wird beispielhaft eine größere Referenzanlage beschrieben, die im Projekt "Solare Nahwärme mit saisonaler Speicherung am Ackermannbogen" in München realisiert wurde. Das Ziel des Energiesystems ist die solarunterstützte

[1] Auszug aus dem Medview2-Handbuch

Wärmeversorgung von acht kleineren Häusern und fünf Wohnblocks mit 333 Wohnungen.

Um die benötigte Wärmemenge aus dem angeschlossenen Fernwärmenetz für die Gebäude zu senken, wurde eine Solarkollektorfläche von 3000m² und ein unterirdischer Erdbeckenspeicher mit 5.600m³ Wasser zur Langzeitspeicherung der Wärme installiert. Ziel dieser Anlage ist es, den sonst ungenutzten Wärmeüberschuss der Solarkollektoren im Sommer in dem Erdbeckenspeicher zu speichern. In der Übergangszeit zum Winter, wenn die Solarkollektoren nicht den benötigten Wärmebedarf liefern, wird die gespeicherte Energie aus dem Wasserspeicher entnommen. Simulationen des Anlagenverhaltens ergaben Einsparungen der konventionellen Energie um bis zu 50%.

Die messtechnische Erfassung der Anlage unterteilt sich in drei Teilbereiche mit insgesamt über 3000 Messkanälen. Der erste Bereich dient der Erfassung der Energiezentrale, des Langzeitspeichers und der Wetterdaten über eine mit Ethernet angebundene Wetterstation. Hierbei werden ca. 160 einzelne Messkanäle, je nach Dynamik der Messwerte, mit unterschiedlichen Scan-Geschwindigkeiten von 10 bis 600 Sekunden aufgenommen. Der zweite Bereich umfasst die bidirektionale Kommunikation zwischen dem ZAE Messrechner und der SPS (Speicherprogrammierbare Steuerung) [1]. Die SPS übernimmt die Regelung und Steuerung der Anlage und stellt eine Datenpunktliste[2] mit ca. 250 Mess- und Statuskanälen bereit. Der dritte und weitaus größte Teilbereich umfasst die Vermessung der Übergabestationen der 333 Wohnungen über Wärmemengenzähler mit insgesamt 2664 Messkanälen. Hierzu kommen ca. 100 zusätzliche Datenpunkte der kleineren Mehrfamilienhäuser. Wie bei allen solarthermischen Energiesystemen bestehen die Messkanäle überwiegend aus Temperaturkanälen. Dazu kommt die Erfassung von Meteorologiedaten, Volumenströmen, Drücken, Wärmemengen und Statussignalen.

Um die erforderliche zeitliche Auflösung der Messdaten in Verbindung mit der hohen Kanalzahl im Rahmen der räumlichen Verteilung realisieren zu können, wurde ein über IP verbundenes System aus 6 PCs und einem Hauptrechner realisiert, auf denen über 30 verschiedene Messprogramme ausgeführt werden. Die Systeme

[2] ASCII Liste mit allen Mess- bzw. Statuswerte die von der SPS gehalten werden.

laufen unter Linux und sind über den Hauptrechner in der Heizzentrale per ISDN erreichbar.

1.3 Zielsetzung

Im Rahmen dieser Arbeit soll die am ZAE Bayern entwickelte, datenbankgestütze Online-Visualisierung von Messdaten um eine dynamische Visualisierung erweitert werden. Hierzu soll ein geeignetes Webframework gefunden werden, welches den gesteckten Anforderungen gerecht wird. Die Einarbeitung in aktuelle Webframeworks soll einen Überblick verschaffen, welche für den produktiven Einsatz in Frage kommen. Anhand von vorher definierten Auswahlkriterien soll ein geeignetes Framework ausgewählt werden, mit dem die dynamische Visualisierung umgesetzt wird. Die Implementierung in die clientseitige Weboberfläche der Visualisierungssoftware mit besonderem Augenmerk auf die dynamische Datenvorhaltung stellt den Abschluss des Vorhabens dar.

2 Stand der Technik

Webapplikationen gewinnen in den letzten Jahren immer mehr an Bedeutung. Verbesserte Bandbreiten, Browserprogramme und Webtechnologien haben dafür gesorgt, dass selbst komplexe Desktop-Programme, wie Excel oder Photoshop mittlerweile ein voll funktionstüchtiges Online-Pendant haben. Im Zuge dessen hat sich eine Vielzahl an Webframeworks entwickelt, mithilfe derer sich eine Realisierung einer Webapplikation schneller und leichter bewerkstelligen lässt. Webtechnologien wie Flash oder Silverlight haben früh gezeigt, was innerhalb des Browsers möglich ist - HTML5 zeigt neuerdings wohin die Reise geht, wenn man nicht auf proprietäre Plug-Ins setzen will.

Die neuen Möglichkeiten, die sich durch den stetigen Technologiefortschritt ergeben, etablieren nun auch dynamische Infografiken im Internet, die gegenüber statischen Diagrammbildern einen erheblichen Mehrwert durch ihre Interaktions- und Darstellungsfunktionalitäten liefern. Einen weitläufigen Einblick in das Thema gibt ein Artikel der Zeitschrift c’t [2].

Technisch werden die Webgrafiken zumeist noch mit dem proprietären Flash-Plug-In erzeugt, jedoch häufen sich in letzter Zeit mehr und mehr Lösungen, die auf HTML5 mit dem canvas-Objekt oder aber den SVG-Standard setzen. Letztere haben zudem den Vorteil, dass sie deutlich unkomplizierter und ohne zusätzlichen

Installationsaufwand mit mobilen Endgeräten zusammenarbeiten - ein aktuelles, meist mitgeliefertes Browserprogramm genügt.

3 Vorabüberlegungen und Auswahlkriterien

In diesem Kapitel soll analysiert werden, wie ein Webframework bei der dynamischen Generierung eines Datengraphen helfen kann, die bereits bestehende statische Umsetzung zu verbessern und zu erweitern. Hierzu wird zunächst der derzeitige Stand des Systems beschrieben, danach ein Anforderungskatalog erstellt und auf dessen Basis Kriterien eingeführt, die zur Auswahl eines geeigneten Frameworks führen.

3.1 Ist-Zustand des bestehenden Systems

Am ZAE Bayern ist die webbasierte Software Medview2 (ZAE Zentralsystem) entwickelt worden, welche es ermöglicht, Messdaten von Projektanlagen zu visualisieren und auszuwerten. Die Messdaten selbst werden in einer PostgreSQL Datenbank bereitgehalten. Über die Weboberfläche konfiguriert man das Diagramm (Zeitraum, Messkanäle, usw.), welches dann auf dem Datenbankserver generiert und als rein statisches Bild an das Webinterface zurückgeliefert wird.

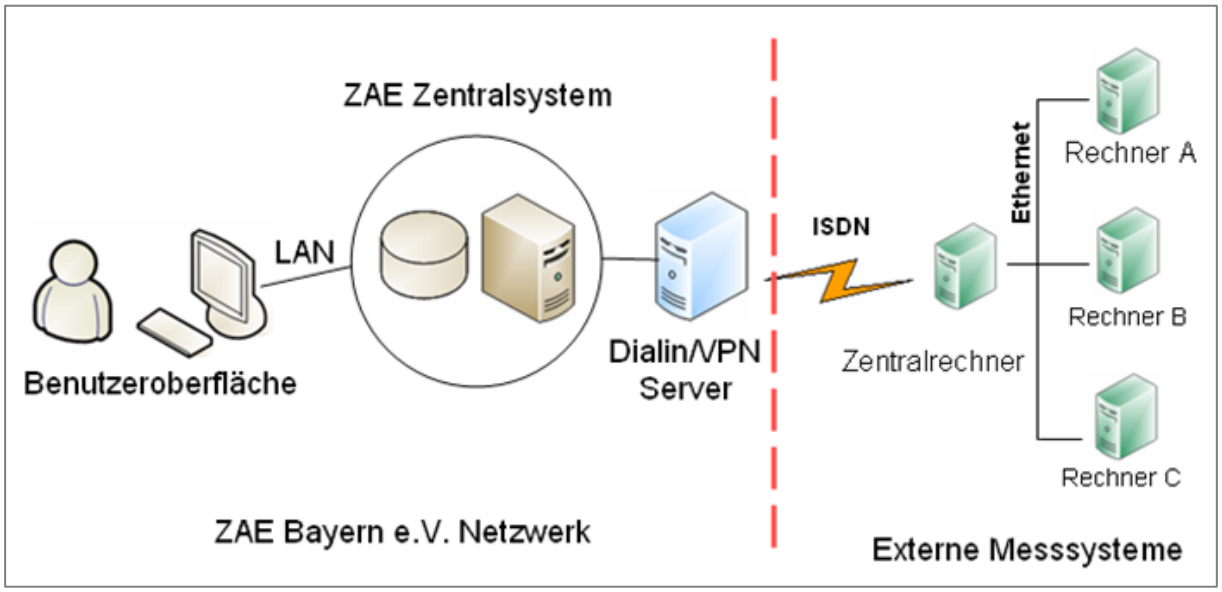

Abbildung 1: Schema des bestehenden Systems

Die erfassten Messdaten werden von externen Rechnern an den Zentralrechner (Medview2) am ZAE Bayern per HTTPS Upload in die Datenbank übertragen (siehe Abbildung 1). Das Backend von Medview2 ist technisch als Webservice (Apache

Axis2) realisiert, so dass man leicht mit verschiedenen Programmen darauf zugreifen kann. Der am ZAE Bayern implementierte SOAP/XML zu JSON Handler macht Axis2 JSON fähig. Dadurch sind mit geringem Aufwand Anfragen über JavaScript möglich, die Messdaten aus der Datenbank liefern. Somit ist die technische Voraussetzung gegeben, sich nicht nur URLs auf statische Diagrammbilder zurückgeben zu lassen, sondern direkt die Messwerte anzufordern, die dann an eine dynamische Visualisierung innerhalb der Weboberfläche weitergereicht werden können.

3.2 Anforderungskatalog

Die Anforderungen die an die dynamische Visualisierung gestellt werden, sind speziell auf die Bedürfnisse des ZAE Bayern zugeschnitten. Das typische Messdatenaufkommen sowie das Messdatenformat legen die groben Rahmenbedingungen fest. Der Anforderungskatalog steckt im Vornherein ab, was die Visualisierung leisten soll und liefert so die Kriterien für die Auswahl eines geeigneten Webframeworks.
Die Anforderungen lauten wie folgt:

3.2.1 Darstellungs- bzw. Diagrammoptionen

- Zwei dimensionales X-Y-Diagramm mit der Unterstützung von Zeitstempeln als X-Werte
- Darstellung einer zweiten Y-Achse
- Verarbeitung von ungültigen Messwerten und Datenlücken
- Messkanäle, die mit unterschiedlichen Zeitstempeln aufgezeichnet wurden, sollen in einem gemeinsamen Plot dargestellt werden
- Skalierung sowohl der Messkanäle (z.B. per Definition eines Skalierungs- und Offset-Faktors) als auch der Zeitachse (Tag, Woche, Monat, usw.)
- Verschiedene Linientypen, Punkttypen und Farben sollen automatisiert vergeben und vom Benutzer verändert werden können
- Interaktive Legende
- Umschaltbare, verschiedene Achsenskalierungen (z.B. linear/logarithmisch)
- Exportfunktion zur Erstellung einer Diagramm-Grafik (z.B. für Berichte)

3.2.2 Dynamische Optionen

- Entfernen von Messpunkten aus dem Diagramm

- Automatische Aktualisierung des Plots bei neuen Werten
- Zoomfunktion – Zoomübersicht als Übersichtsplot anzeigen
- Nachladen bzw. Entfernen von Messkanälen
- Berechnungen über ausgewählte Punkte (z.B. Mittelwertbildung)
- Messung zwischen zwei Punkten (z.B. Abstände)
- Direkter Zugriff auf einzelne Messpunkte (exakte Werte anzeigen)
- Anzeigen mehrerer Diagramme nebeneinander (z.B. für Vergleiche)
- Messwerte markieren und mit Kommentar versehen (z.B. Störung)

3.2.3 Nichtfunktionale Anforderungen

Eine nicht zu unterschätzende Herausforderung stellt die flüssige Interaktion mit den Diagrammen bei typischen Datenmengen dar. Wie in Kapitel 1.2 beschrieben, kommt es gerade hier zu sehr unterschiedlichen Größen an Messdatenaufkommen, die der Visualisierung übergeben werden können, und mit denen sie noch gut bedienbar bleiben muss. Zudem muss das Webframework leicht erweiterbar sein, damit nicht implementierte Anforderungen selbst integriert werden können.

3.3 Kriterien zur Auswahl eines geeigneten Webframeworks

Es werden reine Open-Source-Webframeworks priorisiert. Dies hat mehrerlei Gründe. Zum einen entfallen die Kosten für den Erwerb einer kommerziellen Software. Zudem könnten Änderungen an kommerziellen Frameworks nur über den Hersteller erfolgen und die Folgekosten der Weiterentwicklung erhöhen. Hier bietet ein quelloffenes Softwareprojekt genau die Möglichkeiten zur Spezialisierung an die Bedürfnisse des ZAE Bayern, die für dieses Vorhaben nötig sind. Weiterhin werden ausschließlich JavaScript Webframeworks verglichen. Einerseits, weil die Visualisierung in die bestehende Weboberfläche integriert werden soll, welche selbst bereits auf JavaScript und AJAX setzt. Andererseits, ist die Vielfalt und Zukunftssicherheit der JavaScript Frameworks deutlich höher als bei anderen Web-fremden Sprachen.

Im vorangegangenen Kapitel wurden die Anforderungen an die Visualisierung näher erläutert. Bei der Sondierung der Webframeworks stehen diese Anforderungen konkret im Fokus. So ist es wünschenswert, wenn möglichst viele der Anforderungen von Haus aus schon im Funktionsumfang der Script-Bibliothek verankert sind. All jene Anforderungen, die nicht unmittelbar aus dem Webframework hervorgehen,

sollen aufgrund einer guten Dokumentation des Projekts leicht integrierbar sein. Die Anpassung des Frameworks soll möglichst modular sein, sodass die selbst implementierten Funktionalitäten auch nach einem Update des Webframeworks nicht verloren gehen. Da es sich um ein Open-Source-Projekt handelt, ist es wichtig, dass die Community um das Projekt herum möglichst aktiv ist, damit man regelmäßig mit Bugfixes versorgt wird und seinerseits selbst geschriebene Anpassungen zurück in das Projekt einfließen lassen kann. Außerdem profitiert man so von Verbesserungen anderer User, welche sich mitunter mit den Punkten aus dem Anforderungskatalog decken.

4 Webframeworks für die Visualisierung

Einen guten Überblick über derzeit verfügbare und aktuelle JavaScript-Webframeworks liefert eine Wikipedia-Seite [3], die diese gegenüberstellt und auch insbesondere auf deren Möglichkeit zur Visualisierung[3] eingeht. Anhand dieser Information und einer Internet-Recherche, die weitere Frameworks gefunden hat, haben sich zwei Arten von Webframeworks bezüglich ihrer Visualisierungsfunktionalität herauskristallisiert.
Solche, die eine Basisfunktionalität zum grundsätzlichen Erstellen von Graphen mitbringen und solche, die darüber hinaus auch vielfältige, direkte Interaktions- und Erweiterungsmöglichkeiten mit den Diagrammen erlauben.

4.1 Frameworks mit Grundfunktionalität

Die Webframeworks in diesem Kapitel beherrschen von Haus aus das Erstellen von Diagrammen. Die meisten Zusatzfunktionen müssen vom Programmierer selbst eingepflegt werden. Das liegt meist darin begründet, dass die Webframeworks sich als Komplettpaket zum Erstellen von ganzheitlichen Webapplikationen verstehen. Die Möglichkeit, Diagramme für die Webapplikation zu bauen, nimmt dabei nur einen kleinen, rudimentären Platz im Gesamtkonzept des Frameworks ein. Somit beinhalten diese Visualisierungsbibliotheken, beispielsweise keine Zoommöglichkeit oder die direkte Interaktion mit einzelnen Messpunkten. Da die reinen Basisfunktionaltäten aber nicht dem Anforderungskatalog gerecht werden, wird hier

[3] in der Tabelle unter „Charting & Dashboard“

nur der Vollständigkeit halber kurz auf die Pakete eingegangen und auf Besonderheiten hingewiesen.

4.1.1 Ample SDK

Das Ample SDK versteht sich als JavaScript User Interface Framework für Rich Internet Applications. Es bietet neben den gesuchten Liniendiagrammen unter anderem auch Balkendiagramme, Tortendiagramme und Netzdiagramme. Eine Übersicht findet man auf einer Beispielseite [4] der Ample SDK Homepage.
Das Webframework bringt zwei interaktive Features mit. Zum einen werden die Datenreihen beim Überfahren mit der Maus hervorgehoben, zum anderen kann man durch Click auf eine Datenreihe diese als deaktiviert kennzeichnen.

4.1.2 dhtmlxChart

dhtmlxChart ist eine reine JavaScript Bibliothek zum Erstellen von AJAX-Diagrammen. Über Liniendiagramme hinaus, erlaubt dhtmlxChart das Erstellen von Balken- und Tortendiagrammen. Eine Live Demo ist auf der Homepage [5] des Projekts zu finden.
Interaktive Funktionalitäten sind nicht implementiert, können aber vom Entwickler nachträglich hinzugefügt werden.

4.1.3 dojox.charting

Das Dojo Toolkit ist ein sehr mächtiges und verbreitetes Webframework, um Webapplikationen zu bauen. Das Dojo-Modul dojox.charting bildet die Visualisierungsmöglichkeiten des Frameworks ab. Auch dieses ist sehr umfangreich, und liefert alle nötigen Werkzeuge für die Erstellung einer Vielzahl von Diagrammtypen.
Auf einer ausführlichen Dokumentations-Seite [6] und einer beispielhaften Tutorial-Seite [7] wird erklärt, wie man Graphen erstellt und wie sich interaktive Features, wie Tooltips oder das Hervorheben von Datenreihen, realisieren lassen können.

4.1.4 Ext JS Charting

Ebenso wie das Dojo Toolkit ist Ext JS ein sehr umfangreiches JavaScript-Framework-Paket, welches sich gut dazu eignet, eigenständige Webapplikationen von Grund auf zu programmieren. Mit dem Charting Modul von Ext JS kann man sehr viele, unterschiedliche Diagrammarten erstellen.

Das ein oder andere Feature, wie Tooltips oder Highlightning sind integriert, Funktionen darüber hinaus muss man selbst einpflegen. Die gute Dokumentation [8] und ansprechende Beispiele [9] erleichtern den Einstieg.

4.1.5 YUI Library

Das JavaScript und CSS Framework YUI aus dem Hause Yahoo versteht sich als Komplettpaket zur Erstellung von hoch interaktiven Webapplikationen. Es bietet einige Diagrammtypen und versieht sie mit ein paar wenigen Funktionalitäten, wie z.B. das De- und Aktivieren von Datenreihen und das Anzeigen von Tooltips.
Eine Dokumentations- und Beispielseite [10] zeigt gut dokumentiert, wie man mit YUI Diagramme erstellen kann.

4.1.6 Google Chart Tools

Google Chart Tools ist eine JavaScript Bibliothek aus dem Hause Google, mit der sich schnell und einfach eine große Anzahl von unterschiedlichen Visualisierungen realisieren lassen. Die bereits implementierten interaktiven Möglichkeiten sind Tooltips, Hervorheben und Aktivieren einer Datenreihe, sowie Interaktion mit der Legende.
Sehr gut dokumentierte Anleitungs- und Beispielseiten findet man unter [11].

4.1.7 jQuery Visualize Plugin

Das Visualize Plugin für das jQuery Webframework erstellt gänzlich nicht interaktive Diagramme. Jedoch besteht die Möglichkeit, die Werte, die der Graph anzeigt, direkt innerhalb einer angezeigten Wertetabelle zu verändern.
Eine Beispielseite [12] existiert, die Dokumentation bedarf aber der Verbesserung.

4.2 Frameworks mit erweiterter Funktionalität

Die auf Visualisierung spezialisierten Bibliotheken heben sich deutlich von denen im vorherigen Kapitel vorgestellten Frameworks ab. Da der Schwerpunkt explizit auf der Visualisierung von Infografiken liegt, eignen sich diese Webframeworks im Rahmen dieser Projektarbeit besser als jene mit Grundfunktionalität.
Die Kandidaten haben eine große Auswahl an Features und Schnittstellen, die eine intensive Interaktion mit dem Graphen ermöglicht. Gerade deswegen sind sie für

diese Projektarbeit interessant, da sie bereits viele Anforderungen von Haus aus erfüllen und sich fehlende Funktionen leicht integrieren lassen.
Einige setzten als Basis auf gängige Webframeworks auf, wie jQuery oder Prototype, andere verstehen sich als eigenständige JavaScript Bibliothek.

4.2.1 Highcharts

Highcharts ist eine reine JavaScript Bibliothek zum Erstellen von interaktiven Graphen. Der Fokus liegt bei Highcharts sehr auf dem Look & Feel der Diagramme, welche subjektiv betrachtet, die am optisch Ansprechendsten im Vergleich zu den anderen Kandidaten sind. Es gibt eine Vielzahl an Diagrammtypen zur Auswahl, auch die zu untersuchende Variante eines X-Y-Diagramms mit X als Zeitachse wird vollständig unterstützt. Highcharts setzt bei der Darstellung, anders als die anderen Frameworks mit erweiterter Funktionalität auf den SVG-Standard. Die Dokumentation [13] ist vorbildlich und eine sehr gute Demo-Seite [14] zeigt die bereits implementierten Möglichkeiten von Highcharts.
Die Messdaten werden pro Kanal definiert, die X-Achse wird separat angegeben, wahlweise als Array von Werten oder als Intervall mit Schrittweite. Datenlücken stellen kein Problem dar und es ist auch möglich, Kanäle mit unterschiedlichen Zeitstempeln darzustellen, wenn man die Daten als X-Y-Wertepaare übergibt. Ein Code-Beispiel zeigt eine allgemeine Art, wie die Messwerte zu übergeben sind:

```
chart = new Highcharts.Chart({
series:
    [{
        name: 'Luftdruck',
        pointInterval: 24 * 3600 * 1000,
        pointStart: Date.UTC(2006, 0, 01),
        data: [983.6724,983.6724,983.7744]
    }, {
        name: 'Aussen-Lufttemp_NW',
        pointInterval: 24 * 3600 * 2000,
        pointStart: Date.UTC(2006, 0, 01),
        data: [17,17,17]
    }]
//weitere Optionen...
});
```

Abbildung 2: Highcharts: Beispiel-Code für die Messdatenübergabe

Neben der Möglichkeit die Messdaten per Array direkt im Code anzugeben, ist auch der Import einer CSV-Datei möglich, die dann von einem Parser interpretiert wird. Highcharts gibt es in einer freien Creative Commons Lizenz für nicht-kommerzielle Anwendungen und wird zudem unter einer kommerziellen Lizenz vertrieben. Auch

gerade deshalb wird das Webframework regelmäßig mit Updates versorgt und mit Funktionen erweitert.

4.2.2 jqPlot

Das Plugin jqPlot ist eine Visualisierungsbibliothek für das jQuery Webframework. Es beherbergt die wohl umfangreichsten Anpassungsmöglichkeiten aller hier vorgestellten Frameworks. Neben Linien-, Balken- und Tortendiagrammen gibt es eine Fülle von Darstellungs-arten und -optionen, die auf der Demo-Seite des Projekts [15] eingesehen werden können. Die Dokumentation [16] ist aufgrund der Vielfalt sehr groß und zunächst etwas unübersichtlich, jedoch findet man sich nach etwas Einarbeitungszeit gut darin zurecht und sie hinterlässt keinerlei Zweifel an der Vollständigkeit. Die Diagrammausgabe erfolgt technisch per HTML5 mittels canvas-Objekt.

Die Werteübergabe erfolgt pro Messkanal tupelweise in einem Array. Somit steht jedem Messwert ein Zeitstempel gegenüber, sodass Datenlücken und verschiedene Zeitstempel kein Hindernis darstellen. Jedoch führt man mitunter viele doppelte Zeitstempel mit, falls die Messwerte letztendlich auf den gleichen Messzeitpunkten beruhen. Ein Beispiel veranschaulicht die Messdatenübergabe:

```
var data1 = [['2008-09-30', 4],
             ['2008-10-30', 6.5],
             ['2008-11-30', 5.7]];
var data2 = [['2009-01-30', 8.2],
             ['2009-02-28', 7.6],
             ['2009-03-30', 11.4]];
var plot = $.jqplot('chart', [data1,data2], {
//weitere Optionen...
};
```

Abbildung 3: jqPlot: Beispiel-Code für die Messdatenübergabe

Ein CSV-Parser ist von Haus aus nicht integriert, jedoch gibt es weiterführende Hinweise und Beispiele wie dieser nachträglich zu implementieren ist.

jqPlot ist ein unter MIT-Lizenz frei erhältliches Open Source Projekt. Der Entwickler und die Community sind derzeit aktiv am Projekt beteiligt, was man an den Commits im Projektverzeichnis und den regen Diskussionen in der Mailinglist belegen kann. Somit kann man davon ausgehen, dass man zumindest mittelfristig gesehen mit Updates und Bugfixes versorgt wird.

4.2.3 Flot

Die Visualisierungsbibliothek Flot ist wie jqPlot ein Plugin für das weit verbreitete jQuery Webframework. Es zeichnet sich durch einfache Handhabung und Integration, sowie durch einen ansprechenden Look und zahlreiche Interaktionsmöglichkeiten aus. Das Plugin setzt ebenfalls auf HTML5 mit seinem canvas-Objekt. Auf einer Beispiel-Seite [17] des Projekts werden die vielfältigen Möglichkeiten dargestellt, die Dokumentation [18] ist zwar vollständig, jedoch nicht sehr übersichtlich, da sie lediglich in einer reinen Textdatei vorliegt.

Messwertdaten werden als Array-Tupel übergeben, d.h. zu jedem Y-Wert eines Messkanals muss der Zeitstempel als X-Wert mitgegeben werden. Datenlücken stellen somit kein Problem dar, und auch unterschiedliche Zeitstempel sind damit realisierbar. Auch hier gilt der Hinweis, dass man mitunter sehr viel Overhead mitschleppt, falls bei vielen Messdaten die Zeitstempel nicht verschieden sind. Des Weiteren werden nur reine Zahlenformate als Input unterstützt, d.h. man kann den Zeitstempel nicht direkt übergeben, sondern muss den Zeitstempel vorab in Millisekunden seit dem 1.1.1970 umwandeln. Ein Code-Beispiel zeigt, wie die Messdaten bei Flot zu übergeben sind:

```
$(function () {
    var data = [
        {
            label: 'Luftdruck',
            data: [[40711,983.6724],
                   [40711.00069,983.6724],
                   [40711.00139,983.7744]]
        }, {
            label: 'Aussen-Lufttemp_NW',
            data: [[40711,17],
                   [40711.00069,17],
                   [40711.00139,17]]
    }];
    //weitere Optionen...
    var plot = $.plot(placeholder, data, options);
});
```

Abbildung 4: Flot: Beispiel-Code für die Messdatenübergabe

Flot liegt derzeit in Version 0.7 unter MIT-Lizenz als frei erhältlicher Open Source Code vor. Die Flot-Community ist groß und aktiv, was man sowohl an den Pull Requests im Code-Verzeichnis erkennen kann, als auch an den vielen Einträgen in der Mailinglist. Der letzte Commit vom Entwickler selbst liegt schon etwa ein halbes Jahr zurück, ein Blick auf die Historie zeigt aber, dass dieser stets halbjährlich das Projekt mit Updates versorgt hat.

4.2.4 Flotr

Das Webframework Prototype ist die Grundlage für die Visualisierungsbibliothek Flotr, welche sich, man bemerke die Namensverwandschaft, von Flot inspirieren ließ. Optisch und funktional unterscheidet es sich kaum von Flot, es visualisiert die Graphen ebenfalls mit HTML5 per canvas-Objekt. Die Dokumentation [19] ist deutlich strukturierter und übersichtlicher aufgebaut und auch die Demo-Seite [20] überzeugt. Zudem gibt es eine Playground-Seite [21], auf der man alle Einstellungen live ausprobieren kann.

Messwerte werden auch bei Flotr tupelweise in einem Array übergeben, Datenlücken und unterschiedliche Zeitstempel werden somit unterstützt. Datums- und Zeitangaben müssen ebenfalls vorab konvertiert werden, damit Flotr damit umgehen kann. Ein allgemeines Beispiel veranschaulicht die Werteübergabe in Flotr:

```
var f = Flotr.draw(
    $('container'), [
    {
        data: [[40711,983.6724],
               [40711.00069,983.6724],
               [40711.00139,983.7744]],
        label: "Luftdruck",
    },
    {
        data: [[40711,17],
               [40711.00069,17],
               [40711.00139,17]],
        label: "Aussen-Lufttemp_NW"
    }],
    //weitere Optionen...
);
```

Abbildung 5: Flotr: Beispiel-Code für die Messdatenübergabe

Die zuletzt veröffentlichte Version von Flotr trägt die Versionsnummer 0.2.0 alpha, ist aber trotzdem erstaunlich stabil. Auch Flotr ist Open Source unter MIT-Lizenz, die Community ist überschaubar und der Gründer des Projekts trägt seit längerem nichts mehr zum Fortschritt bei. Jedoch gibt es einen offiziellen Nachfolger, der den einen oder anderen Bugfix bereitstellt, und es gibt eine Abwandlung namens Flotr2, die gänzlich ohne Prototype Framework auskommt. Grundsätzlich sollte man aber nicht allzu viel Support in Zukunft erwarten.

4.2.5 dygraphs

Ein Zitat auf der dygraphs Homepage [22] beschreibt die Visualisierungsbibliothek mit wenigen Worten: „dygraphs is an open source JavaScript library that produces

interactive, zoomable charts of time series. It is designed to display dense data sets and enable users to explore and interpret them.“

Dygraphs setzt also den Fokus ganz bewusst auf zeitbasierte Daten mit hoher Dichte, um den Benutzern zu ermöglichen, diese zu analysieren und zu interpretieren. Balkendiagramme, Tortendiagramme und andere Arten der Visualisierung werden nicht mit angeboten, nur reine X-Y-Graphen stehen im Mittelpunkt. Gerade deshalb ist dygraphs für diese Projektarbeit interessant. Auf der technischen Seite setzt dygraphs wie die meisten anderen Kandidaten auf HTML5 mit dem canvas-Objekt. Eine Galerie [23] zeigt sehr anschaulich und nachvollziehbar einige Möglichkeiten von dygraphs, ein Testverzeichnis [24] vervollständigt die Beispielauswahl. Die Dokumentation ist gut gegliedert in die Menüstruktur der Homepage eingebunden.

Dygraphs erwartet die anzuzeigenden Daten entweder als Array oder als CSV-Datei. Anders als die übrigen Kandidaten werden die Daten nicht als einzelnes Array pro Messkanal übergeben, sondern in einer gemeinsamen Array-Tabelle. So gibt die erste „Spalte“ des multidimensionalen Arrays den Zeitstempel wieder, während die übrigen „Spalten“ die Y-Werte des jeweiligen Messkanals widerspiegeln. Die Zeitstempel können dabei entweder als Zahlenwert, oder direkt als JavaScript Date-Objekt übergeben werden. Will man unterschiedliche Zeitstempel realisieren, genügt es, für die Messkanäle, bei denen zu diesem Zeitstempel kein Wert existiert, den Wert frei zu lassen bzw. mit „null“ aufzufüllen. Datenlücken werden mit „NaN“ (Not a Number) ausgezeichnet. Der Vorteil dieser Art von Datenbereitstellung ist, dass man nicht Gefahr läuft, unnötig doppelte Zeitstempel mitzuführen, wenn viele Messkanäle auf den gleichen Zeitstempeln beruhen. Die Kanalnamen werden im Array-Format separat angegeben, oder über die CSV-Datei aus der ersten Zeile geparst. Ein Code-Beispiel zeigt die Werteübergabe:

```
g = new Dygraph(
    document.getElementById("demodiv"),
    [
        [40711,983.6724,17],
        [40711.00069,983.6724,17],
        [40711.00139,983.7744,17]
    ],
    {
    //weitere Optionen...
    }
);
```

Abbildung 6: dygraphs: Beispiel-Code für die Messdatenübergabe

Die Dygraphs Community ist groß und sehr aktiv, so dass während dieser Projektarbeit zahlreiche Updates und Bugfixes eingespielt wurden. Sowohl der Hauptentwickler, als auch mehrere Projektmitglieder arbeiten an der Verbesserung und Weiterentwicklung mit. Der Open Source Code ist unter MIT-Lizenz frei zugänglich und lässt sich per Versionsverwaltung auf den aktuellsten Stand bringen. Mittelfristig gesehen, wird das Projekt voraussichtlich noch an Funktionalität hinzugewinnen und etwaige Fehler werden behoben.

4.3 Beurteilung der Webframeworks

Es hat sich herausgestellt, dass jedes Webframework, das in Kapitel 4.2 vorgestellt wurde, in etwa die gleichen Punkte aus dem Anforderungskatalog (Darstellungs-/Diagrammoptionen [Kapitel 3.2.1] und dynamische Optionen [Kapitel 3.2.1]) erfüllt. Alle bieten mehr oder weniger die gleichen bereits implementierten Funktionalitäten (Zoom, Interaktion mit einzelnen Messpunkten) und erlauben es, dank guter Dokumentation und Modularität, die nicht vorhandenen Anforderungen relativ leicht selbst einzubinden. Deshalb ist es allein aufgrund dieser Anforderungen nicht möglich, den einen oder anderen Kandidaten zu bevorzugen.
Weiterhin gilt es die nichtfunktionalen Anforderungen [Kapitel 3.2.3] näher zu untersuchen und zu vergleichen, da die Beispiele auf den Projektseiten nur mit überschaubaren Datenmengen gefüttert sind und so keinen Rückschluss auf die Handhabung mit größeren Datenmengen zulassen.
Zu diesem Zweck wurden die fünf Frameworks in ihrer aktuellen Version heruntergeladen und je acht Testseiten erstellt, die mit identischen Rohdaten versehen wurden. Dabei wurde zunächst mit der Grundkonfiguration, also im Auslieferungszustand der Webframeworks gearbeitet. Die Fallbeispiele für typische Datenmengen sind:

- 1 Messkanal, 1 Messtag â 1441 Messwerte
- 1 Messkanal, 7 Messtage â 10038 Messwerte
- 1 Messkanal, 30 Messtage â 43005 Messwerte
- 1 Messkanal, 90 Messtage â 125216 Messwerte
- 10 Messkanäle, 1 Messtag â 1441 Messwerte pro Kanal
- 10 Messkanäle, 7 Messtage â 10038 Messwerte pro Kanal
- 10 Messkanäle, 30 Messtage â 43005 Messwerte pro Kanal

- 10 Messkanäle, 90 Messtage â 125216 Messwerte pro Kanal

Im letztgenannten Fallbeispiel ergeben sich somit insgesamt 10x125216 Messwerte, also über eine Million Datenpunkte.

Die Rohdaten wurden direkt aus der Datenbank des ZAE Bayern gewonnen, in Microsoft Excel importiert und nach einem Export in eine CSV-Datei mittels einem selbst geschriebenen Perl-Script auf die jeweiligen Übergabeformate gebracht. Danach wurden die umformatierten Messdaten in die entsprechenden Testseiten eingepflegt und eine Indexseite erstellt, von der aus man alle Testseiten per Link starten kann. Ein kleines JavaScript-Snippet misst die Dauer, die das Webframework braucht, um die Daten darzustellen und gibt die Zeit in einer PopUp-Meldung aus:

```
var float_ts = new Date().getTime();
//hier steht der Graph-erzeugende Code des jeweiligen Webframeworks
alert('Zeit: '+ ((new Date().getTime() - float_ts)/1000));
```

Abbildung 7: JavaScript-Code zur Ermittlung der Zeit für die Graphgenerierung

Alle Messungen wurden am selben Tag, auf dem gleichen PC (Core2Duo 3GHz, 2GB RAM) und mit jeweils neu gestarteter Browser-Session pro Test ausgeführt. Als Browserumgebung wurde eine aktuelle Version von Chrome[4] herangezogen. Zudem wurde jeder Test dreimal gemessen und daraus der Mittelwert gebildet.

Da Highcharts als einziger Kandidat mit dem SVG-Standard arbeitet, stellte sich heraus, dass die Messung per JavaScript-Snippet in diesem Fall, nicht uneingeschränkt der tatsächlichen Anzeigedauer das Graphen entspricht. Die Erstellung des SVG-Objektes wird zwar korrekt gemessen, jedoch benötigt der Browser, gerade bei hohem Messdatenaufkommen, erheblich mehr Zeit, um den Graphen letztendlich sichtbar im Browserfenster anzuzeigen. Die Highcharts-Tests wurden deshalb nochmals per Handstoppuhr nachgemessen.

Da dygraphs in der Grundkonfiguration nur Linien ohne Schattenwurf ausgibt, und die anderen Kandidaten jeden Messpunkt mitzeichnen und einen Schatten hinter die Linien werfen, wurden noch zwei weitere Kategorien getestet. Zum einen wurde dygraphs so umkonfiguriert, dass jeder einzelne Messpunkt mitgezeichnet wird (P.), zum anderen wurde Flot modifiziert, so dass kein Schattenwurf und keine expliziten Messpunkte mehr mitgezeichnet werden (L.).

Die Messergebnisse sind in folgender Heatmap zusammengefasst:

[4] Stichproben mit Firefox 10 ergaben keine signifikanten Unterschiede

	Ausgangskonfiguration					*modif. Konfiguration*	
	Flot	**Highcharts**	**jqPlot**	**FlotR**	**dygraphs (L.)**	**dygraphs (P.)**	**Flot (L.)**
10x125216	180,15	N/A	279,36	148,99	4,52	5,81	26,01
10x43005	35,41	55,05	40,82	34,51	1,25	1,69	3,10
10x10038	7,01	7,98	7,02	6,83	0,33	0,45	0,33
10x1441	1,00	1,16	1,05	1,03	0,08	0,16	0,07
1x125216	9,80	12,63	10,63	9,52	0,43	0,58	0,72
1x43005	2,95	2,63	2,98	3,03	0,14	0,22	0,14
1x10038	0,72	0,64	0,75	0,74	0,05	0,08	0,05
1x1441	0,15	0,11	0,18	0,15	0,02	0,03	0,02
Zoomgewinn	linear	keiner, buggy	keiner	linear	linear	linear	linear

Abbildung 8: Heatmap zur Veranschaulichung der Messergebnisse der Darstellungsdauer (in Sekunden)

Es fällt auf, dass dygraphs selbst mit zugeschalteter Einzelpunktdarstellung (P.) für den größten Datensatz weniger als 6 Sekunden benötigt, während Flot mit reiner Liniendarstellung (L.) bereits 26 Sekunden braucht um das Diagramm anzuzeigen. Dygraphs ausgenommen, ist für die anderen Webframeworks generell zu sagen, dass die voreingestellten, optischen Effekte schon bei einem kleinen Tagesdatensatz mit 10 Messkanälen die Erstellung des Graphen deutlich einbremsen.

Außerdem wurde die benötigte Zeit für die Darstellung eines Teilbereichs durch einen Zoom in das Diagramm gemessen. Außer bei Highcharts und jqPlot konnte eine lineare Zeitersparnis im Verhältnis zu der im Zoomausschnitt enthaltenen Datenmenge nachgewiesen werden.

Auch der Unterschied der beiden Techniken, SVG (Highcharts) auf der einen, HTML5/canvas auf der anderen Seite ist signifikant. Gerade bei hohem Messdatenaufkommen wird SVG spürbar langsamer, beim größten Datensatz stürzte sogar der Browser ab, so dass keine Zeitmessung möglich war.

Als nächstes wurde eine Abschätzung der Skalierbarkeit der Webframeworks durchgeführt. Dazu wurde zum einen das Verhältnis der Messkanäle (10 zu 1) bei gleichbleibendem Messzeitraum, zum anderen das Verhältnis der Messzeiträume (z.B. 10038 zu 1441 Datenpunkte) bei gleichbleibender Kanalanzahl berechnet. Die Erwartungswerte ergeben sich dabei aus den theoretisch berechneten Verhältnissen (z.B. 10038:1441 = 6,97).

Eine weitere Heatmap zeigt die Resultate:

	Ausgangskonfiguration					*modif. Konfiguration*		
Erwartung	**Flot**	**Highcharts**	**jqPlot**	**FlotR**	**dygraphs (L.)**	**dygraphs (P.)**	**Flot (L.)**	**Verhältnis**
10	18,39		26,27	15,65	10,39	10,05	35,92	10:1 (125216)
10	12,01	20,97	13,68	11,39	8,66	7,56	22,40	10:1 (43005)
10	9,69	12,49	9,33	9,26	6,48	5,57	6,93	10:1 (10038)
10	6,55	10,49	6,02	6,99	4,10	5,15	3,92	10:1 (1441)
2,91	5,09		6,84	4,32	3,61	3,44	8,39	125216:43005 (10)
4,28	5,05	6,90	5,81	5,05	3,78	3,75	9,51	43005:10038 (10)
6,97	7,00	6,85	6,67	6,65	4,03	2,87	4,79	10038:1441 (10)
2,91	3,32	4,81	3,56	3,14	3,01	2,59	5,23	125216:43005 (1)
4,28	4,08	4,11	3,96	4,11	2,83	2,76	2,94	43005:10038 (1)
6,97	4,74	5,75	4,30	5,02	2,55	2,66	2,71	10038:1441 (1)

Abbildung 9: Heatmap zur Veranschaulichung der Skalierbarkeit der Webframeworks

Hierbei zeigte sich, dass außer dygraphs, keiner der anderen Kandidaten hinreichend skaliert und den Erwartungswerten bei hohen Messdatenaufkommen, gerecht wird.

Die folgenden Balkendiagramme zeigen den Vergleich der Verhältnisse zwischen Erwartungswert, Flot (nur Linien) und dygraphs (nur Linien):

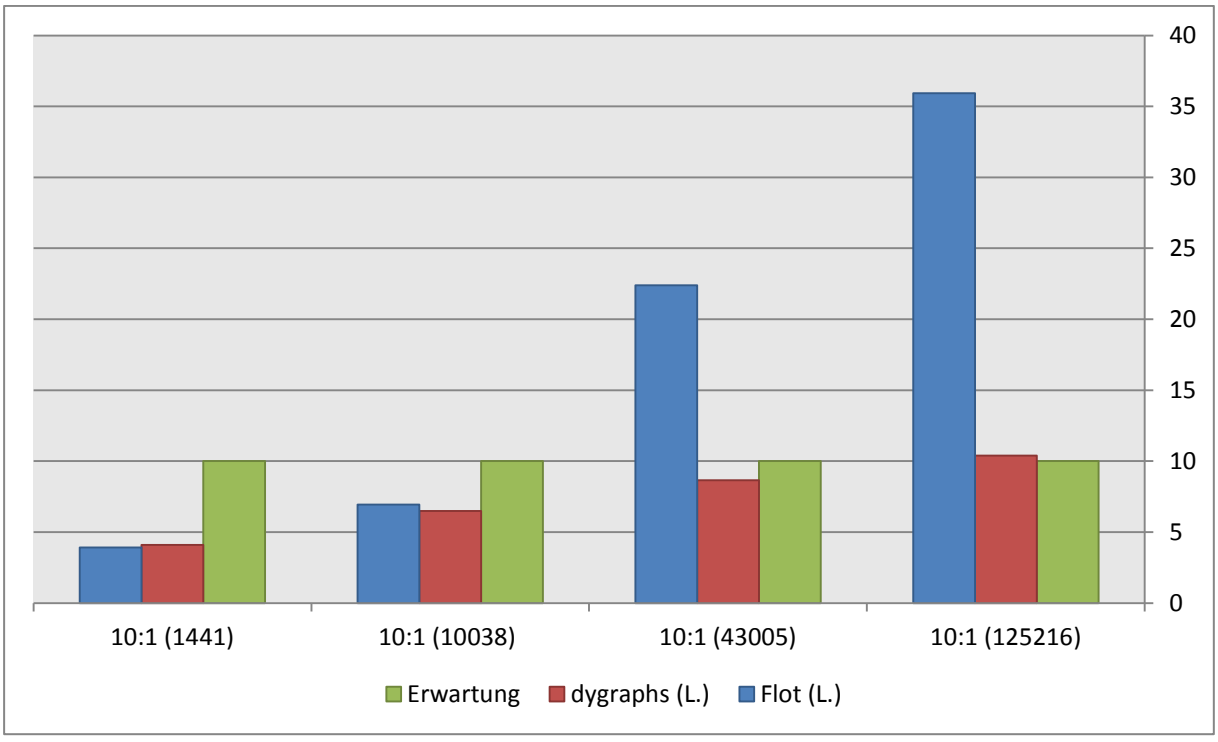

Abbildung 10: Balkendiagramm zur Veranschaulichung der Verhältnisse bei gleichbleibendem Messzeitraum

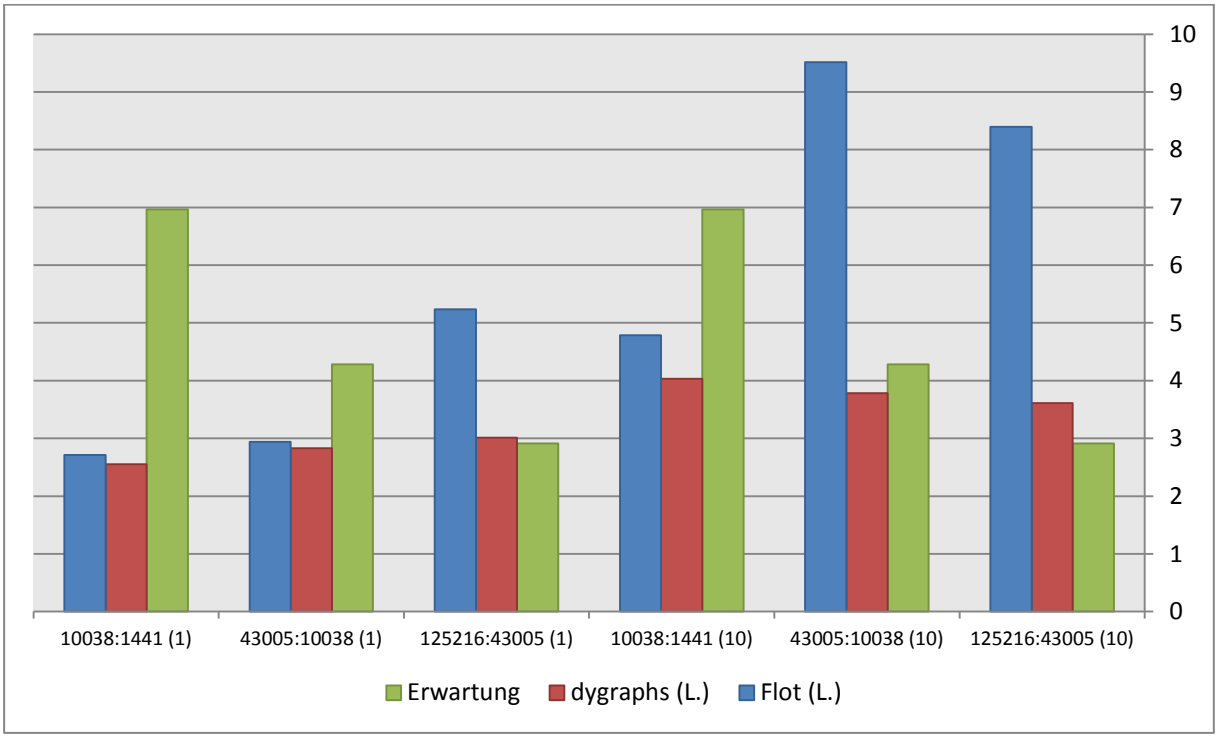

Abbildung 11: Balkendiagramm zur Veranschaulichung der Verhältnisse bei gleichbleibender Messkanalanzahl

Natürlich ist zu erwähnen, dass aufgrund gleicher Zeitstempel für alle Messkanäle hier nur der „Best Case“ getestet wurde, und dies der Datenvorhaltung von dygraphs zugutekommt, da die Zeitstempel nicht unnötigerweise mehrfach mitgeführt werden. Nichtsdestotrotz ist dieses „Best Case“ Szenario gängige Praxis am ZAE Bayern, so dass man in der Regel Messreihen hat, die auf denselben Zeitstempeln beruhen. Deshalb kann man bei der Auswahl eines geeigneten Webframeworks durchaus auf die gewonnenen Erkenntnisse bauen.

Gerade weil dygraphs den Fokus auf zeitbasierte Daten und hohe Performance bei der Interaktion mit dem Graphen setzt, anstatt allzu viel Augenmerk auf die Optik zu legen, empfiehlt es sich für die speziell benötigte Anwendung am ZAE Bayern. Die Messungen haben ergeben, dass dygraphs mit Abstand am besten mit hohen Datenaufkommen zurechtkommt und auch die Skalierung verlässlich abschätzbar ist. Somit fiel die Auswahl auf dygraphs als geeignetes Webframework zur Realisierung des geplanten Vorhabens.

5 Implementierung und Einbindung in das System

Die Sondierung der Webframeworks brachte das geeignete Webframework dygraphs hervor. Nun gilt es, dieses um nicht vorhandene Funktionen zu erweitern und in die bestehende Visualisierungs-Webapplikation Medview2 einzubinden.

5.1 Softwarearchitektur und Konzept

Dygraphs gibt es als Leerzeichen-entfernte JavaScript Datei für den produktiven Einsatz, wenn man nichts weiter am Framework verändern oder erweitern möchte. Man kann jedoch mittels der GIT Versionsverwaltung einen Checkout aus dem Projektverzeichnis machen, mit dem man das komplette Quellcode Paket erhält und gegebenenfalls seine Änderung wieder in das Projektverzeichnis hoch lädt. Nach Überprüfung der betreuenden Entwickler werden Änderungen der Allgemeinheit wieder zur Verfügung gestellt.

Medview2 selbst stellt eine Webservice-Methode bereit, mit der man über gewünschte Messkanäle einen Zeitbereich unter Anwendung von eventuellen Filtern abfragen kann. Um den Datenverkehr zwischen Client und Server zu verringern, wird anstelle von XML das JSON-Format verwendet. Per JavaScript werden die JSON-Objekte erstellt bzw. ausgelesen und an dygraphs weitergereicht.

Das Konzept für die Eingliederung von dygraphs in Medview2 umfasst zunächst, soweit möglich, die Konfiguration von dygraphs anhand des Anforderungskatalogs. Danach werden in einer lokalen Umgebung die noch fehlenden Anforderungsmerkmale implementiert. Abschließend wird dieses speziell konfigurierte Visualisierungspaket in Medview2 eingebunden, so dass man direkt per Datenbankabfrage die Messdaten in das Diagramm laden kann. Zusätzlich wurde eine Strategie entwickelt, wie man mit typischen Messdatenaufkommen umgeht.

5.2 Konfiguration von mitgelieferten Funktionalitäten

Eine der wichtigsten Anforderungen, nämlich der Darstellung von Messwerten mit unterschiedlichen Zeitstempeln, wird von dygraphs von Haus aus unterstützt. Hierbei muss nur darauf geachtet werden, dass bei der späteren Einbindung in das Gesamtsystem das Rückgabeformat der Messwerte dem Eingabeformat von dygraphs entspricht. Weiterhin ist der direkte Zugriff auf einzelne Messpunkte bereits implementiert, so dass man beim Überfahren der Messkurven mit der Maus die exakten Werte in der Legende angezeigt bekommt.

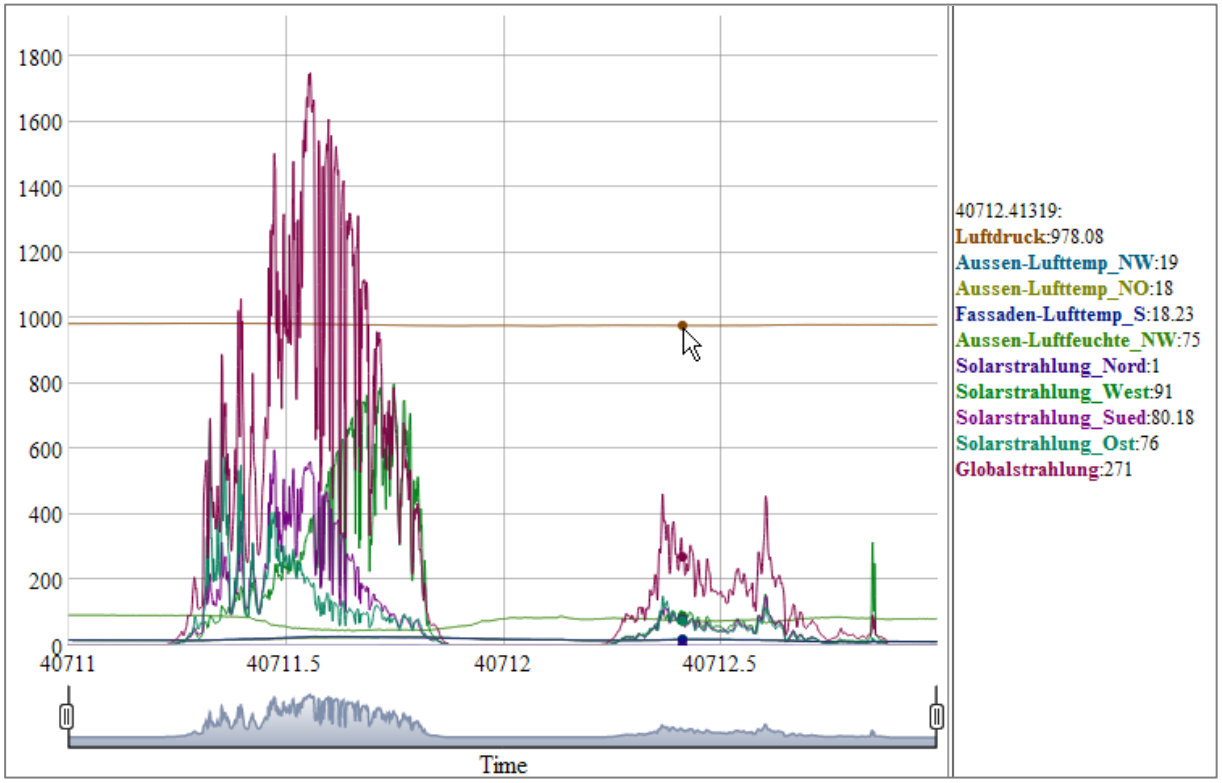

Abbildung 12: Screenshot: Anzeige exakter Messwerte in der Legende und Zoomübersicht

Außerdem ist es möglich, neben der automatisierten Vergabe von Linienfarben, diese manuell einzustellen. Hierbei gilt es bei der Einbindung in Medview2, dem Benutzer die Einstellungen innerhalb der Weboberfläche zu ermöglichen. Auch das Umschalten von einer linearen zu einer logarithmischen Skalierung und umgekehrt ist in dygraphs bereits vorgesehen. Das Exportieren einer Grafikdatei aus dem angezeigten Diagramm wurde im Hauptprojekt bislang nicht realisiert, jedoch existiert eine Erweiterung eines dygraphs Nutzers, der diese zur freien Verfügung unter der MIT-Lizenz bereitgestellt hat [25].

Zoomfunktionen gibt es zweierlei in dygraphs, und zwar immer bereichsweise entweder in X- oder in Y-Richtung, ebenso wie eine Zoomübersicht unterhalb des Graphen, mit dem man auch den Zoombereich verschieben oder verändern kann. Das Ein- und Ausblenden von Messkurven ist ebenfalls möglich. Diese Funktionalität muss dem Benutzer noch im Webinterface durch eigene Implementierung bedienbar bereitgestellt werden. Das Setzen von Markierungen per Mausklick auf einem Messpunkt wird unterstützt und kann nach Belieben funktional erweitertet werden.

5.3 Erweiterung um Funktionalitäten

Die Legende wurde mehrfach modifiziert. Sie hat einen Hoover-Effekt erhalten, der beim Überfahren der Messkanalbeschriftungen mit der Maus die entsprechende Messkurve hervorhebt. Dabei werden die anderen Messkurven im Hintergrund ausgegraut, so dass sie noch sichtbar sind, aber nicht den Blick auf die hervorgehobene Linie stören. Gerade bei vielen Messkanälen, die mitunter auch ähnliche Messwerte haben, kann man diese nun besser voneinander unterscheiden. Außerdem wurde eine Funktion in der Legende hinterlegt, die es ermöglicht mit einem Mausklick auf den Messkanalnamen die jeweilige Messkurve auszublenden, und sie bei erneutem Klick darauf wieder einzublenden.

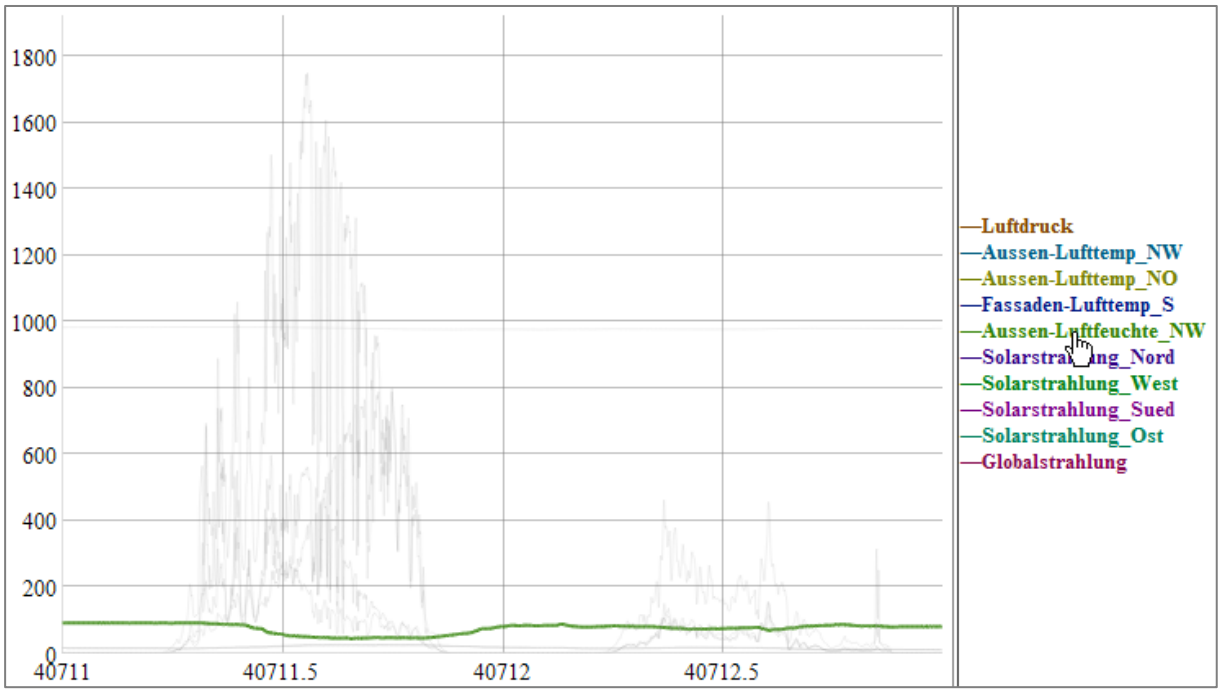

Abbildung 13: Screenshot: Erweiterung der Legende zum Hervorheben von Messkurven

Die Möglichkeit in ein Diagramm hinein oder heraus zu zoomen ist die wohl meist genutzte dynamische Interaktion mit dem Graphen. Dygraphs ermöglicht das per separatem X- bzw. Y-Achsen-Zoom, lässt einem jedoch die Freiheit dies über ein individuelles „interaction model“ zu verändern und zu erweitern. So wurde zunächst ein klassischer X-Y-Zoom eingebaut, mit dem man ein Rechteck innerhalb des Diagramms aufziehen kann, in welches dann hineingezoomt wird. Ein Button auf der Benutzeroberfläche zoomt wieder vollständig heraus. Das Mausrad wurde doppelt belegt. Zum einen kann man prozentual zoomen, indem man per Mausrad raus bzw. runter scrollt, zum anderen kann die Mausradtaste gedrückt und gehalten werden,

um einen Bereichszoom in X- oder Y-Richtung zu vollziehen. Die rechte Maustaste zeigt nicht mehr das übliche Kontextmenü im Browser, wenn sie innerhalb des Diagrammbereichs gedrückt wurde. Gedrückt und gehalten kann man mit ihr den Zoombereich verschieben, und sich so durch einen gezoomten Graphen bewegen. Das Setzten von Messpunktmarkierungen (siehe Abbildung 14) wurde so abgewandelt, dass diese fortlaufend durchnummeriert werden, und bei Klick auf eine Markierung, diese wieder gelöscht wird. Dabei wird die Nummerierung neu „berechnet“, so dass keine Lücken entstehen und die Reihenfolge erhalten bleibt. Die Markierungen werden in zwei Dropdown-Menüs aufgelistet, mit denen man Start und Endpunkt auswählen und verschiedene Berechnungen ausführen kann. Hierfür gibt es noch ein Dropdown-Menü für die Berechnungsfunktion und eines für die Messkanäle, über die die Berechnung erfolgen soll. So ist es möglich die Markierungen auf unterschiedlichen Messkurven zu setzen und die Berechnung wiederum auf einem anderen Messkanal ausführen zu lassen.

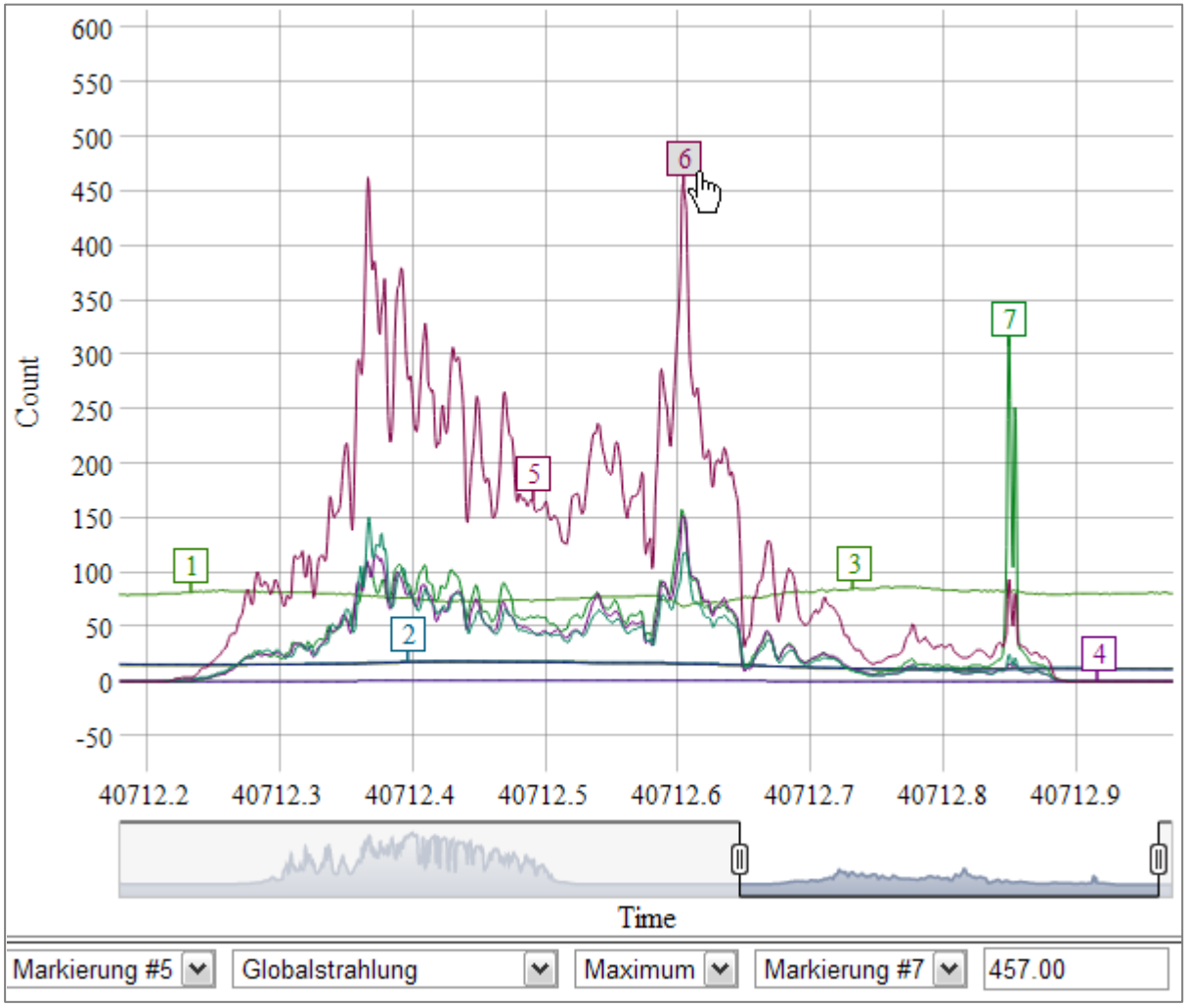

Abbildung 14: Screenshot: Setzen von Markierungen und Berechnung eines Maximums dazwischen

Derzeit sind die Funktionen zur Berechnung eines Maximums, Minimums und des arithmetischen Mittelwertes implementiert.
Das Grafikexport-Modul wurde dahingehend verbessert, dass neben der zweiten Y-Achse auch die Markierungen der einzelnen Messpunkte mit in die PNG-Datei exportiert werden.
Die meisten, spezifischen Änderungen erfolgten im ausgelagerten „interaction model“, welches über Versionsupdates hinaus ohne weitere Anpassung weiter benutzt werden kann. Die Änderungen zum Hervorheben von Messkurven sind im Quellcode des Webframeworks implementiert worden und zurück in das dygraphs Projekt geflossen, so dass zukünftige Updates ohne Modifikation möglich sind.
Alle Erweiterungen und Verbesserungen können auf einer Demo-Seite [26] des ZAE Bayern eingesehen und ausprobiert werden.

5.4 Einbindung in die Server-Client-Struktur

Nachdem die Modifikationen des Webframeworks in einer lokalen Umgebung abgeschlossen waren, wurden diese in die Webapplikation Medview2 integriert. Dazu wurde eine Entwicklungsseite auf Basis der Version mit statisch generierten Diagrammbildern bereitgestellt, die bereits das Gerüst der Webapplikation (u.a. Messkanalauswahl und Zeitintervallfestlegung) beinhaltet. Darin wurde zunächst die modifizierte Demo-Seite eingebunden und die Funktionstüchtigkeit mit lokal hinterlegten Messdaten verifiziert.
Danach wurde die Messdatenbereitstellung so umgebaut, dass der Graph per Datenbankanfrage mit den zurückgelieferten Messwerten gespeist wird. Nun musste eine Strategie ausgearbeitet werden, wie man den unterschiedlichen Zeitintervallanfragen und den damit verbundenen typischen Messdatenaufkommen gleichermaßen gerecht werden kann.
Die Testmessungen hatten ergeben, dass dygraphs auf dem Testsystem bei 10 Messkanälen und 10038 Messpunkten (identische Zeitstempel) sich schnell aufgebaut hat und noch gut bedienbar ist. Weitaus größere Datenmengen, wie bei Monats- oder Jahresplots sind nicht mehr handelbar. In Anbetracht der Tatsache, dass das erstellte Diagramm im Browserfenster 700 Pixel Breite einnimmt und somit der Graph im nicht gezoomten Zustand in etwa 700 Messpunkte anzeigen kann,

wurde zunächst ein Maximalwert von 3000 darzustellenden Messpunkten[5] festgelegt. Dieser Wert ist im Quellcode an zentraler Stelle anpassbar, so dass man ihn nach Rückmeldungen von den Nutzern leicht nach unten oder oben korrigieren kann.
Egal welches Zeitintervall der Benutzer auswählt, es sollen maximal 3000 gezeichnete Diagrammpunkte dargestellt werden. Um dies für große Zeitintervalle zu gewährleisten, kann man mit dem Medview2 Webservice die Messdaten mit einem Filterintervall und einer Filterfunktion (Min-Max bzw. Mittelwert) anfordern. Die Min-Max-Funktion liefert innerhalb des übergebenen Filterintervalls den kleinsten und den größten Messwert pro Kanal mit entsprechenden Zeitstempeln zurück. Die Mittelwert-Funktion bildet den arithmetischen Mittelwert der Messwerte pro Kanal innerhalb des Filterintervalls und gibt diesen mit dem ersten Zeitstempel zurück, der der Intervallgrenze am nächsten ist.
Um das passende Filterintervall für den gewählten Zeitraum zu bestimmen, wird zunächst immer erst ein Datensatz über einen reduzierten Zeitraum mit maximal 1500 Datenzeilen angefordert und analysiert. Beinhaltet die Rückantwort der Datenbank weniger als die 1500 Datenzeilen, ist man bereits fertig und kann die angeforderten Daten direkt an dygraphs weitereichen und ausgeben. Ist der vom Benutzer gewählte Zeitraum größer als der reduzierte Zeitraum, so kann man per Dreisatz die zu erwartenden Datenzeilen x berechnen:

$$\frac{x}{1500} = \frac{dT_{ges}}{dT_{red}} \qquad | * 1500$$

$$x = 1500 * \frac{dT_{ges}}{dT_{red}}\ , mit\ x\text{: } die\ zu\ erwartenden\ Datenzeilen\ im\ Gesamtzeitraum$$

$$mit\ dT_{ges}\text{: } vom\ Benutzer\ ausgewählter\ Gesamtzeitraum$$

$$mit\ dT_{red}\text{: } reduzierter\ Zeitraum\ mit\ 1500\ Datenzeilen$$

Liegt die Erwartung innerhalb der Obergrenze von 3000 Datenzeilen, können die Messwerte im Gesamtzeitraum ungefiltert angefordert und visualisiert werden. Liegt sie darüber, wird entsprechend ein passendes Filterintervall berechnet, so dass die gefilterte Rückantwort der Datenbank maximal 3000 Datenzeilen umfasst.
Nun könnte man den Gesamtzeitraum gefiltert von der Datenbank anfordern, dies hätte jedoch den Nachteil, dass bei sehr großen Zeiträumen die Datenbank länger

[5] entspricht in etwa 2 Messtagen (48h) bei Minutentaktung und identischen Zeitstempeln

benötigt, um die Daten vollständig zurückzuliefern. Deshalb wird ein anderer Ansatz verfolgt, der die Messdaten „häppchenweise“ anfordert und die zurückgelieferten Datensegmente an die Visualisierung übergibt. Das hat den Vorteil, dass der Benutzer während der Datenbeschaffung über den Fortgang graphisch, durch langsamen Aufbau des Plots, informiert wird.

Zuerst wird ein gefiltertes Startsegment angefordert, dem der zuvor ungefilterte, reduzierte Zeitraum zugrunde liegt. Dann wird vorausschauend geprüft, ob das Ende des Gesamtzeitraums innerhalb der nächsten 48 Stunden[6] liegt. Ist das der Fall, wird ein gefiltertes Endsegment angefordert, dass sich nahtlos an das Startsegment anreiht. Ist der Gesamtzeitraum größer, werden nach dem Startsegment gefilterte 48h-Segmente angefordert und ausgegeben, bis das Ende des Gesamtzeitraums erreicht ist und das Endsegment generiert werden kann.

Ein Programmablaufplan zeigt schematisch die Abarbeitung der Datenaufbereitung und Diagrammerzeugung:

[6] Erfahrungswert. Ist im Quellcode variierbar, um größere bzw. kleinere Segmente zu erhalten.

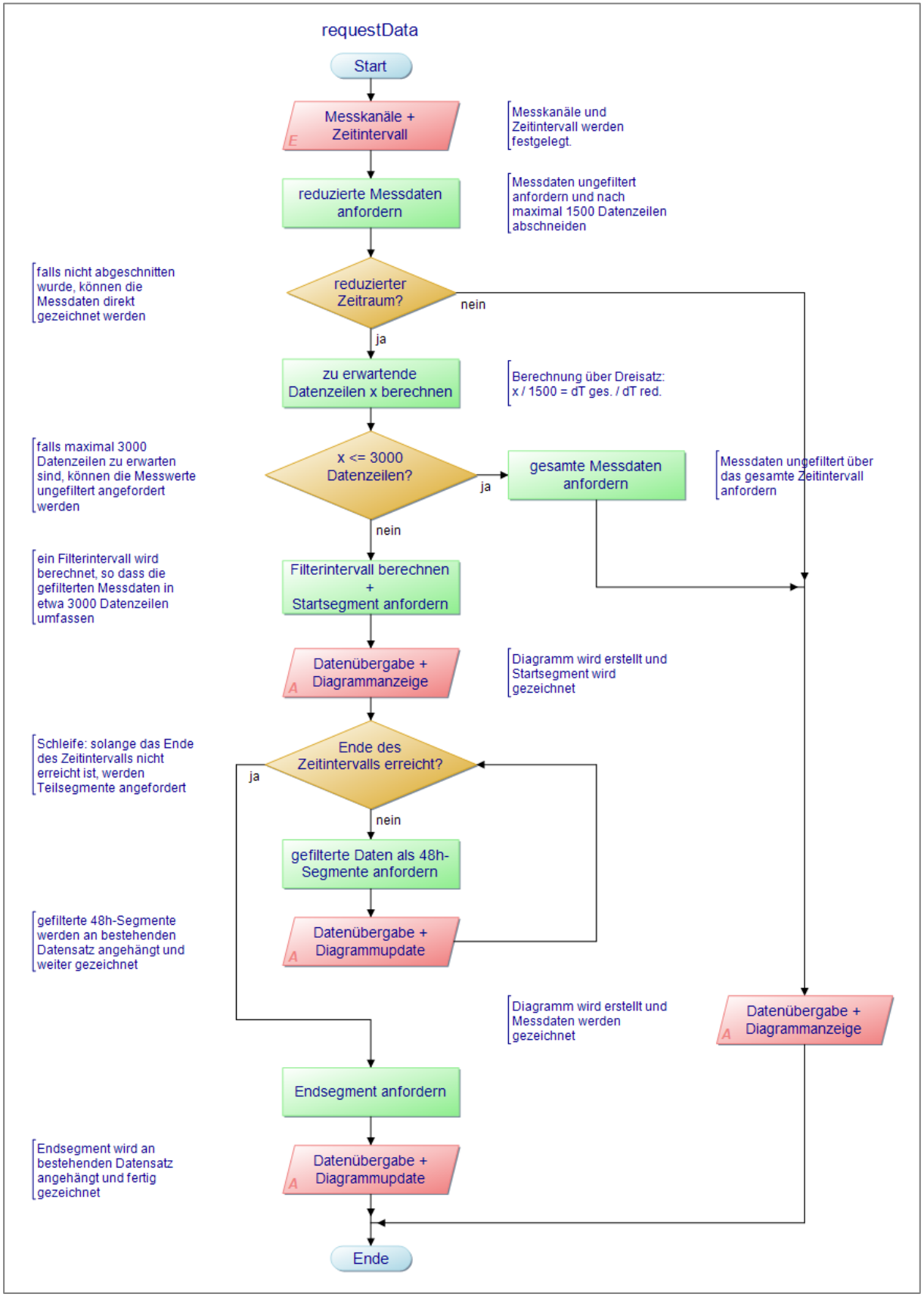

Abbildung 15: Programmablaufplan einer Diagrammerzeugung mit dygraphs in Medview2

Nach der Erstellung des Graphen stehen alle Funktionalitäten der modifizierten Version von dygraphs dem Benutzer zur Verfügung. Wird in das Diagramm hinein gezoomt, passen sich die Zeitraumauswahlelemente automatisch dem aktuellen Zoombereich an. So kann der Benutzer gefilterte Messdaten, die er näher betrachten möchte, per Knopfdruck neu anfordern, die dann je nach Umfang weniger gefiltert bzw. ungefiltert ausgegeben werden.

6 Fazit und Ausblick

Die intensive Auseinandersetzung mit Webframeworks zur dynamischen Visualisierung von Messdaten hat sich rückblickend betrachtet in mehrerlei Hinsicht gelohnt. So konnte zunächst zwischen den Bibliotheken mit Grundfunktionalität und den spezialisierten Visualisierungsframeworks differenziert werden. Daraufhin wurden Letztere näher inspiziert und die Stärken und Schwächen der einzelnen Kandidaten herauskristallisiert. Ein Performance-Test mit real auftretenden Bedingungen zeigte große Unterschiede zwischen den Bibliotheken, und führte letztendlich zu einem geeignetem Webframework für die Umsetzung des Vorhabens.
Nach der Einarbeitung in die Visualisierungsbibliothek dygraphs, konnte dieses um weitere Funktionalitäten aus dem Anforderungskatalog erweitert werden, die auch teilweise in die Entwicklerversion zurückgeflossen sind. Die Einbindung des Webframeworks in die bestehende Visualisierungsumgebung wurde durch eine passende Strategie erfolgreich vollzogen.
Ausblickend sollte das modifizierte Webframework stets mit den Updates aus dem Projektstamm migriert werden, da in absehbarer Zukunft stets neuere Features und Bugfixes in das Projekt einfließen werden. Außerdem wäre es wünschenswert, dass alle Beschriftungen und Markierungen zur Laufzeit abänderbar sind. Ein weiteres ambitioniertes Highlight wäre die dynamische Datenaktualisierung bei Zoom in einen gefilterten Graphen.
Mit Abschluss dieser Projektarbeit wird die dynamische Visualisierung in Medview2 auch allen Mitarbeitern des ZAE Bayern zur Verfügung gestellt und anhand von Rückmeldungen weiter optimiert.

Abbildungsverzeichnis

Abbildung 1: Schema des bestehenden Systems 4
Abbildung 2: Highcharts: Beispiel-Code für die Messdatenübergabe 10
Abbildung 3: jqPlot: Beispiel-Code für die Messdatenübergabe 11
Abbildung 4: Flot: Beispiel-Code für die Messdatenübergabe 12
Abbildung 5: Flotr: Beispiel-Code für die Messdatenübergabe 13
Abbildung 6: dygraphs: Beispiel-Code für die Messdatenübergabe 14
Abbildung 7: JavaScript-Code zur Ermittlung der Zeit für die Graphgenerierung 16
Abbildung 8: Heatmap zur Veranschaulichung der Messergebnisse der Darstellungsdauer (in Sekunden) 17
Abbildung 9: Heatmap zur Veranschaulichung der Skalierbarkeit der Webframeworks 18
Abbildung 10: Balkendiagramm zur Veranschaulichung der Verhältnisse bei gleichbleibendem Messzeitraum 18
Abbildung 11: Balkendiagramm zur Veranschaulichung der Verhältnisse bei gleichbleibender Messkanalanzahl 19
Abbildung 12: Screenshot: Anzeige exakter Messwerte in der Legende und Zoomübersicht 21
Abbildung 13: Screenshot: Erweiterung der Legende zum Hervorheben von Messkurven 22
Abbildung 14: Screenshot: Setzen von Markierungen und Berechnung eines Maximums dazwischen 23
Abbildung 15: Programmablaufplan einer Diagrammerzeugung mit dygraphs in Medview2 27

Literaturverzeichnis

[1] Wikipedia-Artikel: Speicherprogrammierbare Steuerung. [Online] http://de.wikipedia.org/wiki/Speicherprogrammierbare_Steuerung.

[2] **Braun, Herbert.** Datenbilder. *c't.* 2011, 26.

[3] Wikipedia-Artikel: Comparison of JavaScript frameworks. [Online] http://en.wikipedia.org/wiki/Comparison_of_JavaScript_frameworks.

[4] Ample SDK: Charts examples. [Online] http://www.amplesdk.com/examples/markup/chart/.

[5] dhtmlxChart: Live Demo. [Online] http://dhtmlx.com/docs/products/dhtmlxChart/.

[6] dojox.charting: Documentation. [Online] http://dojotoolkit.org/reference-guide/dojox/charting.html.

[7] dojox.charting: Tutorial. [Online] http://dojotoolkit.org/documentation/tutorials/1.7/charting/.

[8] Ext JS Charting: Documentation. [Online] http://docs.sencha.com/ext-js/4-0/#!/guide/drawing_and_charting.

[9] Ext JS Charting: Samples. [Online] http://dev.sencha.com/deploy/ext-4.0.7-gpl/examples/#sample-3.

[10] YUI Library: Charts. [Online] http://yuilibrary.com/yui/docs/charts/.

[11] Google Chart Tools: Homepage. [Online] http://code.google.com/intl/de-DE/apis/chart/.

[12] jQuery Visualize Plugin: Examples. [Online] http://www.filamentgroup.com/examples/charting_v2/.

[13] HighCharts: How to use. [Online] http://www.highcharts.com/documentation/how-to-use.

[14] HighCharts: Demo. [Online] http://www.highcharts.com/demo/.

[15] jqPlot: Examples. [Online] http://www.jqplot.com/tests/.

[16] jqPlot: Usage Documentation. [Online] http://www.jqplot.com/docs/.

[17] Flot: Examples. [Online] http://people.iola.dk/olau/flot/examples/.

[18] Flot: Reference. [Online] http://people.iola.dk/olau/flot/API.txt.

[19] Flotr: Documentation. [Online] http://solutoire.com/flotr/docs/.

[20] Flotr: Example Index. [Online] http://phenxdesign.net/flotr/examples/prototype/.

[21] Flotr: Playground. [Online] http://flotr.googlecode.com/svn/trunk/flotr/playground/index.html.

[22] dygraphs: Homepage. [Online] http://dygraphs.com/index.html.

[23] dygraphs: Gallery. [Online] http://dygraphs.com/gallery/.

[24] dygraphs: Tests. [Online] http://dygraphs.com/tests/.

[25] dygraphs: PNG Export Modul. [Online] http://cavorite.com/labs/js/dygraphs-export/.

[26] Medview2 Entwurf: Demo-Seite mit Erweiterungen und Verbesserungen. [Online] http://muc.zae-bayern.de/dygraphs/tests/medview2_entwurf.html.

Erklärung

Hiermit erkläre ich gemäß § 31 Abs. 7 der Rahmenprüfungsordnung, dass ich die vorliegende Arbeit mit dem Titel „Dynamische Visualisierung von Messdaten mithilfe von aktuellen Webframeworks" selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benützt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

München, den 29. Februar 2012

_____

Unterschrift

Danksagung

Zunächst gilt mein Dank Prof. Dr. François Bry, der sich bereit erklärt hat, diese Projektarbeit an einem externen Institut zu betreuen. Er hat mir während der gesamten Arbeit stets mit guten Hinweisen und wertvollen Ratschlägen zur Seite gestanden.
Ganz besonders möchte ich mich bei Christian Hilgers bedanken, der mich am ZAE Bayern über seine Betreuung hinaus mit technischer Hilfe und richtungsweisenden Tipps unterstützt hat. Insbesondere bedanke ich mich für das geduldige Lesen und die vielen konstruktiven Kommentare, die zur Qualität dieser Projektarbeit enorm beigetragen haben.