Bachelorarbeit

Konzeption und Entwicklung von grafischen Komponenten zur Visualisierung von Projektdaten

vorgelegt bei

Prof. Dr. Ludwig Nastansky

betreut durch

Dipl.-Wirt.-Inf. Bernd Hesse
Dr. Rolf Kremer

Sommersemester 2008

vorgelegt von

Oliver Köhler

Kapplenkamp 31
33378 Rheda-Wiedenbrück

Inhaltsverzeichnis

Inhaltsverzeichnis....2
1 Einleitung....1
1.1 Thematische Einführung....1
1.2 Zielsetzung....1
1.3 Aufbau der Arbeit....2
2 Thematische Grundlagen....3
2.1 Projekte....3
2.2 Komponenten / Widgets....4
2.3 JavaScript....5
2.4 Anwendungsumgebung....5
3 Konzepte / Methodik....7
3.1 Anforderungen an die Lösung....7
3.2 Auswahl eines geeigneten JavaScript Frameworks....8
3.3 Schnittstelle zwischen Widget und einbindender Seite....11
4 Prototypische Realisierung....14
4.1 Beschreibung der Fähigkeiten der Lösung....14
4.2 Bekannte Fehler....18
5 Fazit....20
Literaturverzeichnis....21
Monographien....21
Online-Quellen....22
Anhang....24
Eidesstattliche Erklärung....29

Abbildungsverzeichnis

Abbildung 1: Komponente im Zusammenhang mit Konfigurations- und Sprachdateien.....8
Abbildung 2: Vergleich Frameworks.....11
Abbildung 3: Aufgaben und Kommunikation von Client und Server.....12
Abbildung 4: Vergleich JSON und XML.....13
Abbildung 5: Technischer Aufbau der Widgets.....14
Abbildung 6: Beispieleinbindung des Widgets „timeline“ in der PCS (Fotomontage).....15
Abbildung 7: Das Widget „costtrend“ im Schweberahmen.....16
Abbildung 8: Das Widget "pie".....18

Abkürzungsverzeichnis

BSD – Berkeley Software Distribution

CSS – Cascading Style Sheets

DOM – Document Object Model

DHTML – Dynamic Hypertext Markup Language

HTML – Hypertext Markup Language

IBM – International Business Machines Corporation

JSON – JavaScript Object Notation

MIT – Massachusetts Institute of Technology

PCS – Process Control Suite

PHP – Hypertext Preprocessor (ursprünglich: Personal Homepage Tools)

WWW – World Wide Web

XML – Extensible Markup Language

1 Einleitung

1.1 Thematische Einführung

In der Anfangszeit der elektronischen Datenverarbeitung wurden Informationen rein textbasiert dargestellt. Mit ihrer Weiterentwicklung entstanden neue Methoden der Datenaufbereitung. Eine dieser Methoden ist die grafische Darstellung von Informationen.

Im World Wide Web inklusive der webbasierten Anwendungen ist eine Analogie dieser Situation zu beobachten. Die erste Version der für das WWW ausgelegten Auszeichnungssprache HTML (vom 3. November 1992) war nur für die Darstellung von Text bestimmt.[1] Mit der nächsten Version, welche am 30. April 1993 erschien, wurde erstmalig die Möglichkeit zur Einbindung von statischen Grafiken gegeben.

Eine Weiterentwicklung dieser rein statischen Seiten wurde erst durch die Einführung der sogenannten Web 2.0-Techniken[2] ermöglicht. Unter diesem Begriff ist eine Art der gemeinsamen Verwendung bereits vorhandener Techniken zusammengefasst.[3] Durch diese Techniken stehen Verfahren zur Generierung von dynamischen Seiten bzw. Elementen zur Verfügung. Im Zuge dieser Entwicklung entstehen immer neue Techniken zur Erzeugung von dynamischen Anwendungen. Die erweiterten interaktiven Funktionalitäten werden durch Web 2.0-Techniken für das Word Wide Web sowie für webbasierte Anwendungen gegeben. Dynamisch erzeugte Grafiken, welche auf die Veränderungen der Daten reagieren, sind bisher aber nur selten zu finden. Die Darstellung von variablen Daten erfolgt derzeit fast ausschließlich textbasiert.

1.2 Zielsetzung

Diese Arbeit wird von der PAVONE AG in Paderborn betreut. Eines ihrer Produkte ist die PAVONE Process Control Suite (PCS), welche eine Funktion zur Verwaltung von Projekten beinhaltet. Ziel der Arbeit ist es, grafische Elemente zur Visualisierung von Projektdaten zu entwickeln, um diese in der PCS verwenden zu können. Diesen Elementen werden Daten übergeben, mittels derer sie eine dynamische Grafik erzeugen sollen.

Anhand der Programmierung von geeigneten Komponenten sollen die Möglichkeiten der technischen Umsetzung gezeigt und deren Lauffähigkeit untersucht werden.

1 vgl [HTML Version 1, 2008]
2 s. [OReilly, 2008]
3 vgl.[Alby, 2008] S. 144

1.3 Aufbau der Arbeit

Im zweitem Kapitel werden die thematischen Grundlagen erörtert. Es wird ein Überblick über die wichtigsten Begriffe und Techniken dieser Arbeit gegeben. Das Konzept und die untersuchten Methoden, welche zur Lösungsfindung beigetragen haben, werden im dritten Kapitel vorgestellt. Hier wird insbesondere die Auswahl der verwendeten Werkzeuge begründet. Die prototypische Realisierung im vierten Kapitel befasst sich mit der Art der Umsetzung des Konzeptes in den Programmen und zeigt die Funktionen der erstellten Komponenten. Zusätzlich werden die bisher nicht gelösten Fehler der Prototypen analysiert. Im fünften Kapitel wird das Resümee und der Ausblick in einem Fazit zusammengefasst.

An verschiedenen Stellen werden Hinweise auf Quellen zu weiterführenden Informationen gegeben, welche für Leser, insbesondere für Entwickler von Komponenten, nützlich sein können. Diese Informationen werden bewusst in dieser Arbeit nicht weiter ausgeführt, da sie für die direkten Untersuchungen in dieser Arbeit nicht relevant sind.

2 Thematische Grundlagen

2.1 Projekte

Diese Arbeit befasst sich mit der Auswertung von Projektdaten. Um die Daten genauer klassifizieren zu können, wird der Begriff Projekt betrachtet. In der DIN 69 901 „Projekt-management-Begriffe“ ist ein Projekt definiert als

> *„Vorhaben, das im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist wie z.B.*
>
> *- Zielvorgabe*
>
> *- zeitliche, finanzielle, personelle oder andere Begrenzungen*
>
> *- Abgrenzung gegenüber anderen Vorgaben*
>
> *- projektspezifische Organisation“.*[4]

Unter der Zielvorgabe versteht man die projektbezogenen Ziele. Diese werden in der Regel vom ausführenden Unternehmen vorgegeben und betreffen den mittelbaren oder unmittelbaren Ablauf eines Projekts. Die Projektplanung befasst sich mit der Vorbereitung der Projektdurchführung im Sinne einer gedanklichen Vorwegnahme zukünftiger Handlungen.[5] Welche Zielsetzungen bei der Planung eines Projekts verfolgt werden, hängt von dessen Planungshorizont ab. Hier unterscheidet man zwischen langfristiger (strategischer), mittelfristiger (taktischer) und kurzfristiger (operativer) Projektplanung. Die langfristige Projektplanung befasst sich mit einem Zeitraum von mehreren Jahren. Insbesondere nicht kurz- oder mittelfristig verfügbare Ressourcen werden dabei betrachtet. Die mittelfristige Projektplanung kalkuliert einen Zeitraum von mehreren Monaten bis hin zu wenigen Jahren. Die Ressourcenplanung berücksichtigt neben Schlüsselressourcen auch kurz- oder mittelfristig am Markt zu beschaffende Ressourcen. Bei der kurzfristigen Projektplanung liegt ein Planungszeitraum von wenigen Wochen zugrunde. Die Ressourcen werden als fix angenommen, daher werden hier eher die zeitbezogenen Ziele betrachtet. Für eine effiziente Projektplanung sind detaillierte Informationen über das Projekt notwendig.[6]

Die Zeitplanung versucht die Durchlaufzeit für die gesamten Vorgänge eines Projekts zu ermitteln. Laut Fischer, Spiekermann wird die Zeitplanung oft als entscheidend betrachtet, obwohl Sachziele erst im Zeitablauf durch ihre organisatorische und personelle Verankerung realisiert werden. Zeiten sind oft kein sachlich bedingtes Ziel, sondern eine Vorsteuergröße, um den Termindruck bei

4 DIN 69 901 zitiert nach [Burghardt, 2002] S. 21
5 vgl. [Zimmermann; Stark; Rieck, 2006] S. 3
6 vgl. [Zimmermann; Stark; Rieck, 2006] S. 37ff

Durchlaufzeiten zu vermeiden.[7]

Das Projektmanagement umfasst die Gesamtheit aller Planungs-, Steuerungs-, Koordinierungs- und Überwachungsaufgaben zur sach-, termin-, und kostengerechten Realisierung von Projekten sowie die hierfür benötigten Konzepte und Methoden.[8]

2.2 Komponenten / Widgets

In dieser Arbeit werden Komponenten zur grafischen Darstellung von Projektdaten entwickelt. Der Begriff Komponente ist nicht eindeutig definiert.[9] Unter Komponenten versteht man im Allgemeinem Softwareteilstücke, welche eine bestimmte Aufgabe erledigen. Diese können an verschiedenen Stellen, an welchen die Aufgabe benötigt wird, eingesetzt werden. Hierfür besitzen diese Komponenten eine entsprechend definierte Schnittstelle. Ein wesentliches Charakteristikum ist, dass Komponenten nicht eigenständig betrieben werden können. Vielmehr benötigen sie ein Programm, welches sie mit entsprechenden Parametern aufruft. Weitere zentrale Eigenschaften von Komponenten sind die Konfigurierbarkeit, Anpassbarkeit und Adaptierbarkeit. Dies bedeutet, dass das Verhalten und ggf. das Aussehen der Komponente, je nach Einbindungszweck, unterschiedlich angepasst und an unterschiedlichen Stellen eingebunden werden kann. Der Unterschied zu Klassen und Objekten liegt in der Austauschbarkeit der Komponenten. Diese sollen ohne Anpassungen am aufrufendem Programm vorgenommen werden können.

Die Wiederverwendung von Komponenten bietet grundsätzlich eine Reihe von Vorteilen. Geringere Entwicklungskosten werden, durch das Zurückgreifen auf die bereits mit der Komponente realisierte Lösung, erreicht. Die zentrale Wartbarkeit der Komponente verringert zusätzlich die Wartungskosten. Sollen bereits vorhandene und sich im Einsatz befindende Komponenten an anderer Stelle eingebunden werden, so ist hier eine höhere Softwarequalität vorhanden, da diese bereits getestet wurden. Komponenten, die bereits auf effizienten Algorithmen basieren, tragen zur Konstruktion von effizienter Gesamtsoftware bei.[10]

Nachteilig bei der Entwicklung von Komponenten ist der zusätzliche Aufwand für die Definition der Schnittstelle. Es müssen zwischen dem einbindendem Programm und der Komponente Daten ausgetauscht werden. Für diese Kommunikation müssen beide Programmteile sich an eine festgelegte Form der Datenübergabe halten. Der Mehraufwand, welcher hierfür notwendig ist, wird durch die Ersparnis beim Zurückgreifen auf die Komponente an anderen Stellen aufgefangen.

7 vgl. [Fischer; Spiekermann, 2006] S. 51f
8 vgl. [Zimmermann; Stark; Rieck, 2006] S. 3
9 vgl. [Oestereich, 2004] S. 8)
10 vgl. [Pree, 1997] S. 2

In letzter Zeit spricht man immer mehr von Widgets anstelle von Komponenten. Der Begriff Widget wird oft für Komponenten im Web verwendet. Hier ist, im Gegensatz zu gewöhnlichen Programmen, zu beobachten, dass die Widgets und das aufrufende Programm häufig in unterschiedlichen Programmiersprachen implementiert sind. Der Datenaustausch erfolgt über ein standardisiertes Format.

2.3 JavaScript

Die Programmiersprache JavaScript entstand aus der im Netscape Browser Navigator eingebundenen Skriptsprache LiveScript. Diese war stark an den von Sun Microsystems bereitgestellten Java-Klassen angelehnt. Die Umbenennung von LiveScript in JavaScript erfolgte laut Wenz wohl aus marketingtechnischen Gründen.[11] Programme in der Sprache JavaScript werden auf einem Server gespeichert, als Code an den aufrufenden Client übertragen und erst auf diesem ausgeführt (mobiler Code). Der Vorteil einer solchen Technik liegt darin, dass der Server im Vergleich zu serverseitig ausgeführtem Code entlastet wird. Diese Entlastung ist besonders bei hohen Zugriffszahlen und großen Berechnungen sinnvoll. Ein Nachteil des mobilen Codes besteht bei der Sicherheit des Clients. Auf diesen können durch den fremd gespeicherten und somit vorher nicht einsehbaren Code schädliche Programme übertragen werden. Dieser Nachteil führt dazu, dass auf vielen Clients die Ausführung von JavaScript deaktiviert ist. Die Ausführung der hier entwickelten Widgets ist somit nicht möglich.

2.4 Anwendungsumgebung

Diese Arbeit entwickelt Komponenten für die grafische Darstellung von Projektdaten. Die Komponenten finden in verschiedenen Anwendungsumgebungen Gebrauch. Eine Verwendungsart ist die Ausführung im Webbrowser, man spricht an dieser Stelle von webbrowserbasierenden Anwendungen. Vorteile dieser Anwendungen liegen sowohl auf Entwickler- als auch auf Anwenderseite. Der Entwickler braucht kein eigenes Front-End zu erstellen, da dieses bereits durch den Webbrowser geliefert wird. Auf Anwenderseite ergeben sich Vorteile durch das gewohnte Arbeitsumfeld im Browser. Da der Zugriff auf die Anwendung über einen Server erfolgt, entfällt eine Installation auf dem Client des Anwenders. Weiterhin können die Komponenten in alle auf Webtechniken wie HTML basierenden Anwendungen eingebunden werden.

Die entwickelten Komponenten sollen in der PAVONE Process Control Suite der PAVONE AG in Paderborn einsetzbar sein. Mit Hilfe der Process Control Suite können Geschäftsideen in eine IT

11 vgl. [Wenz, 2001] S. 18

Infrastruktur umgesetzt werden. Schwerpunkte bilden dabei die Themen integriertes Prozess- und Projektmanagement, Kundenbeziehungen, Mailmanagement und Office Management.[12] In der PAVONE Process Control Suite werden verschiedene Arten von Projektdaten verwaltet. Die Abbildung der Daten erfolgt bisher bis auf wenige Ausnahmen textbasiert.

12 vgl. [PCS, 2008]

3 Konzepte / Methodik

3.1 Anforderungen an die Lösung

In Absprache mit der PAVONE AG sollen drei verschiedene Widgets zur Visualisierung von Projektdaten erstellt werden. Die Ermittlung und Auswahl dieser Komponenten wird anhand ihres Nutzens in der PAVONE Process Control Suite getroffen. In dieser werden Projektdaten oft als Text dargestellt. Durch eine grafische Darstellung soll die visuelle Erfassung dieser Gegebenheiten verbessert werden.

Eine der zu entwickelnden Komponenten ist ein Zeitstrahl zur Verdeutlichung des Projektfortschritts. Projekte besitzen einen bestimmten Zeitrahmen. Der zeitliche Fortschritt lässt sich anhand eines Zeitstrahls mit entsprechenden Markierungen der Ereignisse erkennen. Dafür soll eine Grafik mit gegebenen Anfangs- und einem Enddaten erzeugt werden. Das aktuelle Datum wird aus dem System ausgelesen und auf dem Zeitstrahl angezeigt. Die Möglichkeit während des Projekts weitere Ereignisse anzuzeigen, soll gegeben sein.

Im Verlauf von Projekten fallen zu bestimmten Zeitpunkten Kosten an. Um den Verlauf dieser zu zeigen, wird eine Grafik mit einem Verlauf der bisherigen Kosten angestrebt. Alle Zeitpunkte zu denen Kosten angefallen sind, werden in dieser Komponente mit ihren jeweiligen Daten veranschaulicht. Die Begrenzung der Projekte durch ein bestimmtes Budget wird durch eine eingezeichnete Gerade dargestellt. Weiterhin soll eine Vorschaugrafik angeboten werden, welche eine grobe Einschätzung des Verlaufs ermöglicht.

Um Anteile eines Ganzen zu zeigen, eignet sich das Kreisdiagramm. Hier kann beispielsweise der Aufwand verschiedener Projekte einer Abteilung verglichen werden. Weitere übergebene Daten, welche nicht direkt in der Grafik dargestellt werden, sollen beim Überfahren des jeweiligen Abschnitts mit der Maus in einem Pop-Up angezeigt werden. Hierbei handelt es sich um ein über der eigentlichen Grafik schwebendes Fenster.

Alle zu entwickelnden Komponenten sollen hinsichtlich ihrer Wartbarkeit, Erweiterbarkeit und Konfigurierbarkeit optimiert werden. Die Wartbarkeit zeigt an, mit welchem Aufwand und mit welchem Erfolg Änderungen an einem Zustand der Software durchgeführt werden können. Um diesen Aufwand möglichst gering zu halten, wurde der Quellcode vollständig dokumentiert und auf die Verwendung von Funktionen zurückgegriffen. Hierdurch wird eine zentrale Wartung innerhalb der Komponenten ermöglicht. Dieses Vorgehen bietet anderen Entwicklern die Gelegenheit, die

Komponenten nach ihrer Fertigstellung um neue Funktionen zu erweitern, bzw. auf neue Situationen anzupassen. Sinn der Konfigurierbarkeit ist es, die entwickelten Komponenten an verschiedenen Stellen und für verschiedene Zwecke nutzen zu können. Hierfür ist eine Anpassung an den jeweiligen Verwendungszweck notwendig. Dieses wird durch den Gebrauch von Konfigurationsdateien realisiert. Der Entwickler kann das Verhalten und Aussehen der Widgets an unterschiedlichen Stellen, jeweils dem Zweck entsprechend, separat bestimmen.

Für die Mehrsprachigkeit der Widgets wird eine ähnliche Technik verwendet. Für jede Sprache kann eine eigene Datei angelegt werden. So wird nicht nur eine Verwendung von unterschiedlichen Sprachen, sondern auch der Einsatz von verschiedenen Begriffen innerhalb einer Sprache möglich gemacht. Beispielsweise kann die Beschriftung der Geraden im Kostenverlauf unterschiedlich gewählt werden.

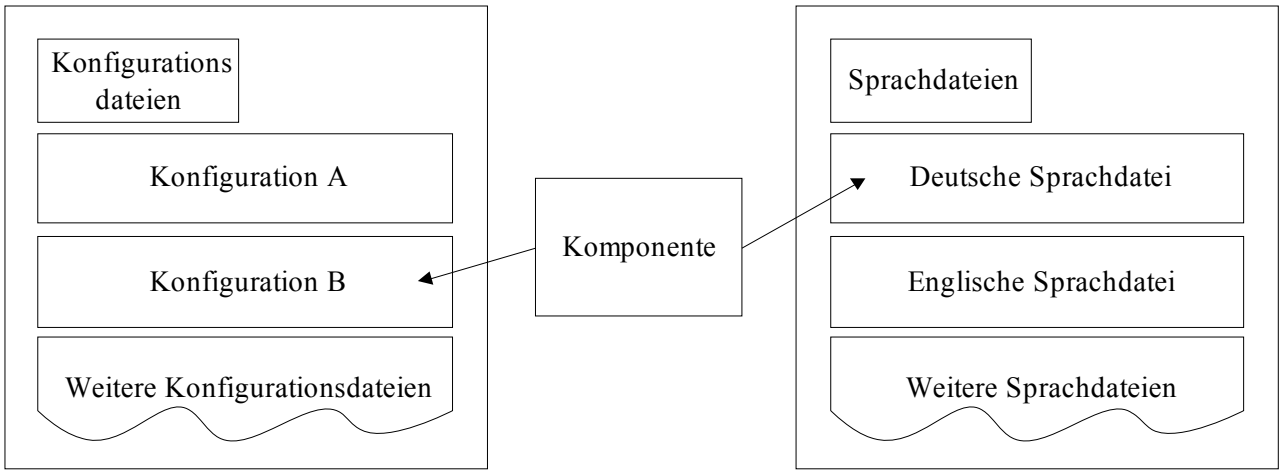

Abbildung 1: Komponente im Zusammenhang mit Konfigurations- und Sprachdateien

In Abbildung 1 ist dargestellt wie die Komponenten sich mit den verschiedenen Konfigurations-dateien verbinden lassen.

3.2 Auswahl eines geeigneten JavaScript Frameworks

Für die Erzeugung der Grafiken soll auf ein Webframework zurückgegriffen werden. Frameworks sind wichtige Bausteine der komponentenorientierten Entwicklung von Software. Laut Definition von Wolgang Pree entspricht ein Framework einer Sammlung verschiedener, individueller Komponenten mit definiertem Kooperationsverhalten und soll eine bestimmte Aufgabe erfüllen[13]. Durch diese vorgefertigten Bibliotheken wird ein erheblicher Entwicklungsaufwand eingespart. Mit

13 vgl. [Pree, 1997] S. 7

der Einführung immer größerer Interaktivität in Webseiten bzw. webbasierten Anwendungen sind zahlreiche Frameworks erschienen, welche die hierzu benötigten Techniken unterstützen. Eine Übersicht der verbreitetsten Webframeworks wird auf der Webseite ajax-info.de[14] gegeben.

Es werden verschiedene Frameworks zur Verwendung in dieser Arbeit in Betracht gezogen. Voraussetzung für die Betrachtung ist die Unterstützung von grafischen Darstellungen. Die Frameworks MooTools[15], Ruby on Rails[16] und Dojo Toolkit[17] haben spezielle Bibliotheken für diese Funktion.

MooTools ist eine Kombination der JavaScript-Bibliotheken Moo.Fx (Effekte), Moo.Ajax (asynchrones Laden) und Moo.Dom (Schnittstelle für den Zugriff auf HTML- und XML-Elemente). Für die Untersuchungen in dieser Arbeit wurde die aktuelle Version 1.2 gewählt.

Ruby on Rails basiert auf der Skriptsprache Ruby. Diese ist eine objektorientierte Sprache, welche sich aus Elementen der Sprachen Perl, Smalltalk, Eiffel, Ada und Lisp zusammensetzt.[18] Zusätzlich zu JavaScript werden in Ruby on Rails auch noch weitere Techniken wie HTML, CSS oder Datenbankmanagement unterstützt.[19] Die erste Version erschien im Dezember 2005, aktuell ist Version 2.1 veröffentlicht.

Das Dojo Toolkit ist ein freies in JavaScript geschriebenes DHTML-Toolkit. DHTML bezeichnet die dynamische Veränderung HTML-Elementen. Die Version 1.0 des Dojo Toolkits erschien im November 2007. Aktuell steht die Version 1.1 zum Download bereit. Das Dojo Toolkit entstand mit Hilfe von namhaften Unternehmen, welche Codeteile gespendet haben. Zu den Unterstützern der Dojo Foundation zählen unter anderem IBM, Sun, Sitepen, AOL, JotSpot und OpenLaszlo.[20] Diese Unternehmen nutzen das Dojo Toolkit teilweise in ihren kommerziellen Anwendungen. IBM setzt es beispielsweise in seiner Social Software IBM Lotus Connections ein. Weiterhin soll es ab dem Release 8.5 in IBM Lotus Domino integriert sein.

Die Unterstützung der Grafikerzeugung wurde für jedes der Frameworks einzeln betrachtet. MooTools hat eine eigene Bibliothek namens MooFx für visuelle Effekte. Diese beschränkt sich allerdings auf die Manipulation von HTML-Elementen. Somit ist die Erstellung eigener Klassen zur Generierung von dynamischen Grafiken notwendig. Ruby on Rails liefert diese Klassen ebenfalls

14 s. [ajax-info, 2008]
15 s. [MooTools, 2008]
16 s. [Ruby on Rails, 2008]
17 s. [Dojo Toolkit, 2008]
18 vgl. [About Ruby, 2008]
19 vgl. [Morsy; Otto, 2008] S. 27ff
20 vgl. [About Dojo Foundation, 2008]

nicht mit. An dieser Stelle lässt sich aber JFREE Chart[21] einbinden. Hierbei handelt es sich um eine Open Source Java Chart Bibliothek. Durch diese wird der Aufwand für eine eigene Erstellung eingespart. Das Dojo Toolkit liefert eine komplexe Bibliothek zur Erzeugung von Grafiken mit. In ihr befinden sich neben Beispieldateien auch vorgefertigte Widgets, welche als Grundlage für neu zu erstellende Komponenten genutzt werden können.

Ein weiteres Kriterium bezüglich der Auswahl eines geeigneten Frameworks ist die Unterstützung der verbreitetsten Webbrowser. Die Firma NET APPLICATIONS erstellt Statistiken über die Nutzung des Internets.[22] Laut einer Untersuchung dieser Firma für den bisherigen Teil des Jahres 2008 ist der Microsoft Internet Explorer (ca. 74% Marktanteil) der verbreitetste Browser. Weiterhin zu beachten sind der Mozilla Firefox (ca. 18% Marktanteil) und der Safari von Apple (ca. 6% Marktanteil).[23] Alle drei betrachteten Frameworks wurden positiv auf die Unterstützung dieser Browser geprüft.

Um einen möglichst schnellen Seitenaufbau zu gewährleisten, spielt auch die Größe der zu ladenden Frameworks eine wichtige Rolle. Je größer das Framework desto länger die Ladezeit der Seite. MooTools ist ein sehr schlankes Framework, welches im unkompriemiertem Zustand eine Dateigröße von 90 kb hat. Für Ruby on Rails lässt sich die Größe nicht pauschalisieren, da die Funktionen aus verschiedenen Paketen geladen werden. Das Dojo Toolkit belegt im unkomprimiertem Zustand 260 kb.

Um diese Frameworks nutzen zu können, ist es weiterhin wichtig die Lizenzmodelle zu betrachten. MooTools und Ruby on Rails sind unter der Open Source MIT License[24] veröffentlicht. Diese beinhaltet unter anderem das Recht, die Software zu verändern und zu verbreiten. Die Entwickler vom Dojo Toolkit bieten ihr Framework unter Academic Free License Version 2.1[25] und unter der BSD-Lizenz[26] für freie Software an. Beide Lizenzmodelle erlauben die Verbreitung und Veränderung des Quellcodes. Die BSD-Lizenz eignet sich besonders für kommerzielle (proprietäre) Projekte, da sie den Entwickler nicht zwingt, den veränderten Quellcode zu veröffentlichen.

Da diese Arbeit auch die Erstellung eigener Widgets umfasst, wurde die Dokumentation der verschiedenen Frameworks betrachtet. MooTools ist umfangreich dokumentiert, es existieren, neben der Dokumentation der Entwickler auf der Webseite von MooTools, sowohl Bücher als auch zahlreiche Artikel und Forenbeiträge im Web. Ähnlich verhält es sich bei Ruby on Rails. Für das

21 s. [JFREE, 2008]
22 s. [NET APPLICATIONS, 2008]
23 s. [Statistik BV, 2008]
24 s. [MIT License, 2008]
25 s. [AFL V2.1, 2008]
26 s. [BSD, 2008]

Dojo Toolkit ist die Dokumentation noch sehr lückenhaft und im für diese Arbeit besonders wichtigen Bereich der grafischen Effekte noch gar nicht vorhanden. Bücher zu Versionen unter 0.9 des Toolkits können nicht verwendet werden, da mit dieser Version eine komplette Umstrukturierung des Frameworks durchgeführt wurde.

Abbildung 2 fasst den Vergleich der erwähnen Frameworks zusammen.

	MooTools	Ruby on Rails	Dojo Toolkit
Unterstützung für Grafik	Eigene Klassen notwendig	Über zusätzliche Bibliothek	Bereits integriert
Browserunterstützung	Unterstützung der verbreitetsten Browser		
Größe / Ladezeit	90 kb	keine pauschale Aussage möglich	260 kb
Lizenz	Open Source MIT License		Academic Free License Version 2.1 BSD Lizenz
Dokumentation	Ausführlich	Ausführlich + zahlreiche Bücher	kaum vorhanden

Abbildung 2: Vergleich Frameworks

In der PAVONE Process Control Suite wird bereits das Dojo Toolkit eingesetzt. Von der Einbindung eines zweiten Frameworks ist aus Gründen der Performance abzuraten, da alle Frameworks beim Seitenaufbau nacheinander geladen werden müssten. Weiterhin würde sich ein großer Teil der Funktionen überschneiden.

Das Dojo Toolkit wurde, trotz seiner lückenhaften Dokumentation, zur Verwendung in dieser Arbeit gewählt, da es eine komplexe Bibliothek für die grafische Darstellung besitzt und dem Entwickler die Entscheidung über die Veröffentlichung seines Quellcodes überlässt. Weiterhin spricht für das Dojo Toolkit, dass es bereits in der PAVONE Process Control Suite eingebunden ist. Die fehlende Dokumentation wird durch Quellcodestudium der bereits mitgelieferten Widgets ausgeglichen. Zusätzlich kann auf die Unterstützung durch das Entwicklerforum zurückgegriffen werden.

3.3 Schnittstelle zwischen Widget und einbindender Seite

Die erstellten Widgets werden in eine auf HTML-Technik basierende Seite eingebunden. Für die Funktionen müssen bestimmte Daten von der aufrufenden Seite an das Widget übergeben werden. Hierfür existieren verschiedene Möglichkeiten: Die Datenübergabe kann über einzelne Parameter oder über ein Array erfolgen. Im Widget selber müssen alle übergebenen Parameter bekannt sein. Da in den Widgets mit einer variablen Anzahl an Datensätzen gearbeitet wird, sind die Daten mit

Hilfe eines Array zu übertragen.

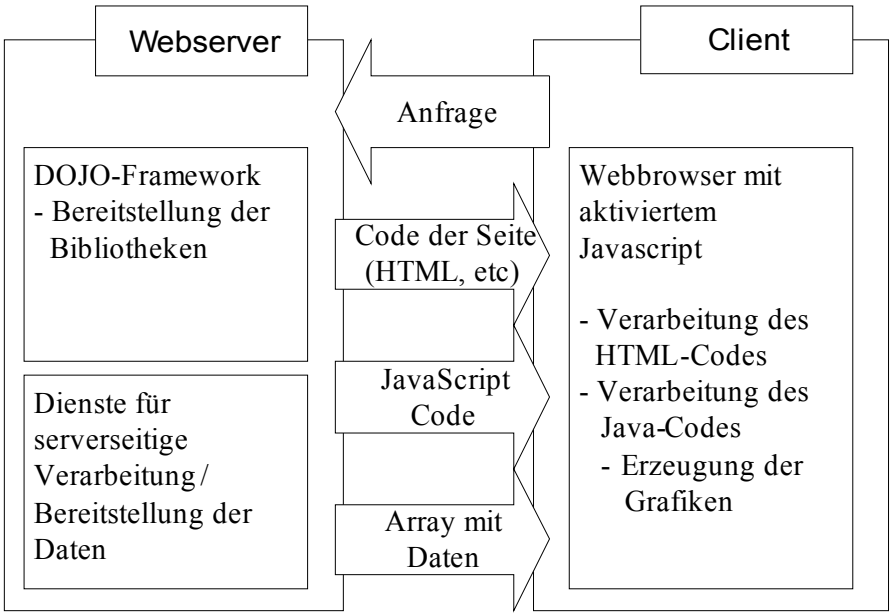

Abbildung 3: Aufgaben und Kommunikation von Client und Server

Für den Austausch von Daten hat XML (Extensible Markup Language) in den letzten Jahren stark an Bedeutung gewonnen[27]. Ein weiteres sich dafür eignendes Format ist JSON (JavaScript Object Notation). Aufgrund seiner Struktur ist es für Computer wie auch Menschen einfach zu lesen bzw. zu schreiben[28]. JSON entstand aus der Programmiersprache JavaScript, ist mittlerweile aber auch in viele andere Programmiersprachen eingebunden (Bsp.: LotusScript, PHP, C, C++, Java, Perl, Python). In diesen Sprachen existieren Module für JSON, welche die übergebenen Werte in die jeweiligen Objekte umwandeln können. Der Zugriff auf die einzelnen Eigenschaften erfolgt über einen Attributzugriff.[29] Im Vergleich zu XML besitzt JSON eine kompaktere Kodierung der Datenstrukturen (siehe Abbildung 4). Die Daten werden nur einmalig für die Übergabe an die Komponenten benötigt. Die Vorteile der langfristigen Speicherung in XML, welche in der großen Akzeptanz und Verwendung liegen, sind an dieser Stelle dementsprechend nicht relevant.

27 vgl. [Gamperl, 2006] S. 127
28 vgl. [Gamperl, 2006] S. 114ff
29 vgl. [JSON, 2008]

JSON

```
{
 "Projekt 1" : [
    { "Kosten" : 100 },
    { "Bearbeiter“ : "Frau Schmidt" }
  ],
 ”Projekt 2″ : [
    { "Kosten" : 200 },
    { "Bearbeiter“ : “Herr Meier“ }
  ]
}
```

XML

```
<xml>
 <Projekt_1>
  <Kosten>100</Kosten>
  <Bearbeiter>Frau Schmidt</Bearbeiter>
 </Projekt 1>
 <Projekt 2>
  <Kosten>200</Kosten>
  <Bearbeiter>Herr Meier</Bearbeiter>
 </Projekt_2>
</xml>
```

Abbildung 4: Vergleich JSON und XML

Abbildung 4 zeigt das mit Parametern arbeitende JSON-System im Vergleich zum XML-System mit öffnenden und schließenden Tags.

In webbasierten Anwendungen werden bei einem Großteil der Anfragen Daten zwischen einem Server und dem Client übertragen (siehe Abbildung 3). Um den Umfang dieser Daten gering zu halten und somit möglichst kurze Antwortzeiten zu erreichen, wurde das JSON-Format als Übertragungsart gewählt.

4 Prototypische Realisierung

4.1 Beschreibung der Fähigkeiten der Lösung

Die entwickelten Komponenten generieren die ihrem Bestimmungszweck zugeordneten Arten von Grafiken. Die hierfür erforderlichen Daten werden vorher entsprechend berechnet bzw. umgewandelt (siehe Abbildung 5).

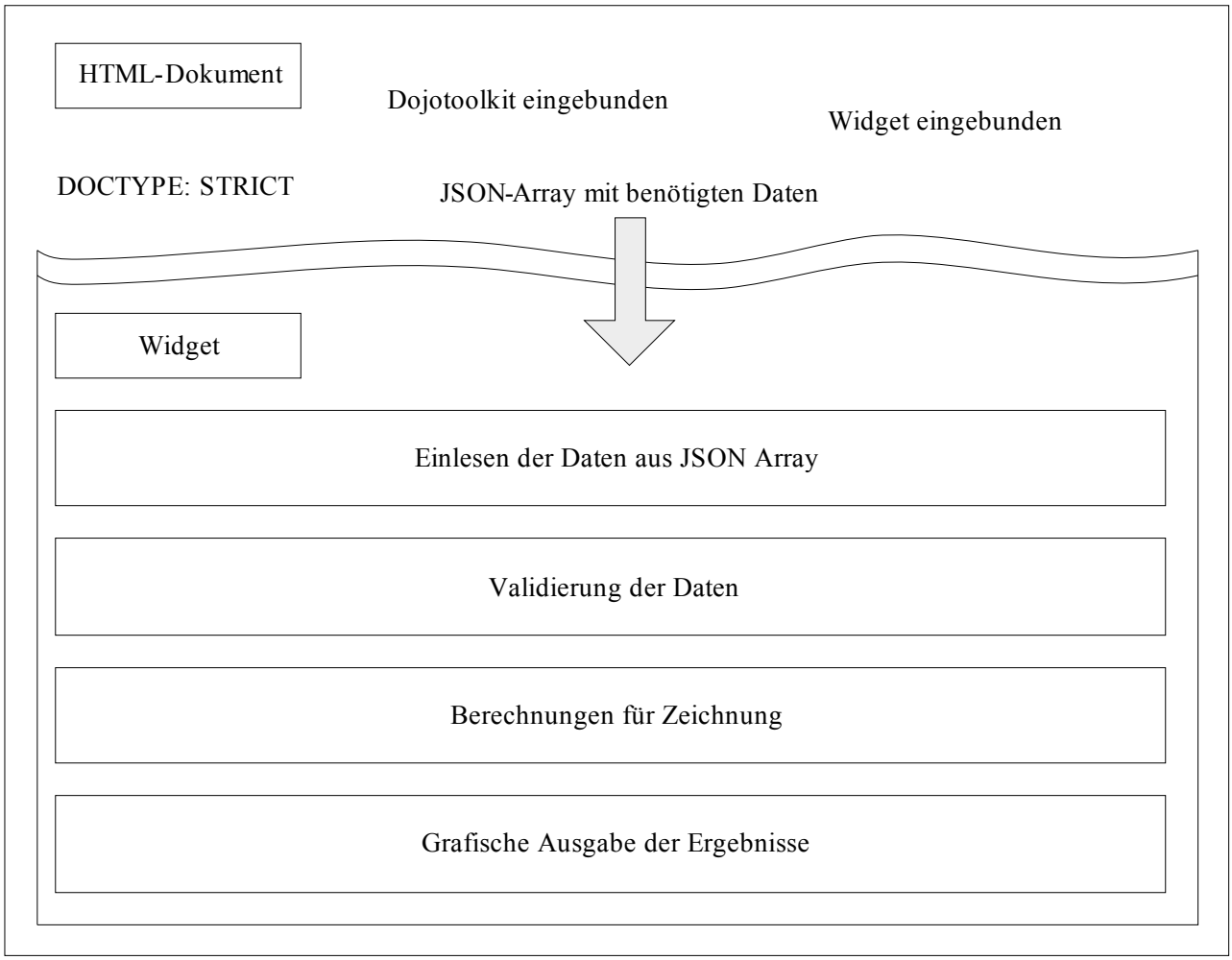

Abbildung 5: Technischer Aufbau der Widgets

Alle in dieser Arbeit entwickelten Komponenten arbeiten nach dem Prinzip, dass sie einen Abschnitt der einbindenden Seite durch einen von sich selbst generierten Teil ersetzen. Beispielsweise wird ein gewöhnlicher Text durch eine Vorschaugrafik mit einem Link, welcher ein Schwebefenster öffnet, ersetzt. Verfügt der Betrachter nicht über die vorausgesetzten Techniken (aktiviertes JavaScript, etc), wird der ursprüngliche Text angezeigt. Einziger Unterschied zu einer

Seite ohne die eingebundene Komponente ist die einmalige Prüfung, ob JavaScript vorhanden ist.

Für jedes Widget lässt sich über die Konfigurationsdatei bestimmen, ob sie direkt in die Seite eingebunden oder in einem beweglichem Schweberahmen über der Seite angezeigt werden sollen. Eine eingebaute Sperre verhindert, dass sich ein bereits geöffneter Schweberahmen nochmals öffnen lässt.

- **PAVONE widget „timeline“**

  Ein Zeitstrahl wird über das Widget „timeline“ generiert. Hier ist das aktuelle Datum in Relation zum Anfangs- und Enddatum dargestellt. Weiterhin besteht die Möglichkeit auf diesem Zeitstrahl bestimmte Ereignisse (z.B. Meilensteine) abzubilden. Eine grobe Einschätzung des zeitlichen Verlaufs wird durch die generierte Vorschaugrafik ermöglicht. Sie zeigt direkt einen eingebundenen Balken in der aufrufenden Seite, welcher je nach Projektfortschritt unterschiedlich eingefärbt wird. Fährt man mit der Maus über diesen, wird der prozentuale Fortschritt des Projekts angezeigt (siehe Abbildung 6). Dadurch wird bereits an dieser Stelle eine Prognose über den Projektfortschritt möglich, ohne dass der Zeitstrahl betrachtet werden muss.

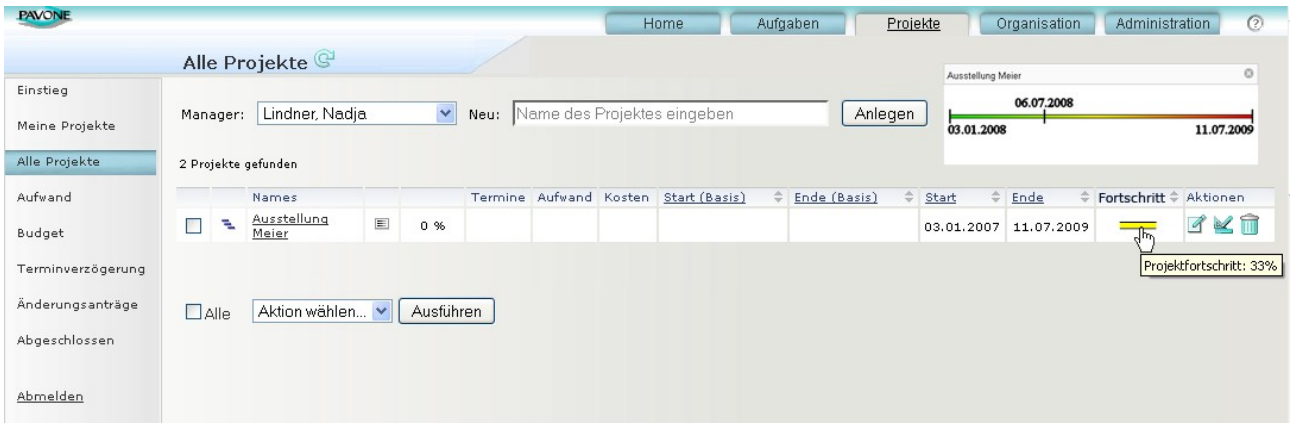

Abbildung 6: Beispieleinbindung des Widgets „timeline“ in der PCS (Fotomontage)

Betrachtet man die bisherige Darstellung von zeitlichen Abläufen in der PAVONE Process Control Suite stellt man fest, dass eine Einschätzung des zeitlichen Fortschritts kaum möglich ist. Es wird nur das Anfangs- und das Enddatum angezeigt. Besonders bei Projekten über mehrere Jahre ist eine Einschätzung des Fortschritts schwer möglich. Wird an dieser Stelle das Widget, wie in Abbildung 6 gezeigt, eingebunden, ermöglicht dies die schnelle Erfassung genauer Informationen.

- **PAVONE widget „costtrend“**

  Das Widget „costtrend“ ermöglicht die Darstellung von Kostenverläufen. Hierfür werden Datenpaare aus Zeitpunkt und Kosten übergeben. Diese Punkte werden dann in einem Koordinatensystem dargestellt. Zusätzlich wird durch ein angegebenes Budget eine Budgetgerade eingezeichnet. Darüber hinaus besteht über die Konfigurationsdatei die Möglichkeit, einen durchschnittlichen Kostenverlauf und eine hierauf basierende Prognose einzuzeichnen.

  Bei der Programmierung dieses Widgets konnte nicht auf das bereits im Dojo Toolkit vorhandene Widget zur Darstellung von Verläufen zurückgegriffen werden. Hier besteht keine Möglichkeit die Beschriftung der Achsen individuell anzupassen. Für die Darstellung von Kostenverläufen wird auf der X-Achse eine Beschriftung mit dem jeweiligem Datum benötigt. Das im Dojo Toolkit vorhandene Widget bietet bisher jedoch nur die Möglichkeit zur numerischen Beschriftung. Umgesetzt wurde das Koordinatensystem durch eine Menge von Geraden, für welche jeweils der Start und Endpunkt berechnet wird.

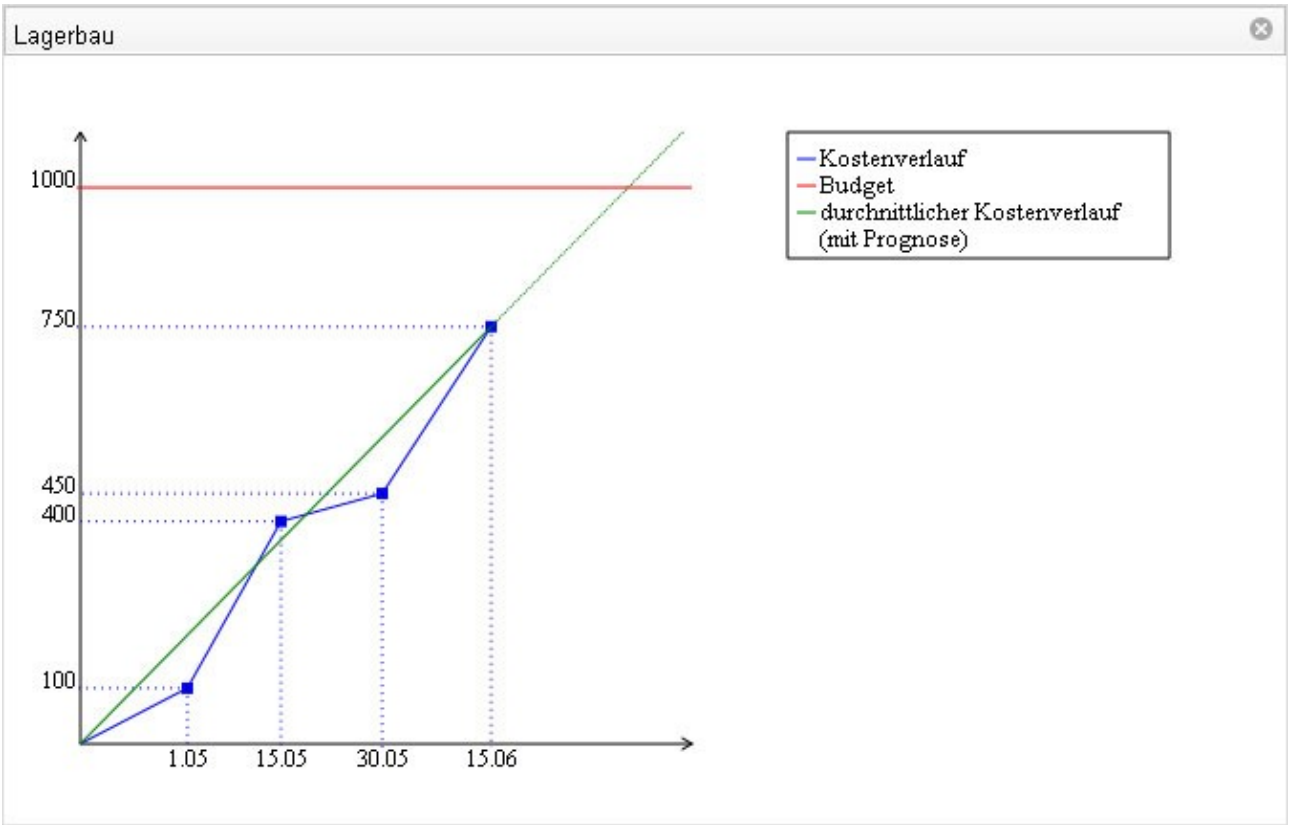

Abbildung 7: Das Widget „costtrend“ im Schweberahmen

Abbildung 7 zeigt das Widget „costtrend“: Die Blaue Linie zeigt den bisherigen Kostenverlauf eines Lagerbaus. Der durchschnittliche Verlauf wird anhand der grünen Geraden gezeigt. Nach

dem letzten eingegebene Kostenpunkt (15.06.) wird eine Prognose, basierend auf dem durchschnittlichem Kostenverlauf eingezeichnet. In diesem Beispiel lässt sich erkennen, dass das durch die rote Linie gekennzeichnete Budget, bei gleich bleibenden Durchschnittskosten, bereits vor dem Projektende aufgebraucht wird.

- **PAVONE widget „pie“**

  Das Widget „pie“ erstellt ein Kreisdiagramm, anhand dessen die verschiedenen Anteile eines Ganzen gezeigt werden. Weitere übergebene Daten sollen zusätzlich angezeigt werden können. In der Version 1.1 des Dojo Toolkit ist eine Möglichkeit zur Erzeugung von Kreisdiagrammen geben. Diese bietet allerdings keine Interaktivität, im Rahmen von zusätzlich eingeblendeten Informationen, bei Mauskontakt an. Die Informationen wurden dementsprechend für alle Teilstücke des Diagramms gleichzeitig angezeigt. Bei einer Vielfalt von Daten würde die Übersichtlichkeit verloren gehen. Für die Lösungsfindung wurden die angekündigten Funktionen der nächsten Version des Dojo Toolkits betrachtet. Die neue Version bietet in Kreisdiagrammen einen Tooltip an, welcher bei Mauskontakt gezeigt wird. Um die Funktion im entwickelten Widget anbieten zu können, wurde dieses unter der Verwendung eines Nighty-Builds des Dojo Toolkits programmiert. Hierbei handelt es sich um eine Vorabversion, welche von den Entwicklern täglich bereitgestellt wird. Die für dieses Widget zugrunde liegende Version ist das Nighty-Build vom 26. Juni 2008.

  Werden in einem Kreisdiagramm viele kleine Abschnitte angezeigt, geht die Übersichtlichkeit verloren. Um diesem entgegenzuwirken, wurde eine Option zum Zusammenfassen einer gegebenen Anzahl von Teilen des Diagramms eingebaut. Hier wird die Anzahl der kleinsten Elemente angegeben, welche im Diagramm als ein Teil angezeigt werden sollen. Dem Tooltip dieses Diagrammstückes sind dann die einzelnen Anteile zu entnehmen (Abbildung 8).

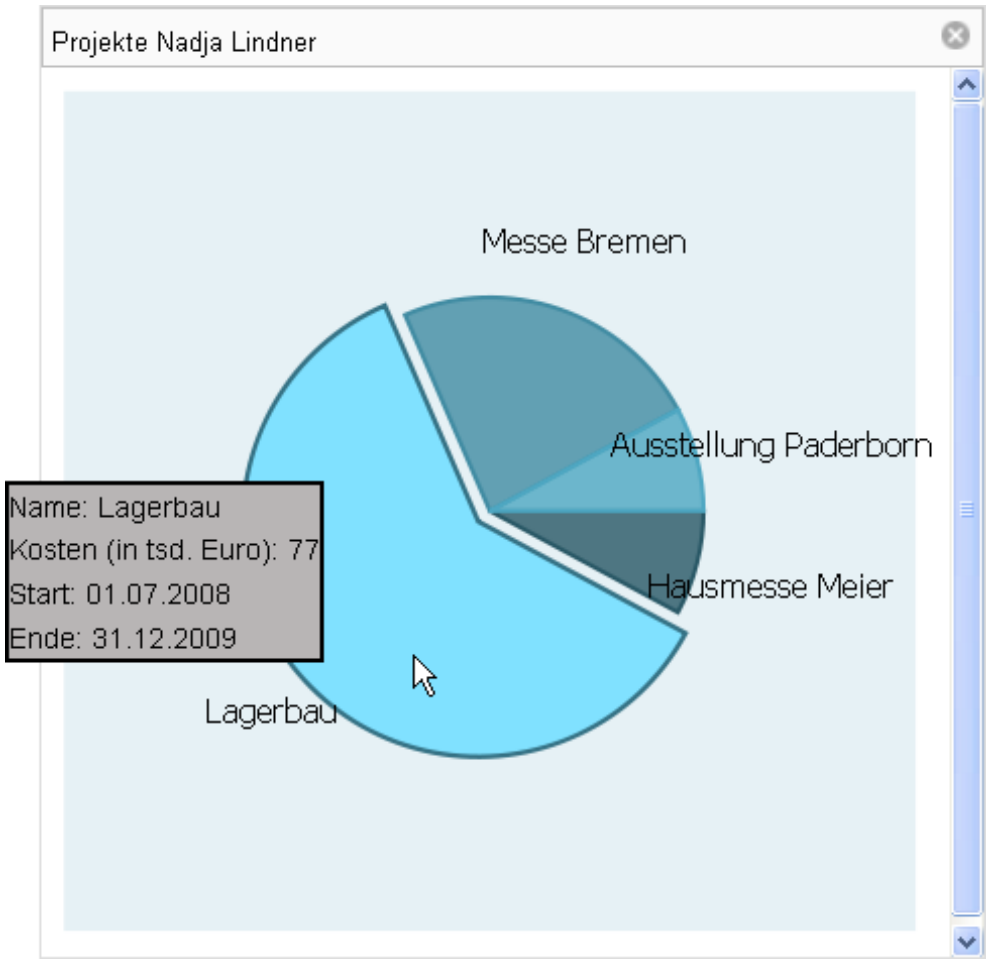

Abbildung 8: Das Widget "pie"

Vor der grafischen Darstellung werden die Teile des Diagramms anhand ihrer Größe absteigend sortiert.

4.2 Bekannte Fehler

Die Widgets wurden mit größter Sorgfalt entwickelt und zahlreichen Tests unterzogen. Bei diesen Tests wurden einige Fehler festgestellt, welche bis zum jetzigem Zeitpunkt nicht behoben werden konnten.

In allen Texten, die mit dem grafischem Paket für die Darstellung von Text erzeugt wurden, werden Umlaute nicht korrekt angezeigt. Umlaute in den Überschriften der Schweberahmen können auch nicht korrekt dargestellt werden, zusätzlich werden dem Umlaut folgende Zeichen nicht angezeigt.

Beim Widget „costtrend“ wäre für die eingetragenen Punkte auf dem Linie ein Tooltip mit den dargestellten Werten wünschenswert. Tooltips in Grafiken lassen sich mit dem Dojo Toolkit Version 1.1 nicht realisieren.

Beim Widget „pie“ wurde auf ein Nightly-Build zurückgegriffen. Diese unvollendeten Versionen verursachen bei der Ausführung oft Fehler. Sollen verschiedenen Grafiken auf einer Seite im Schweberahmen angezeigt werden, öffnen sich alle Rahmen, allerdings werden die Grafiken alle übereinander in den ersten Rahmen geladen, die restlichen Rahmen bleiben leer. Auch werden angezeigte Tooltips häufig nicht mehr geschlossen und bleiben damit permanent auf der Seite. Wird ein Schweberahmen geschlossen und wieder aufgerufen, so wird die Grafik nur bei jedem zweitem Öffnen angezeigt.

Darüber hinaus treten bei Verwendung des Microsoft Internet Explorers weitere Fehler auf. Der Text im Widget „timeline“ wird nicht angezeigt. Die Fehlersuche und Beratung im Entwicklerforum brachten hier bisher noch keine Besserung. Weiterhin gelingt die Verarbeitung der Sprachdatei mit dem Microsoft Internet Explorer derzeit nicht. Die Verarbeitung der Konfigurationsdatei, welche auf der gleichen Technik basiert, wird dagegen ohne Probleme durchgeführt. Aufgrund der Schwierigkeiten mit dem verbreitetstem Browser wird empfohlen die entwickelten Prototypen mit dem Mozilla Firefox zu betrachten.

5 Fazit

Die Untersuchungen in dieser Arbeit haben gezeigt, dass der Einsatz von Komponenten zur Visualisierung eine geeignete Möglichkeit ist, um Projektdaten darzustellen. Die entwickelten Komponenten stellen den Entwicklern eine einfache Art zur Erstellung dynamischer Grafiken zur Verfügung. Die Verwendung des Dojo Toolkits und den Widgets erspart hier erheblichen Entwicklungsaufwand. Allerdings ist der Programmierer der Widgets stark abhängig von den Entwicklern dieses Frameworks. Als Beispiel kann an dieser das Umlautproblem genannt werden. Umlaute werden bei der Ausgabe von Text in Grafiken nicht richtig erkannt und fehlerhaft dargestellt. Das Dojo Toolkit findet seinen größten Entwickler- und Anwenderbereich im englischsprachigem Raum. Da hier das Interesse an einer Behebung dieses Fehlers zweitrangig ist, ist mit einer schnellen Lösung nicht zu rechnen.

Die hier entwickelten Widgets sind Prototypen und demonstrieren die Möglichkeiten zur grafischen Darstellung von Projektdaten. Diese sollten vor ihrem Einsatz in kommerzieller Software noch erweitert bzw. verbessert werden. Eine sinnvolle Erweiterung wäre eine Funktion, welche die Größe der angezeigten Grafik und den darin enthaltenen Texten, abhängig vom zur Verfügung stehenden Platz, bestimmt.

Das Dojo Toolkit wird laufend weiterentwickelt. Fehler werden behoben, neue Funktionen und Technologien bereitgestellt. Einige neue Funktionen wurden bereits durch die Verwendung des Nightly-Builds im Widget „pie“ gezeigt. Infolge solcher Updates kann geprüft werden, ob die erstellten Widgets anhand neu zur Verfügung stehender Funktionen erweitert oder alte Funktionen ersetzt werden können.

Auch die PAVONE Process Control Suite steht in einem laufendem Entwicklungsprozess. Hier kann für neue Funktionen geprüft werden, ob bereits ein geeignetes Widgets zur grafischen Darstellung vorhanden ist oder ob ein neues Widget entwickelt werden soll.

Literaturverzeichnis

Monographien

[Alby, 2008]
Alby, Tom: Web 2.0 – Konzepte, Anwendungen, Technologien, 3. Auflage, Carl Hanser Verlag, München, 2008

[Burkhardt, 2002]
Burkhardt, Manfred: Projektmanagement – Leitfaden für die Planung, Überwachung und Steuerung von Entwicklungsprojekten, 6. Auflage, Publics Corporate Publishing, Erlagen, 2002

[Chen; Härdle; Unwin, 2008]
Chen, Chun-houh; Härdle, Wolfgang; Unwin, Antony: Handbook of Data Visialization, Springer-Verlag, Berlin Heidelberg, 2008

[Fischer; Spiekermann, 2006]
Fischer, Joachim; Spiekermann, Markus: Grundlagen von Projektmanagementsystemen, 5. Auflage, Innovation Publication, Bingen – Paderborn, 2006

[Gamperl, 2006]
Gamperl, Johannes: AJA X - Web 2.0 in der Praxis, Galileo Press, Bonn, 2006

[Morsy; Otto, 2008]
Morsy, Hussein; Otto, Tanja: Ruby on Rails 2 - Das Entwicklerhandbuch, Galileo Press, Bonn, 2008

[Oestereich, 2004]
Oestereich, Bernd: Objektorientierte Softwareentwicklung – Analyse und Design mit der UML 2.0, 6. Auflage, Oldenbourg Verlag, München Wien, 2004

[Pree, 1997]
Pree, Wolfgang: Komponentenbasierte Softwareentwicklung mit Frameworks, dpunkt-Verlag für digitale Technologie GmbH, Heidelberg, 1997

[Wenz, 2001]
Wenz, Chrisitan, JavaScript, Galileo Press, Bonn, 2001

[Zimmermann; Stark; Rieck, 2006]
Zimmermann, Jürgen; Stark, Christoph; Rieck, Julia: Projektplanung - Modelle, Methoden, Management, Springer Verlag, Berlin Heidelberg, 2006

Online-Quellen

[About Dojo Foundation, 2008]
Informationen über die Förderer des Dojo Toolkits
http://dojotoolkit.org/foundation
Abgerufen: 08.07.2008

[About Ruby, 2008]
Informationen über die Scriptsprache Ruby
http://www.ruby-lang.org/en/about/
Abgerufen: 08.07.2008

[AFL V2.1, 2008]
Text der Academic Free License Version 2.1
http://opensource-definition.org/licenses/afl-2.1.html
Abgerufen: 09.07.2008

[ajax-info, 2008]
Vergleich von verschiedenen JavaScript-Frameworks
http://www.ajax-info.de/uebersicht-ajax-frameworks/javascript-frameworks
Abgerufen: 15.08.2008

[BSD, 2008]
Text der BSD-Lizenz
http://www.opensource.org/licenses/bsd-license.php
Abgerufen: 09.07.2008

[Dojo Toolkit, 2008]
Offizielle Homepage des Dojo Toolkits
http://www.dojotoolkit.org
Abgerufen: 08.07.2008

[HTML Version 1, 2008]
Funktionsumfang der ersten HTML-Version vom 3.11.1992
http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/MarkUp.html
Abgerufen: 08.07.2008

[JFREE, 2008]
Homepage der Bibliothek zur Grafikerzeugung JFREE
http://www.jfree.org/jfreechart/
Abgerufen: 08.07.2008

[JSON, 2008]
Homepage des Übertragungsstandards für Daten JSON
http://www.json.org/index.html
Abgerufen: 09.07.2008

[MIT License, 2008]
Text der MIT Lizenz
http://www.opensource.org/licenses/mit-license.php
Abgerufen: 09.07.2008

[MooTools, 2008]
Offizielle Homepage des MooTools-Frameworks
http://www.mootools.net
Abgerufen: 08.07.2008

[NET APPLICATIONS, 2008]
Produkte der Firma NET APPLICATIONS
http://www.netapplications.com/products.aspx
Abgerufen: 08.07.2008

[OReilly, 2008]
Artikel „What is the Web 2.0“ von Tim O'Reilly, veröffentlicht am 30.09.2005
http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
Abgerufen: 15.07.2008

[PCS, 2008]
Produktbeschreibung der PAVONE Process Control Suite auf der Firmenhomepage
http://www.pavone.de/pages.nsf/goto/pcs
Abgerufen: 15.07.2008

[Ruby on Rails, 2008]
Offizielle Homepage des Ruby On Rails-Frameworks
http://www.rubyonrails.org
Abgerufen: 08.07.2008

[Statistik BV, 2008]
Statistik der Browserverteilung im Zeitraum vom 01.01.2008 bis zum Abrufdatum
http://marketshare.hitslink.com/report.aspx?
qprid=0&qpmr=100&qpdt=1&qpct=3&qptimeframe=Y&qpsp=2008&qpnp=1#
Abgerufen: 15.07.2008

Anhang

Dokumentation der Widgets

Allgemeines

Alle Widgets sind an jeder eingebundenen Stelle einzeln konfigurierbar. Dafür wird ein Parameter „cfg“ beim Aufruf übergeben. Dieser gibt den Pfad zu einer entsprechenden Konfigurationsdatei an. Hierbei handelt es sich um eine Javascriptdatei im utf-8-Format. In dieser Datei können die Parameter geändert werden. Die Funktion der Parameter ist den jeweiligen Kommentaren zu entnehmen. Gleiches gilt für die Sprachdatei welche über den Parameter „lang“ angegeben wird

Widget: Timeline

Einbindung

Im Head-Bereich der Seite, in welche das Widget eingebunden werden soll, müssen zuerst drei verschiedenen JavaScript-Dateien eingebunden werden.

- Dojo Toolkit:

```
<script type="text/javascript" src="../dojoroot/dojo/dojo.js"
djConfig="isDebug:true, parseOnLoad: true" ></script>
```

- Pavone Widget Timeline

```
<script type="text/javascript"
src="../dojoroot/pavone/timeline.js"></script>
```

- Funktionselemente aus der gemeinsamen Umgebung

```
<script type="text/javascript"
src="../dojoroot/dijit/tests/_testCommon.js"></script>
```

Im Anschluss werden die Designdateien vom Type CSS geladen. Hierbei kann es sich um die mit dem DOJO-Toolkit mitgelieferten oder um eigene Dateien handeln. Optional können noch weitere Designangaben gemacht werden.

```
<style type="text/css">
      @import "../dojoroot/dojo/resources/dojo.css";
      @import "../dojoroot/dijit/themes/dijit.css";
      @import "../dojoroot/dijit/themes/tundra/tundra.css";
      @import "../dojoroot/dijit/tests/css/dijitTests.css";
      @import "../dojoroot/dojox/layout/resources/FloatingPane.css";
                .alternateDock {
                        position:absolute;
                        background-color:#ededed;
                        right:0px; top:0px;
                        border-left:1px solid #ccc;
                        height:100%;
```

```
                }
                #alternateDock ul.dojoxDockList { display:block; }
                .testFixedSize {
                        width:300px;
                        height:200px;
                        padding:7px;
                }
</style>
```

Im Body-Bereich der Seite wird jedes Widget an entsprechender Stelle in einen div-Tag eingebunden.

```
<div dojoType="pavone.widget.timeline" id="time5"
jdata='/*JSON-Array*/' ?>'
cfg='../dojoroot/pavone/conf/timeline_cgf.js'
lang='../dojoroot/pavone/lang/de.js'>Alternativtext</div>
```

Zuerst wird im Parameter „dojoType“ das entsprechende Widget zugeordnet. Dem Paremeter id muss eine eindeutige ID übergeben werden. Das JSON-Array, mit oben beschriebener Struktur, wird per Parameter „jdata“ übergeben. Die Paramter „cfg“ und „lang“ beschreiben den Pfad zur jeweiligen Konfigurations- bzw. Sprachdatei.

Aufbau des JSON-Arrays:

```
1. jdata = array (
2.     'name'=> 'Bachelorarbeit',
3.     'startdate'=> '15.04.2008',
4.     'enddate'=> '15.07.2008',
5.     'eventdates' => array('20.05.2008', '20.06.2008'),
6.     'eventnames' => array('1. Meilenstein', '2. Meilenstein')
7. );
```

Widget: Costtrend

Einbindung

Im Head-Bereich der Seite, in welche das Widget eingebunden werden soll, müssen zuerst drei verschiedenen JavaScript-Dateien eingebunden werden.

- Dojo Toolkit:

```
<script type="text/javascript" src="../dojoroot/dojo/dojo.js"
djConfig="isDebug:true, parseOnLoad: true" ></script>
```

- Pavone Widget costtrend

```
<script type="text/javascript"
src="../dojoroot/pavone/costtrend.js"></script>
```

- Funktionselemente aus der gemeinsamen Umgebung

```
<script type="text/javascript"
src="../dojoroot/dijit/tests/_testCommon.js"></script>
```

Im Anschluss werden die Designdateien vom Type CSS geladen. Hierbei kann es sich um die mit dem DOJO-Toolkit mitgelieferten oder um eigene Dateien handeln. Optional können noch weitere Designangaben gemacht werden.

```
<style type="text/css">
      @import "../dojoroot/dojo/resources/dojo.css";
      @import "../dojoroot/dijit/themes/dijit.css";
      @import "../dojoroot/dijit/themes/tundra/tundra.css";
      @import "../dojoroot/dijit/tests/css/dijitTests.css";
      @import "../dojoroot/dojox/layout/resources/FloatingPane.css";
                .alternateDock {
                        position:absolute;
                        background-color:#ededed;
                        right:0px; top:0px;
                        border-left:1px solid #ccc;
                        height:100%;

                }
                #alternateDock ul.dojoxDockList { display:block; }
                .testFixedSize {
                        width:300px;
                        height:200px;
                        padding:7px;
                }
</style>
```

Im Body-Bereich der Seite wird jedes Widget an entsprechender Stelle in einen div-Tag eingebunden.

```
<div dojoType="pavone.widget.costtrend" id="costtrend4"
jdata='/*JSON-Array*/' ?>'
cfg='../dojoroot/pavone/conf/costtrend_cgf.js'
lang='../dojoroot/pavone/lang/de.js'>Alternativtext</div>
```

Zuerst wird im Parameter „dojoType“ das entsprechende Widget zugeordnet. Dem Parameter „id“ muss eine eindeutige ID übergeben werden. Das JSON-Array, mit unten beschriebener Struktur, wird per Parameter „jdata“ übergeben. Die Paramter „cfg“ und „lang“ beschreiben den Pfad zur jeweiligen Konfigurations- bzw. Sprachdatei.

Aufbau des JSON Arrays:

```
1. jdata = {
2.     'name'=> 'Lagerbau',
3.     'startdate'=> '15.04.2008',
4.     'enddate'=> '15.07.2008',
5.     'budget' => '1000',
6.     'dates' => array('1.05.2008', '15.05.2008', '30.05.2008', '15.06.2008'),
7.     'costs' => array('100', '300', '50', '300')
```

```
8. }
```

Widget: pie

Einbindung

Im Head-Bereich der Seite, in welche das Widget eingebunden werden soll, müssen zuerst drei verschiedenen JavaScript-Dateien eingebunden werden.

- Dojo Toolkit:

```
<script type="text/javascript" src="../dojoroot/dojo/dojo.js"
djConfig="isDebug:true, parseOnLoad: true" ></script>
```

- Pavone Widget Timeline

```
<script type="text/javascript"
src="../dojoroot/pavone/pie.js"></script>
```

- Funktionselemente aus der gemeinsamen Umgebung

```
<script type="text/javascript"
src="../dojoroot/dijit/tests/_testCommon.js"></script>
```

Im Anschluss werden die Designdateien vom Type CSS geladen. Hierbei kann es sich um die mit dem DOJO-Toolkit mitgelieferten oder um eigene Dateien handeln. Optional können noch weitere Designangaben gemacht werden.

```
<style type="text/css">
      @import "../dojoroot/dojo/resources/dojo.css";
      @import "../dojoroot/dijit/themes/dijit.css";
      @import "../dojoroot/dijit/themes/tundra/tundra.css";
      @import "../dojoroot/dijit/tests/css/dijitTests.css";
      @import "../dojoroot/dojox/layout/resources/FloatingPane.css";
                .alternateDock {
                        position:absolute;
                        background-color:#ededed;
                        right:0px; top:0px;
                        border-left:1px solid #ccc;
                        height:100%;

                }
                #alternateDock ul.dojoxDockList { display:block; }
                .testFixedSize {
                        width:300px;
                        height:200px;
                        padding:7px;
                }
</style>
```

Im Body-Bereich der Seite wird jedes Widget an entsprechender Stelle in einen div-Tag eingebunden.

```
<div dojoType="pavone.widget.pie" id="pie2" join="0" jdata='/*JSON-Array*/'
cfg='../dojoroot/pavone/conf/pie_cfg.js'
lang='../dojoroot/pavone/lang/de.js'>Alternativtext</div>
```

Zuerst wird im Parameter „dojoType“ das entsprechende Widget zugeordnet. Dem Parameter „id“ muss eine eindeutige ID übergeben werden. Das JSON-Array, mit unten beschriebener Struktur, wird per Parameter „jdata“ übergeben. Mit dem Parameter „join“ lässt sich die Anzahl der kleinsten Teilstücke bestimmen, welche zu einem Teil zusammengefasst werden sollen. Die Paramter „cfg“ und „lang“ beschreiben den Pfad zur jeweiligen Konfigurations- bzw. Sprachdatei.

Aufbau des JSON Arrays:

```
1. jdata= array (
2.          'title'=> 'Projekte Nadja Lindner',
3.          'labels' => array('Name', 'Umsatz (in tsg. Euro)', 'Start', 'Ende'),
4.          'projects' => array(
5.                array('Hausmesse Meier', '10', '12.05.2008', '16.08.2008'),
6.                array('Lagerbau', '77', '01.07.2008', '31.12.2009'),
7.                array('Messe Bremen', '30', '15.08.2008', '16.09.2008'),
8.                array('Ausstellung Paderborn', '10', '01.01.2008',
   '31.12.2008')
9.          )
10.);
```

Anmerkungen zum Array:

- „labels“ sind die Beschriftungen der Daten im Tooltip. Reihenfolge wie in „projects“
- In „projects“ werden die Projekte einzeln als Array aufgelistet. Als erstes ist eine Bezeichnung gefolgt vom jeweiligem Wert des Abschnittes anzugeben

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht.

Diese Arbeit lag in gleicher oder ähnlicher Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht.

Paderborn, 16.07.2008

.....