TANNER

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

Integration eines SVG-Viewers in eine beispielhafte ETK-Umgebung

Projektarbeit T3000

im Studiengang **Netz- und Softwaretechnik**
an der DHBW Ravensburg
Campus Friedrichshafen

Florian Peschka

13. März 2012

Bearbeitungszeitraum	KW 1 / 2012 - KW 13 / 2012
Matrikelnummer	6192194
Partnerunternehmen	TANNER AG, Lindau
Betreuer im Partnerunternehmen	Andreas Fessler

Erklärung

gemäß §*5(2)* der Studien- und Prüfungsordnung DHBW Technik vom 18. Mai 2009.

Hiermit erkläre ich, dass ich die vorliegende Arbeit mit dem Titel

Integration eines SVG-Viewers in eine beispielhafte ETK-Umgebung

selbständig angefertigt, nicht anderweitig zu Prüfungszwecken vorgelegt, keine anderen als die angegebenen Hilfsmittel benutzt und wörtliche sowie sinngemäße Zitate als solche gekennzeichnet habe.

Florian Peschka

Kurzfassung

Die Vorzüge von vektorbasierten Grafikformaten sind in der herstellenden Industrie schon lange mit dem Standard des CGM (Computer Graphics Metafile) bekannt und beliebt. Jedoch lassen sich mit diesem Format sehr schwer elektronische Online-Ersatzteilkataloge (ETK) entwickeln. Hier zeigte sich, dass SVG ein wesentlich besser unterstütztes Format in den modernen Browserumgebungen ist, das zugleich auch alle wesentlichen Anforderungen an technische Grafiken erfüllt [Pes11, Kapitel 6.1.2].

Nach der prototypischen Entwicklung eines Systems zum Anzeigen von SVG-Grafiken in modernen Browserumgebungen (vgl. [Pes11, Kapitel 4]) beschäftigt sich diese Projektarbeit mit der konkreten Implementierung dieses Systems für das ETK-System eines Kunden der TANNER AG.

Das entwickelte System zum Anzeigen der SVG-Grafiken soll die Qualität des ETKs verbessern und die darin enthaltenen Informationen auch Benutzern mit anderen Browsern als dem Internet Explorer zugänglich machen. Hierzu wird die entwickelte Beispielapplikation in die Umgebung eines bereits bestehenden ETKs eingebunden und bis zur vollen Funktionsfähigkeit mit Rücksprache zum Kunden entwickelt.

Hierbei soll neben der eigentlichen Implementierung der Bibliothek gleichzeitig festgestellt werden, ob sie sich im produktiven Einsatz bewähren kann und eventuelle Probleme aufgedeckt und gelöst werden, damit die Applikation für weitere ETKs einfacher und schneller eingesetzt werden kann.

Dies geschieht vor dem Hintergrund eines komplett geplanten Projektes im Sinne einer Weiterentwicklung eines bereits ausgelieferten Produktes. Dabei wird besonders Wert auf die korrekte Vorgehensweise der Anforderungserstellung, Testdurchführung und Abnahmekontrolle gelegt.

Abstract

The advantages of vector-based graphics have been well known and popular in the industry through the standard of CGM (Computer Graphics Metafile). But this format can hardly be implemented into electronic online spare-parts-catalogues (SPC). It was shown, that SVG (Scalable Vector Graphics) is a considerably better suited format to use in modern browser environments, while still fulfilling the basic requirements of displaying technical graphics [Pes11, Chapter 6.1.2].

After developing a prototype that could display and interact with SVG-Graphics in modern browser environments (see [Pes11, Chapter 4]), this project documentation is about the actual implementation of the developed system into an already existing SPC of a customer of TANNER AG.

The developed prototype to display SVG-Graphics is implemented to improve the quality of the SPC for users that navigate with a different browser than Internet Explorer. To achieve this, the prototype is constantly improved, revised by the customer and implemented into the publicly visible version of the SPC.

Apart from actually implementing the prototype, it should be constantly improved in terms of functionality, stability and usability. Occurring problems are to be solved and documented, so the prototype can later be implemented more easily and with less effort.

Against the background of a complete project development process, the report is focussed on the correct implementation of a software engineering process in the sense of an iterative improvement of already existing software, e. g. the definition of requirements, the functional testing process, and the approval of the resulting system by the customer.

Inhaltsverzeichnis

1 Einleitung

Ersatzteilkataloge (im folgenden „ETK“ abgekürzt) sind in großen Teilen der herstellenden Industrie ein fester Bestandteil der täglichen Arbeit. Sie kapseln alle Daten über Maschinen, deren einzelne Bestandteile und sogar die Dokumentation des Zusammenbaus in einem System.

ETKs sind sowohl für Zwischenhändler und professionelle Industrien nützlich, werden aber zunehmend auch von privaten Kunden verwendet, um für ihre Geräte Ersatzteile zu suchen und zu bestellen.

Die wesentlichen Bestandteile solcher Kataloge sind meistens Details, die eine Bestellung ermöglichen (Teilenummern, Preise, Bezeichnungen) und eine Darstellung des Ersatzteils. Diese Darstellungen sind oftmals als Explosionszeichnungen realisiert, aus welcher der Zusammenbau und die Position des Teils im Gesamtsystem klar wird.

Während früher Kataloge dieser Art fast ausschließlich in Papierform vorlagen, in welchen mühsam nach den Teilen über diverse Indizes gesucht werden musste, hat sich in den letzten Jahren ein Trend hin zu elektronischen, interaktiven ETKs gezeigt. Diese haben enorme Vorteile gegenüber ihren gedruckten Vorgängern.

So können Teile über ihren Namen, die Teilenummer oder anderen Eigenschaften (z. B. „ist Teil von Maschine...“) gefunden werden. Auch sind die Möglichkeiten zur Darstellung der Explosionsgrafiken der Maschinen durch Zoomen oder Verschieben des Bildausschnittes sehr viel komfortabler geworden. Auch können nun ETKs direkt online oder per CD ausgeliefert werden und so einer wesentlich größeren Zahl an Anwendern zugänglich gemacht werden.

Die neuesten Entwicklungen gehen mittlerweile hin zu 3D-Grafiken, die zudem animiert werden können, um den Zusammenbau und die Position der einzelnen Teile besser darstellen zu können.

Für Informationen, Prozesse und Möglichkeiten, welche für solche Kataloge relevant sind, wurde vom *Verein Deutscher Ingenieure* eine Richtlinie veröffentlicht, die „[...] die grundlegenden Anforderungen an Schnittstellen eines elektronischen Ersatzteilkatalogs (eETK) zwischen Lieferanten, Herstellern und Kunden [definiert]“ [uP06].

2 Aufgabenstellung

2.1 Inhalt dieser Projektarbeit

In dieser Projektarbeit soll der Online-ETK der Firma *ACME Inc.*[1] (im Folgenden „ACME“ genannt) erweitert werden. Die Projektarbeit umfasst dabei die Erfassung der Anforderungen an die Änderungen, die Umsetzung, Problemlösung und anschließende Diskussion von aufgetretenen Problemen und möglichen weiteren Entwicklungen des Projektes.

2.2 Ausgangssituation

ACME verwaltet insgesamt drei Versionen des gleichen Ersatzteilkataloges (ETK), die sich durch verschiedene Zugriffsstufen unterscheiden. Einer ist für jeden Kunden frei im Internet verfügbar, ein zweiter für registrierte und freigeschaltene Benutzer während ein dritter nur für interne Verwendung gedacht ist.

Die Explosionsgrafiken des ETKs konnten bis dato nur über ein spezielles Plugin[2] angesehen werden. Dieses bot alle nötigen Funktionen jedoch nur für den Windows®Internet Explorer an. Um die Usability des ETKs zu steigern und den Kundenkreis zu erweitern, musste daher eine Lösung für eine möglichst browser- und herstellerunabhängige Lösung erarbeitet werden, die das Anzeigen von komplexen SVG-Grafiken erlaubt.

Hierzu konnte auf die bereits im Vorlauf des Projekts geleistete Arbeit zurückgegriffen werden, in welcher eine Machbarkeitsanalyse und eine Technologieentscheidung getroffen wurde[Pes11, Kapitel 6.2].

2.3 Anforderungen

2.3.1 Funktionale Anforderungen

Die funktionalen Anforderungen an die Integration des SVG-Viewers wurden von ACME vorgegeben und sind für die erfolgreiche Abnahme des Projekts zwingend erforderlich. Die folgende Liste ist als Ausschnitt aus dem *Lastenheft* zu interpretieren, die für diese Dokumentation in verkürzter Form extrahiert wurde.

Anzeige

Der ETK soll browserunabhängig funktionsfähig sein.

Interaktion

Es soll möglich sein, die Grafiken zu vergrößern und zu verkleinern, sowie den Bildausschnitt zu verschieben.

Hervorhebungen in der Grafik

Es soll möglich sein, Teilenummern in den Grafiken visuell hervorzuheben.

[1] Name aus Geheimhaltsungsgründen geändert.
[2] Der Adobe®SVG Viewer: http://www.adobe.com/svg/viewer/install/

Hervorhebungen in der Teileliste

Beim Hervorheben von Teilenummern in der Grafik soll gleichzeitig auch der entsprechende Listeneintrag hervorgehoben werden.

Integration

Der neue SVG-Viewer soll sich nahtlos in das bereits bestehende System integrieren und die anderen Funktionalitäten nicht beeinträchtigen.

2.3.2 Technische Anforderungen

Aus den funktionalen Anforderungen (*Lastenheft*) wurden die folgenden technischen Anforderungen (*Pflichtenheft*) erstellt. Die Liste ist in verkürzter Form dargestellt, damit der Rahmen dieser Dokumentation nicht strapaziert wird.

Anzeige

Folgende Browser sollen in vollem Funktionsumfang unterstützt werden:

- Firefox ab Version 3.0
- Internet Explorer ab Version 7
- Safari ab Version 4
- Opera ab Version 9
- Chrome ab Version 10

Es kann vorausgesetzt werden, dass der Client JavaScript aktiviert hat und den Adobe®Flash®Player (ab Version 10) installiert und aktiviert hat.

Interaktion

Die möglichen Interaktionen, welche der SVG-Viewer unterstützen muss sind *Zoomen* und das *Verschieben des Viewports*. Hierzu soll eine Miniaturgrafik als Referenzpunkt der aktuellen Position des Viewports und der aktuellen Zoomstufe dienen.

Ebenso soll ein *Zurücksetzen*-Button implementiert werden, der Zoomstufe und Viewport wieder auf ihre Ausgangswerte zurücksetzt.

Hervorhebungen

Hervorhebungen sollen nur auf den textuellen Repräsentationen der Teilenummern geschehen. Es soll stets nur eine Hervorhebung gleichzeitig aktiv sein. Sowohl der Klick auf eine Teilenummer in der Grafik als auch ein Klick auf den entsprechenden Listeneintrag soll zu einer Hervorhebung in beiden Komponenten führen.

Beim Klick auf den *Zurücksetzen*-Button sollen auch alle Hervorhebungen in Grafik und Liste zurückgesetzt werden.

Integration

Um die volle Funktionalität der alten Funktionen des ETKs zu gewährleisten, sollen möglichst nur Dateien auf der Frontend-Seite des ETKs verändert, erstellt oder gelöscht werden.

2.3.3 Sonstige Anforderungen

Im Zuge der Umsetzung soll die Server-Software, welchen den ETK darstellt, auf Tomcat Version 7 umgestellt werden. Diese Anforderung hat nur indirekt mit den Anforderungen an den SVG-Viewer zu tun, muss jedoch in der Umsetzung beachtet werden, da hier unerwünschte Nebeneffekte auftreten könnten.

Diese Anforderung wird nicht in dieser Projektarbeit behandelt, da sie von einem anderen Mitarbeiter der TANNER AG übernommen wurde.

2.4 Projektplanung

Für die Planung des Projektes wurden im Vorfeld bereits mehrere Meetings mit ACME abgehalten, in welchem die oben definierten Anforderungen festgehalten wurden. Der Projektablauf wurde mit ACME besprochen und abgenommen. Hiernach ist vorerst ein Prototyp zu entwickeln, der die Möglichkeiten des SVG-Viewers im Zusammenhang mit dem bestehenden System des ETKs zeigt und die Kombination der beiden als funktionierend beweist.

Weiterhin soll folgend an die Abnahme des Prototypen ein erster Entwurf entwickelt werden, für den alle oben gezeigten Anforderungen erfüllt werden. Dieser Entwurf wird dann iterativ mit Rücksprache mit ACME weiterentwickelt und führt gegebenenfalls zur erfolgreichen Abnahme des Projekts.

3 Umsetzung

3.1 Eingesetzte Technologien

3.1.1 Scalable Vector Graphics

Scalable Vector Graphics (SVG) bezeichnen eine „Sprache zur Beschreibung zweidimensionaler Grafiken in XML“[Con03, Kapitel 1.1]. Es wurde vom World Wide Web Consortium (W3C) als Recommendation veröffentlicht und somit für den allgemeinen Gebrauch zugänglich gemacht. Besonders im Internet werden SVG-Grafiken immer häufiger verwendet und erlauben es somit, immer aufwändigere und gleichzeitig visuell ansprechendere Applikationen im Web-Umfeld zu realisieren.

Grund für die immer weiter voranschreitende Verbreitung vektorbasierter Grafiken ist, dass diese eine semantische Beschreibung von Grafiken zur Verfügung stellen, die im Gegensatz zu rasterbasierten Formaten eine Interpretation auch auf technischer Ebene erlauben.

Weiterhin wird durch eine beschreibende Art der Darstellung sichergestellt, dass die Ortsauflösung eines Bildes irrelevant wird und beliebige Zoomstufen mit der gleichen Schärfe und Exaktheit dargestellt werden können, wie im Ausgangsformat.

ACME verwendet seit langem zur Darstellung von Explosionsgrafiken von Maschinen und Bauteilen das SVG-Format. Diese Explosionsgrafiken werden dafür verwendet, Ersatzteile von Maschinen zu nummerieren und zu identifizieren, um sie z. B. nachbestellen zu können.

Diese Grafiken sollen auch weiterhin im ETK zur Verwendung kommen, möglichst ohne eine manuelle Änderung daran vornehmen zu müssen.

3.1.2 JavaScript

JavaScript ist eine auf ECMA-Script basierende objektorientierte Programmiersprache, die Berechnungen und Manipulationen von Objekten in einer Wirtsumgebung durchführen kann[Int11].

Diese in nahezu allen modernen Browsern standardmäßig aktivierte Programmiersprache[Ham11] erlaubt es, Seiteninhalte dynamisch ohne erneutes Laden der Seite in jeglicher Weise zu manipulieren.

Auch kann dadurch auf Benutzeraktionen (Klicken auf Elemente, Bewegen der Maus, Eintippen von Text) reagiert werden, was die Entwicklung einer sehr reaktiven Oberfläche ermöglicht. Zudem wird wie in der Prototypenerstellung bereits erläutert[Pes11], das JavaScript-Framework *jQuery* eingesetzt, um die Kompatibilität zu sichern und eine vereinheitlichte Schnittstelle bereitzustellen.

Nachteil des Einsatzes von JavaScript für die Umsetzung dieses ETKs ist, dass Benutzer in ihren Browsern die Option haben, JavaScript teilweise oder vollständig zu deaktivieren – was nach Absprache mit ACME jedoch als aktiviert vorausgesetzt werden kann und damit als Risiko für das Projekt nicht eingeplant werden muss.

3.1.3 Flash

Die von Adobe®entwickelte Flash®-Technologie ermöglicht das Entwicklent von Rich Internet Applications (RIAs). Dies bezeichnet Programme, die im Browser ablaufen und der Komplexität und Bedienfreundlichkeit von klassischen Desktopanwendungen in nichts nachstehen sollen[Inc12].

Flash®basiert, wie auch SVG, auf Vektorgrafiken, die in einem eingebetetten Element innerhalb der HTML-Struktur angezeigt werden können. Die die Verbreitung des Flash®-Players derer der nativen Unterstützung von SVG weit voraus ist[Pes11], wird für die Umsetzung des ETKs die Verwendung von Flash der von nativem SVG bevorzugt.

3.1.4 HTML

Hyper Text Markup Language (HTML) ist die „[...] universal verständliche Sprache, eine Art Muttersprache, die von allen Computern potenziell verstehen“[RHJ99, Kapitel 2.2]. Sie bildet die Basis nahezu jeder Internetseite und ist auch der Hauptbestandteil der sichtbaren Oberfläche des ETKs von ACME.

Die HTML-Struktur des ETKs ist bereits seit der Auslieferung des Gesamtsystems vorgegeben und kann gegebenenfalls durch die Anpassungen, die zur Implementierung des SVG-Viewers nötig sind, angepasst werden.

3.1.5 JavaServer Pages

Die bereits in 3.1.4 erwähnte HTML-Struktur des ETKs besteht nicht nur aus statischen HTML-Seiten. Vielmehr werden diese durch eine von Sun®Microsystems entwickelte Technik, die „beschreibt, wie eine Anfrage in eine Antwort umgewandelt werden soll. Die Beschreibung vermischt Vorlagendaten mit dynamischen Aktionen und verwendet die Java™-Plattform.“[DLR06, Kapitel „Overview“]

Diese Technologie bildet die Verbindung zwischen dem sichtbaren Teil des ETKs (HTML) und dem darunter liegenden Java™-System, welches die Datenhaltung in der Datenbank überwacht, sowie die Geschäftslogik verwaltet.

3.2 Rahmenprogramme

3.2.1 SVGWeb

Um sicherzustellen, dass möglichst viele Benutzer des ETKs in der Lage sind, die enthaltenen SVG-Grafiken anzusehen und die volle Funktionalität des ETKs nutzen können, wird eine Funktionsbibliothek verwendet, die den Zugriff auf die Elemente der SVG-Grafik kapselt und vereinheitlicht. Zugriff auf die Elemente werden dann über diese Bibliothek übersetzt und entsprechend ausgeführt.

Der Vorteil von SVGWeb liegt vor allem darin, dass es eine bereits implementierte Flash®-Schnittstelle bereitstellt, die das Ersetzen der SVG-Grafik durch einen Flash®-Wrapper regelt, der die Grafik anstelle des Originals darstellt und alle Zugriffe auf diesem Wrapper ausführt.

So kann die SVG-Grafik auch auf Browsern, die keine eigene Implementierung des SVG-Standards haben, dargestellt und mit allen Funktionalitäten verwendet werden.

3.2.2 TANNER ETK Environment

Für das grundlegende Aufbauen eines ETKs mit dem Einsatz von SVGWeb wurde bereits ein Protoyp erstellt, welcher mit dem Arbeitstitel *TANNER ETK Environment* (TetkEnv) betitelt wurde. Dieser bildet die Basis der Entwicklung für die nun folgende konkrete Umsetzung für den ETK von ACME.

Die Funktionsbibliothek bietet Zugriff auf die grundsätzlichen Funktionalitäten eines ETKs wie das Hervorheben von Bauteilen oder Teilenummern sowohl in der Grafik als auch in der Stückliste, das Zommen und Bewegen des Bildausschnittes sowie die initialisierung von Miniaturgrafiken und der SVGWeb-Bibliothek.

Für die Umsetzung des ETKs wird die Bibliothek nur als Ausgangspunkt für weitere Entwicklungen und Anpassungen verwendet, da je nach Kundenanforderungen und bereits bestehenden Umgebungen nicht davon ausgegangen werden kann, dass die prototypische Entwicklung bereits alle Eventualitäten abdeckt.

4 Ergebnisse

4.1 Bestehendes ETK-System

Das bestehende ETK-System baut auf einem Java-Tomcat-Server sowie einer JSP-Weboberfläche auf. Intern läuft ein in Java programmierter Generator, der aus den von ACME gelieferten Stücklisten die Datenbank, welche vom ETK angefragt wird, generiert.

Die Änderungen für die Implementierung des SVG-Viewers des TetkEnv belaufen sich größtenteils auf den visuellen Teil des ETK-Systems – die unterliegende Java-Ebene musste kaum geändert werden.

Die Oberfläche des ETKs ist durch mehrere Framesets (s. Abb. 1) in HTML aufgebaut, die jeweils eigenständige Seiten beinhalten, welche verschiedene Funktionen des ETKs widerspiegeln. So wird die Stückliste, die Grafik, die Navigation, die Kopf- und Fußzeilen sowie diverse Abstandshalter jeweils in eigenen Frames verwaltet.

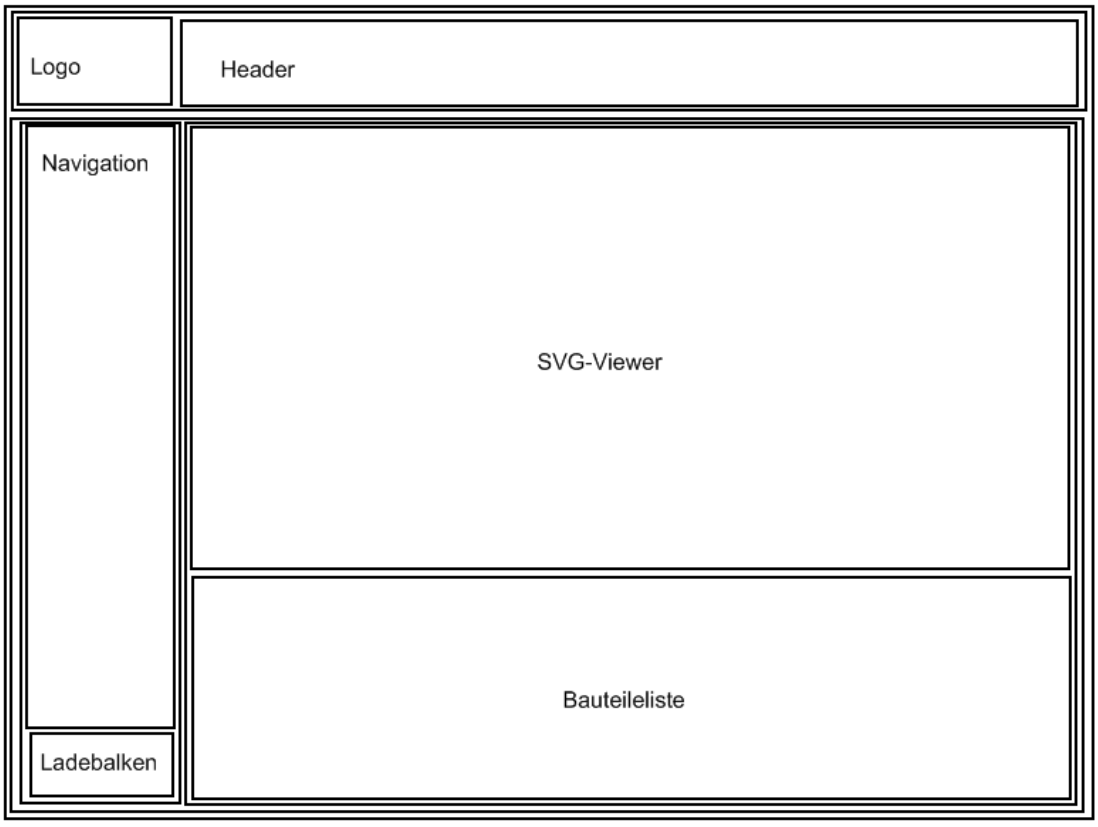


Abbildung 1: Frameset-Struktur des ETKs

Frames erlauben es, „Dokumente in mehreren Ansichten, die unabhängige Fenster oder Unterfenster [sein können]“[RHJ99, Kapitel 16.1] anzuzeigen. Dies erschwert die Anpassung an das nach modernen Kenntnissen programmierten TektEnv deutlich, da die JavaScript-Funktionen nicht ohne weiteres auf Elemente aus anderen Frames zugreifen können. Der Zustand der alten ETK-Oberfläche lässt sich jedoch auf die zur Zeit der Erstellung des ETKs aktuellen Programmierkenntnisse und -standards zurückführen.

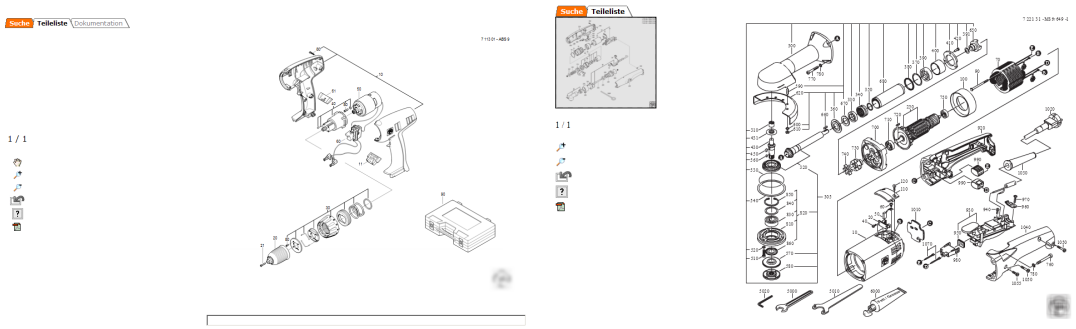
4.2 Implementierung der Rahmenprogramme

4.2.1 HTML-Struktur

Das TetkEnv wurde in die bereits bestehende HTML-Struktur des ETKs zunächst direkt eingebunden. Dafür wurde die aus der prototypischen Entwicklung hervorgegangene Bibliothek als Basis verwendet. Aufgrund der bereits aufgezeigten unterschiedlichen Struktur der tatsächlichen ETK-Umgebung und der in der Entwicklung verwendeten, mussten jedoch zahlreiche Veränderungen am Programmcode der Bibliothek vorgenommen werden.

An der Grundstruktur der HTML-Dateien des ETK wurden zunächst die alten JavaScript-Bibliotheken entfernt, die die vorherige Funktionalität in Verbindung mit dem *Adobe® SVG Viewer* ermöglicht haben. Diese stehen zwar den Funktionen des TetkEnv nicht unmittelbar im Wege, wurden aber dennoch zur Sicherheit deaktiviert; auch um die Ladezeiten zu Gunsten der neuen Bibliothek zu optimieren.

Für die neue Anzeigemethode des SVG-Viewers wurde die HTML-Struktur des Frames, in welchem die Grafiken angezeigt werden ebenfalls angepasst um Platz für das neu eingefügte Thumbnail der Grafik zu machen, welches automatisiert durch den Generator erstellt wird (s. Abb. 2).



(a) Vorher (b) Nachher

Abbildung 2: Vergleich der Basisansichten des SVG-Viewers

4.2.2 Hervorhebungen und Hotspots

Da die gelieferten SVG-Grafiken nicht über korrekt geschachtelte Gruppen verfügen, konnten nicht die im Prototypen verwendeten Algorithmen zur Hervorhebung verwendet werden. Vielmehr musste hier erneut angesetzt werden und die entsprechenden Methoden des TetkEnv neu entwickelt werden, um mit der veränderten SVG-Struktur arbeiten zu können.

Die bereits im Vornherein bei der prototypischen Entwicklung vorgenommenen Änderungen hinsichtlich der `<hotspot>`-Elemente war hierbei hilfreich, musste jedoch erneut erweitert werden. Dies hat den Grund, dass nicht mit eingeplant wurde, dass ein `<hotspot>`-Element auch mehrfach vorkommen kann, wenn das entsprechende Bauteil mehrfach in der Grafik vorhanden ist.

Die vorhandene Methode zur Erkennung der `<hotspot>`-Elemente wurde so angepasst, dass jeder Hotspot für sich beliebig oft in der Grafik vorkommen kann und er korrekt mit den entsprechenden Listeneinträgen hervorgehoben werden kann. Hierzu musste auch die Methode des Hervorhebens und der Löschung des Hervorhebens angepasst werden.

Listing 1: Neue Hervorheben-Methode

```
1 // Über alle hotspots iterieren, anstatt nur den ersten
2 $(tspanIdMapping[id]).each(function() {
3   self.highlight(id, image.svgObject.getElementById(this));
4 });
5 return;
```

Aufgrund der in 4.1 angesprochenen Frameset-Struktur musste zur Beibehaltung der bereits entwickelten Hervorhebungsmethodik der Aufruf der `init()`-Methode angepasst werden, sodass die Liste der in der Grafik vorhandenen Bauteile aus einem anderen Frame aufgerufen wird. Die ist dank der Verwendung der *jQuery*-Bibliothek sehr einfach, da nur das entsprechende `window`-element im Document Object Model (DOM) mit an die Methode gegeben werden muss.

Listing 2: Aufruf der init()-Methode

```
1 tetkenv.init({
2   // Liste befindet sich in einem anderen Frame!
3   list: $(".spclisttable", window.parent.infoTop.document),
4   ...
```

Durch den Aufruf von `window.parent.infoTop.document` lässt sich der Frame des SVG-Viewers aus dem Frame der Liste heraus identifizieren. Dies ist durch den DOM-Baum möglich, da dieser rekursiv alle Elemente der gesamten Seite enthält, und nicht etwa nur diese der Seite, in welchem sich das Skript befindet.

4.2.3 Bauteile der Grafik

In der prototypischen Entwicklung wurden die in einer Grafik vorhandenen Bauteile über ein externes Datenskript[Pes11, Kapitel 4.6] geliefert. Da ein solches Skript in der Produktivumgebung nicht existiert, wurde der entsprechende asynchrone Aufruf entfernt und durch ein manuelles Parsen der Stückliste, die vom ETK für jede Grafik geliefert wird, ersetzt.

Dieses Parsen wird nicht direkt in der Funktionsbibliothek durchgeführt, sondern im JavaScript-Block der HTML-Datei, die die aufzurufende Grafik enthält. Nachdem alle Bauteile identifiziert wurden, werden diese an die Funktionsbibliothek übergeben, die darauf hin die Elemente mit dem bereits vorhandenen *Mapping*[3] verbindet.

Listing 3: Methode zum Setzen der Bauteile einer Grafik

```
this.setParts = function(setParts, altMessage) {
  parts = setParts;

  // Alle Hotspot-Elemente selektieren
  var hotspots = image.svgObject.getElementsByTagNameNS("http://www.w3.org/2000/
      svg", "hotspot");

  // Über alle gefundenen Hotspots iterieren
  $(hotspots).each(function() {

    // Das Mapping der Teilennummern und der Hotspot-IDs erstellen
    var partNum = this.firstChild.nodeValue;
    var textElement = image.svgObject.getElementById(this.id);
    if (typeof tspanIdMapping[partNum] == "undefined") {
      tspanIdMapping[partNum] = new Array();
    }
    var length = tspanIdMapping[partNum].length;
    tspanIdMapping[partNum][length] = this.id;
    // ...
  });
```

Der zweite Teil der Methode ermöglicht das Klicken auf Bauteilenummern in der Grafik. Erst durch diese Methode lassen sich Hotspots in der Grafik anklicken, was zum Hervorheben der Bauteilenummer in der Grafik und in der Liste führt. Der Handler verwendet die gleiche Methode, die auch beim Klicken des Bauteils in der Liste aufgerufen wird.

So spielt es technisch keine Rolle, woher der Aufruf zum Hervorheben eines Bauteils stammt. Die Methode könnte auch direkt aus der JavaScript-Konsole des Browsers heraus aufgerufen werden.

[3] vgl. [Pes11, S. 38]

Listing 4: Methode zum Setzen der Bauteile einer Grafik

```
20   // Über alle gesetzten Bauteile der Liste iterieren
21   $(parts).each(function() {
22     var partNum = parseInt(this);
23     try {
24
25       // Jeden Hotspot in der Grafik mit einem "click"-handler versehen
26       $(tspanIdMapping[partNum]).each(function() {
27         var element = image.svgObject.getElementById(this);
28         element.addEventListener("click", function() {
29           tetkenv.toggleHighlight(partNum);
30         }, true);
31       });
32     } catch (e) {
33       console.log(e);
34     }
35   });
36 };
```

4.2.4 Anpassungen für Browser

Um den ETK auch weiterhin für den Windows®Internet Explorer®6 zugänglich zu machen, musste für den Hintergrund des Miniaturbildes eine neue, komplett transparente GIF-Grafik erstellt werden. Ohne diese ergab sich ein Effekt, der beim Versuch, den Bildausschnitt zu verschieben, die Maus „durch“ den bewegbaren Ausschnitt klicken lies, anstatt ihn festzuhalten. Da Internet®Explorer 6 halbtransparente Grafiken nicht korrekt anzeigen kann[Cor07], musste eine Alternative im GIF-Format erzeugt werden.

Durch das Hinterlegen einer transparenten GIF-Grafik wurde dieses Problem behoben. Der einzige Nachteil ist jedoch, dass nun der bewegbare Teil der Miniaturgrafik komplett transparent ist und keine deutliche visuelle Abgrenzung mehr zum eigentlichen Miniaturbild herstellen kann (s. Abb. 3).

4.2.5 Vereinfachung der Publikationsstrecke

Die automatische Verarbeitung der SVG-Grafiken durch die JavaScript-Bibliotheken ermöglicht die direkte Verwendung der Grafiken in ihrer nativen Form, die durch den Generatorprozess erstellt wird. Dies vereinfacht die Publikation neuer Versionen des ETKs deutlich, da im Nachhinein keine manuelle Änderung an den SVG-Grafiken vorgenommen werden muss.

Dies hat jedoch auch den entscheidenden Nachteil, dass die Ladezeiten des SVG-Viewers deutlich beeinträchtigt werden, da mehrfach komplexe Operationen auf die Grafiken durchgeführt werden müssen. Dies ist aufgrund der Komplexität der Grafiken mit deutlichen Geschwindigkeitseinbußen verbunden.

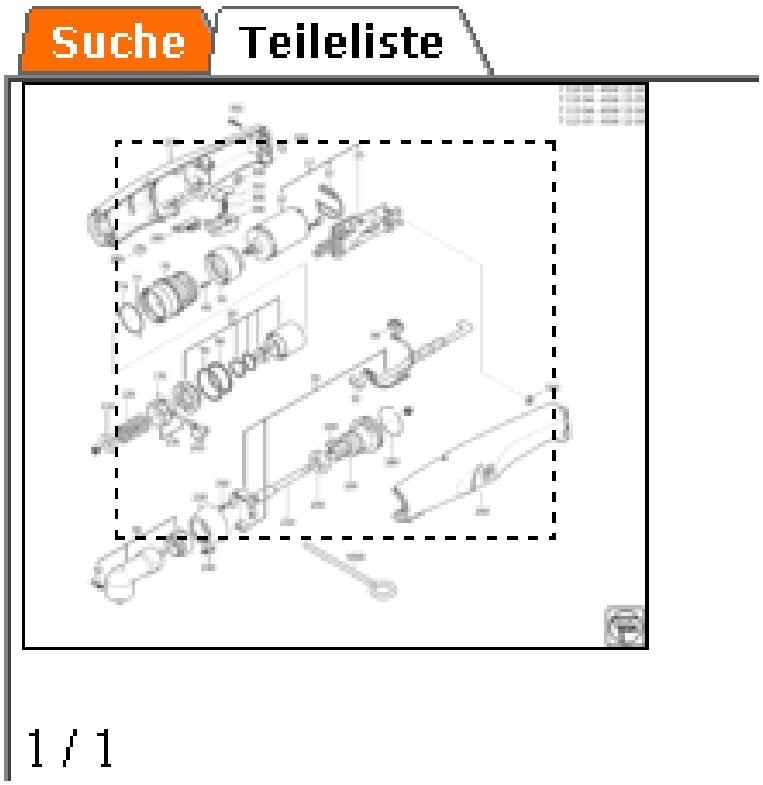


Abbildung 3: Internet Explorer 6 GIF-Ersatz

4.3 Erweiterung der Funktionalität

4.3.1 Begründung

Während der iterativen Implementierung der Funktionsbibliothek wurden von ACME mehrere, nicht vorhergesehene Änderungswünsche angebracht, die mit in die finale Version des SVG-Viewers implementiert werden sollten. Diese konnten in der prototypischen Entwicklung nicht vorbereitet werden und mussten somit während der eigentlichen Implementierung umgesetzt werden.

4.3.2 Prüfung der Flashversion

Um Benutzer, die eine zu niedrige Version von Flash oder gar keine installiert haben, darauf hinzuweisen, dass sie den ETK nicht verwenden können, musste eine Prüfung auf die installierte Version eingebaut werden. Diese wird bei jedem Seitenaufruf ausgeführt, bevor die SVG-Grafik geladen wird (s. Abb. 4).

Abbildung 4: Warnmeldung bei nicht aktiviertem Flash

4.3.3 Größenveränderung der Grafik

Da der SVG-Viewer in einem Frame eingebettet ist, muss damit gerechnet werden, dass der Benutzer die Größe dieses Frames verändern kann. Damit kein vertikaler Scrollbalken entsteht, wird die Größe der SVG-Grafik an die Größe des enhaltenden Frames gekoppelt. Wird der Frame in seiner Größe verändert, wird diese Veränderung auf die Grafik übersetzt.

Dadurch kann die Grafik jedoch sehr klein werden, wenn der Frame entsprechend klein gezogen wird. Daher wurde hier ein Schwellwert definiert, ab dem die Grafik nicht weiter verkleinert werden kann. Nur in diesem Fall wird dann der vertikale Scrollbalken angezeigt.

Listing 5: Methoden zum Auslesen der Framedimensionen

```
1  window.onresize = function(event) {
2      var height = $(window).height();
3      var newHeight = height - 60;
4      if (height > 500) {
5          $("#SVGImage").height(newHeight);
6          $("#SVGImage").width(newHeight * svgRatio);
7      } else {
8          $("#SVGImage").height(540);
9          $("#SVGImage").width(540 * svgRatio);
10     }
11 };
```

4.3.4 Selektive Tooltips für Bauteile

In der alten Version des ETKs konnten die Daten von Bauteilen in einem Tooltip angezeigt werden, wenn mit der Maus darüber gefahren wurde. Dies war im Standardumfang des TETKEnv nicht vorhanden und musste nachgerüstet werden. Dafür liefert der ETK den kompletten Text des Tooltips aus der Datenbank. Die Funktionsbibliothek musste insoweit angepasst werden, als dass diese Tooltips an ihren jeweiligen Bauteilenummern sichtbar wurden und der Mausbewegung folgten.

Dazu mussten zuerst alle relevanten Bauteile aus der Liste der Bestandteile ausgelesen werden. Nur die Bauteile, die sich auch in der Liste befinden, durften mit einem Tooltip versehen werden, der den entsprechenden Text der Datenbank enthält. Für alle Bauteile, die nicht in der Liste auftauchen, ist ein Standard-Text vorgesehen, der den Benutzer darauf hinweist, dass das Bauteil nicht verwendet wird.

Listing 6: Methoden zum Darstellen des Tooltips

```
var mouseX, mouseY;

// Erkennen der aktuellen Position der Maus auf der Seite
$(document).mousemove(function(event) {
  if ($.browser.msie)
  {
    mouseX = event.clientX;
    mouseY = event.clientY;
  }
  else
  {
    mouseX = event.pageX;
    mouseY = event.pageY;
  }
});

// Zeigt den Tooltip des Bauteils mit der Nummer "num" an
this.hoverElement = function(num) {
  var tooltip = $("#tooltip");

  // Wenn "num" eine Zahl ist, wird der Tooltip des entsprechenden Bauteils
  // in den Tooltip geschrieben
  if (typeof num == "number")
    tooltip.text(eval("itemInfo.tooltip" + num));

  // Wenn "num" keine Zahl ist, wird der Tooltip mit dem Inhalt von "num"
     gefüllt
  else
    tooltip.text(num);
  tooltip.show();
  tooltip.offset({ left: mouseX + 10, top: mouseY });
};
this.hoverElementOut = function(num) {
  $("#tooltip").hide();
};
```

Um den Tooltip beim Überfahren mit der Maus anzeigen zu können, müssen den Text-Elementen der Hotspots so genannte `mouseover`- und `mouseout`-Handler zugewiesen werden. Diese werden ausgeführt, sobald erkannt wird, dass sich der Mauszeiger über das Element hinein bzw. daraus hinausbewegt.

Listing 7: Zuweisen der Tooltip-Handler

```
// Über alle gesetzten Bauteile der Liste iterieren
$(parts).each(function() {
  var partNum = parseInt(this);
  try {

    // Jeden Hotspot in der Grafik mit einem "click"-handler versehen
    $(tspanIdMapping[partNum]).each(function() {
      var element = image.svgObject.getElementById(this);
      element.addEventListener("mouseover", function() {
        tetkenv.hoverElement(partNum);
      }, true);
      element.addEventListener("mouseout", function() {
        tetkenv.hoverElementOut(partNum);
      }, true);
    });
  } catch (e) {
    console.log(e);
  }
});
```

4.3.5 Ladebildschirm

Aufgrund der durch die Größe der Grafiken bedingte relativ lange Ladezeit des TetkEnv wurde von ACME ein Ladebildschirm gewünscht, der so lange aktiv bleibt, bis alle Komponenten des Viewers komplett geladen wurden. Dies soll die unschönen leeren Boxen und Felder verstecken, die vor dem korrekten Laden der SVG sichtbar sind.

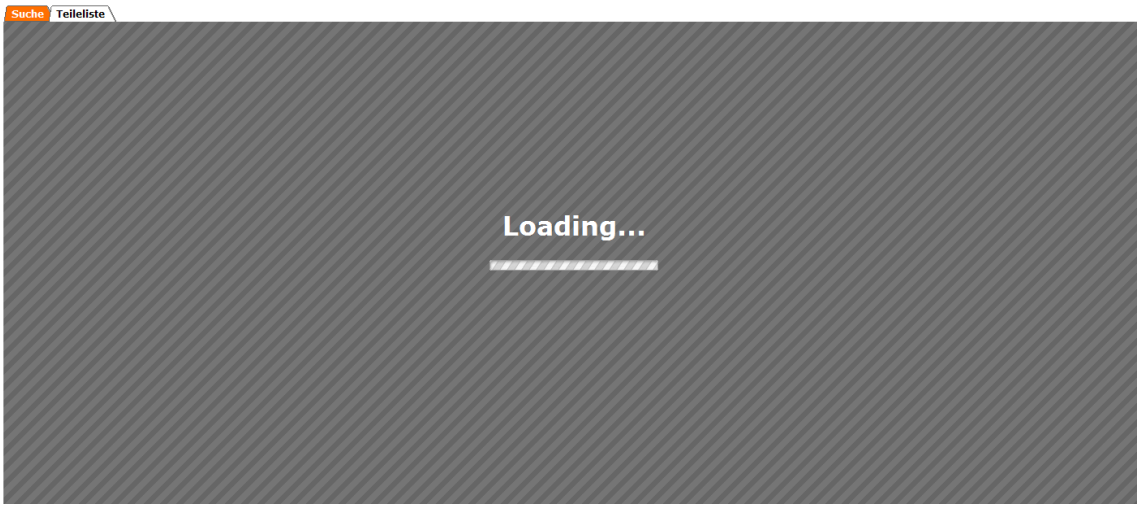


Abbildung 5: Ladebildschirm während des Renderns

Der Ladebildschirm wurde als einzelnes HTML-`<div>`-Element erstellt, der über allen Elementen des Viewers liegt und sie durch ein nicht-transparentes Hintergrundbild verdeckt (s. Abb. 5).

Listing 8: Code des Ladebildschirms

```
<div id="loadScreen"
    style='background: url(i/disabled.png);
      z-index: 100; height: 1000px; position: absolute;
      width: 100%; text-align: center;'>
    <div style="padding-top: 15%;">
        <h1>Loading...</h1>
        <img src="i/loader.gif" />
    </div>
</div>
```

4.3.6 Scrolling in der Stückliste beim Hervorheben

Die Stücklisten der einzelnen Maschinen können bis zu 50 einzelne Bauteile enthalten. Wenn eines dieser Bauteile in der Grafik hervorgehoben wurde, musste man den entsprechenden Eintrag in der Liste der Bauteile mühsam suchen. Dieses Verhalten wurde durch Einführung eines automatischen Scrollens in der Stückliste zum hervorgehobenen Bauteil hin im Hinblick auf die Usability des Viewers deutlich verbessert.

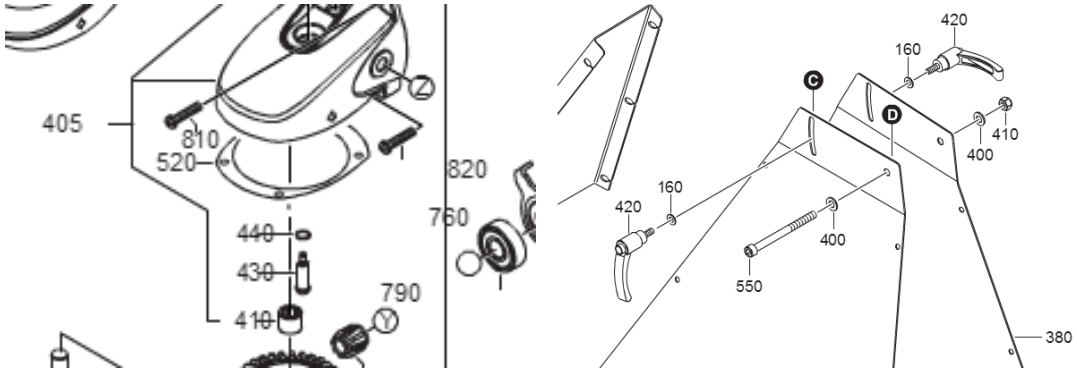
Listing 9: Methode zum Scrollen in der Bauteilliste

```
this.scrollTo = function(num) {
  $("html,body", window.parent.infoTop.document).animate({
    scrollTop: list.object.find("tr#" + num).offset().top - 15
  }, 500);
};
```

4.3.7 Schriftgrößen und -arten in der Grafik

Die gelieferten Grafiken von ACME waren mit einer Standardschriftart versehen, die von dem Viewer zu Grunde liegenden Flash-Renderer unzureichend interpretiert wurden. Die Schriften waren in Größe und Lesbarkeit stark eingeschränkt und konnten in dieser Form nicht als dem Endbenutzer zumutbar eingestuft werden. Ebenso wurden die Positionen der Bauteilbezeichnungen nicht korrekt dargestellt (s. Abb. 6).

Daher wurden alle Schriften nach Rücksprache mit ACME auf die Schriftart „Arial“ umgestellt. Diese Schriftart hat eine deutlich breitere Unterstützung und wird wie der Vergleich zeigt deutlich besser dargestellt.



(a) Alte Schriftart (b) Neue Schriftart

Abbildung 6: Vergleich der Schriftarten

5 Diskussion

5.1 Tests

5.1.1 Vorbereitung der Testumgebung

Um die umfangreichen Anforderungen an Browserumgebungen testen zu können, wurde für die funktionalen Tests des Viewers auf mehrere bestehende externe Virtual Machines zurückgegriffen, auf welchen diverse Versionen der gängigen Browser installiert sind.

Zugleich wurden auf der lokalen Entwicklungsmaschine mehrere Versionen der gleichen Browser installiert, die auf den Virtual Machines nicht verfügbar waren. Ingesamt wurden damit folgende Versionen auf den jeweiligen Betriebssystemen getestet:

- Microsoft Windows XP Professional x86 Service Pack 3
  - Mozilla Firefox Version 3.0, 3.5, 3.6, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0
  - Google Chome Version 14.0, 15.0, 16.0, 17.0
  - Microsoft Internet Explorer Version 6.0, 7.0, 8.0
  - Apple Safari Version 4.0, 5.0
  - Opera Version 9.52, 10.54, 11.61
- Microsoft Windows 7 Enterprise x64
  - Microsoft Internet Explorer Version 9.0

5.1.2 Testspezifikationen

Die Spezifikation der Tests wurde anhand der funktionalen Anforderungen (2.3.1) definiert. Sie sollen das System in einem Umfang testen, der der schlussendlichen Benutzung entspricht, der das System im tatsächlichen Betrieb ausgesetzt sein wird. Dabei wurden Standard-Vorgänge definiert, die der SVG-Viewer mit einem erwarteten Ergebnis quittieren muss.

Anzeigen einer Maschine Der Benutzer will eine Maschine ansehen und die entsprechende Explosionsgrafik anzeigen.

- **Ablauf**
  - Benutzer ruft den ETK über seinen Browser auf
  - Benutzer navigiert in der linken Navigationsleiste zur gewünschten Maschine
  - Benutzer klickt die Bezeichnung der Maschine an
- **Erwartetes Ergebnis**
  - Der Ladebildschirm wird angezeigt
  - Die Grafik der Maschine wird nach einiger Wartezeit angezeigt
  - Im unteren Frame wird die Stückliste der Ersatzteile angezeigt
  - Die Grafik und die Miniaturgrafik sind komplett sichtbar
  - Der Bildbereich ist nicht gezoomt und nicht verschoben

Interaktion: Zoomen Der Benutzer will in die Grafik hinein- bzw. herauszoomen und den Bildbereich damit vergrößern bzw. verkleinern.

- **Vorbedingung**
  - Benutzer hat den Vorgang *Anzeigen einer Maschine* erfolgreich abgeschlossen
- **Ablauf**
  - Benutzer klickt auf das Icon zum Hineinzoomen in der Funktionsleiste des SVG-Viewers
  - Benutzer klickt auf das Icon zum Herauszoomen in der Funktionsleiste des SVG-Viewers
- **Erwartetes Ergebnis**
  - Der Bildbereich der Grafik wird entsprechend der Zeit, die der Benutzer das entpsrechende Icon klickt, verkleinert.
  - Der Miniaturbildbereich wird proportional entsprechend des Bildbereichs der Grafik verkleinert.

  - Der Bildbereich der Grafik wird entsprechend der Zeit, die der Benutzer das entsprechende Icon klickt, vergrößert und bleibt stehen, sobald der Bildbereich auf seinem ursprünglichen Zustand ankommt.
  - Der Miniaturbildbereich wird proportional entsprechend des Bildbereichs der Grafik vergrößert und bleibt stehen, sobald er auf seinem ursprünglichen Zustand ankommt.

Interaktion: Verschieben Der Benutzer will den gezoomten Bildbereich verschieben.

- **Vorbedingung**
  - Benutzer hat den Vorgang *Anzeigen einer Maschine* erfolgreich abgeschlossen
- **Ablauf**
  - Benutzer klickt und zieht den Miniaturbildbereich in verschiedene Richtungen
- **Erwartetes Ergebnis**
  - Der Bildbereich der Grafik wird entsprechend der Position des Miniaturbildbereichs verschoben.
  - An den Grenzen der Grafik lässt sich der Miniaturbildbereich nicht weiter verschieben.

Hervorheben von Bauteilen Der Benutzer will entweder aus der Liste oder aus der Grafik heraus ein Bauteil hervorheben.

- **Vorbedingung**
  - Benutzer hat den Vorgang *Anzeigen einer Maschine* erfolgreich abgeschlossen
- **Ablauf**
  - Benutzer klickt auf die Bestellnummer eines Bauteils in der Grafik
  - Benutzer klickt auf das „Anzeigen“-Icon eines anderen Bauteils in der Liste
- **Erwartetes Ergebnis**
  - Die Bestellnummer wird in der Grafik hervorgehoben. Die gesamte Zeile des entsprechenden Bauteils in der Liste wird hervorgehoben. Die Liste scrollt selbstständig an die Position des hervorgehobenen Bauteils.
  - Das vorher hervorgehobene Bauteil wird wieder normal angezeigt und das neue Bauteil in der Grafik hervorgehoben. Die Liste scrollt selbstständig an die Position des neuen hervorgehobenen Bauteils. Die Zeile des neu hervorgehobenen Bauteils wird hervorgehoben und die alte Zeile wieder in ihren Urzustand versetzt.

5.1.3 Testergebnisse

Die Ergebnisse sind in Tabelle 1 dargestellt.

Umgebung	Anzeigen einer Maschine	Interaktion: Zoomen	Interaktion: Verschieben	Hervorheben von Bauteilen
Mozilla Firefox				
Version 3.0	OK	OK	OK	OK
Version 3.5	OK	OK	OK	OK
Version 3.6	OK	OK	OK	OK
Version 4.0	OK	OK	OK	OK
Version 5.0	OK	OK	OK	OK
Version 6.0	OK	OK	OK	OK
Version 7.0	OK	OK	OK	OK
Version 8.0	OK	OK	OK	OK
Version 9.0	OK	OK	OK	OK
Version 10.0	OK	OK	OK	OK
Microsoft Internet Explorer				
Version 6.0	OK	OK	OK	OK
Version 7.0	OK	OK	OK	OK
Version 8.0	OK	OK	OK	OK
Version 9.0	OK	OK	OK	OK
Opera				
Version 9.52	-	-	-	-
Version 10.54	OK	OK	OK	OK
Version 11.61	OK	OK	OK	OK
Apple Safari				
Version 4.0	OK	OK	OK	OK
Version 5.0	OK	OK	OK	OK
Google Chrome				
Version 14.0	OK	OK	OK	OK
Version 15.0	OK	OK	OK	OK
Version 16.0	OK	OK	OK	OK
Version 17.0	OK	OK	OK	OK

Tabelle 1: Testergebnisse

5.2 Erfüllung der Anforderungen

5.2.1 Funktionale Anforderungen

Aus den in Tabelle 1 dargestellten Testergebnissen lässt sich die Erfüllung der Anforderungen ableiten. Das ausgelieferte System erfüllte demnach die vorher festgelegten funktionalen Anforderungen folgendermaßen:

Anzeige

Der ETK soll browserunabhängig funktionsfähig sein.

Diese Anforderung wird als erfüllt betrachtet, da alle aus Sicht von ACME wichtigen Browser ausreichend mit allen Funktionalitäten unterstützt werden.

Interaktion

Es soll möglich sein, die Grafiken zu vergrößern und zu verkleinern, sowie den Bildausschnitt zu verschieben.

Diese Anforderung wird als vollständig erfüllt betrachtet.

Hervorhebungen in der Grafik

Es soll möglich sein, Teilenummern in den Grafiken visuell hervorzuheben.

Diese Anforderung wird als vollständig erfüllt betrachtet.

Hervorhebungen in der Teileliste

Beim Hervorheben von Teilenummern in der Grafik soll gleichzeitig auch der entsprechende Listeneintrag hervorgehoben werden.

Diese Anforderung wird als vollständig erfüllt betrachtet.

Integration

Der neue SVG-Viewer soll sich nahtlos in das bereits bestehende System integrieren und die anderen Funktionalitäten nicht beeinträchtigen.

Diese Anforderung wird als vollständig erfüllt betrachtet.

Die Abnahme zeigte, dass alle funktionalen Anforderungen zumindest mit geringfügigen Einschränkungen als erfüllt betrachtet werden können und ist damit als insgesamt erfolgreich abgeschlossen.

5.2.2 Technische Anforderungen

Die genauere Erfüllung der Anforderungen wird in der nachfolgenden Liste gezeigt:

Anzeige

Folgende Browser sollen mindestens in vollem Funktionsumfang unterstützt werden:

- *Firefox ab Version 3.0* - Vollständig erfüllt
- *Internet Explorer ab Version 7* - Vollständig erfüllt
- *Safari ab Version 4* - Vollständig erfüllt
- *Opera ab Version 9* - Eingeschränkt erfüllt
- *Chrome ab Version 10* - Vollständig erfüllt

Es kann vorausgesetzt werden, dass der Client JavaScript aktiviert hat und den Adobe® Flash® Player (ab Version 10) installiert und aktiviert hat.

Alle Anforderungen an die Browserversionen konnten mindestens erfüllt und teilweise sogar übertroffen werden (so ist z. B. auch Internet Explorer Version 6 unterstützt). Die einzige Einschränkung hierbei ist Version 9.52 des Opera-Browsers. Dies wird jedoch als trivial angesehen und schränkt die Erfüllung dieser Anforderung nicht ein.

Interaktion

Die möglichen Interaktionen, welche der SVG-Viewer unterstützen muss sind Zoomen und das Verschieben des Viewports. Hierzu soll eine Miniaturgrafik als Referenzpunkt der aktuellen Position des Viewports und der aktuellen Zoomstufe dienen.

Ebenso soll ein Zurücksetzen-Button implementiert werden, der Zoomstufe und Viewport wieder auf ihre Ausgangswerte zurücksetzt.

Die Anforderungen können als vollständig erfüllt angesehen werden.

Hervorhebungen

Hervorhebungen sollen nur auf den textuellen Repräsentationen der Teilenummern geschehen. Es soll stets nur eine Hervorhebung gleichzeitig aktiv sein. Sowohl der Klick auf eine Teilenummer in der Grafik als auch ein Klick auf den entsprechenden Listeneintrag soll zu einer Hervorhebung in beiden Komponenten führen.

Beim Klick auf den Zurücksetzen-Button sollen auch alle Hervorhebungen in Grafik und Liste zurückgesetzt werden.

Die Anforderungen können als vollständig erfüllt angesehen werden.

Integration

Um die volle Funktionalität der alten Funktionen des ETKs zu gewährleisten, sollen nur Dateien auf der Frontend-Seite des ETKs verändert, erstellt oder gelöscht werden.

Diese Anforderung konnte nicht erfüllt werden. Zur Umsetzung des Hervorhebens eines Bauteils sowohl in der Grafik wie auch in der Liste, musste der Rückgabewert des entsprechenden Icons in der Java-Quelldatei verändert werden. Diese Änderung kann jedoch als geringfügig betrachtet werden, da hierdurch die grundsätzliche Funktionsweise des ETKs nicht verändert wurde.

5.2.3 Sonstige Anforderungen

Im Zuge der Umsetzung soll die Server-Software, welchen den ETK darstellt, auf Tomcat Version 7 umgestellt werden. Diese Anforderung hat nur indirekt mit den Anforderungen an den SVG-Viewer zu tun, muss jedoch in der Umsetzung beachtet werden, da hier unerwünschte Nebeneffekte auftreten könnten.

Diese Anforderung kann als vollständig erfüllt angesehen werden. Die neue Tomcat Version konnte problemlos eingefügt werden und ermöglicht nun die Weiterentwicklung des ETKs auf einem höheren Technologiestand.

5.3 Probleme und Lösungsvorschläge

5.3.1 Flash außerhalb des Viewports

Bei den ersten Tests wurde eine auffällig lange Ladezeit der SVG-Grafiken festgestellt, die nicht erklärt werden konnte. In der prototypischen Entwicklung konnten die größten Grafiken (ugf. 10 MB Dateigröße) innerhalb von maximal 15 Sekunden geladen werden. Diese Ladezeit wurde im Produktivsystem deutlich überschritten.

Es konnte durch systematisches Testen festgestellt werden, dass dieses Verhalten immer dann auftrat, wenn die Grafik außerhalb des *Viewports* lag, während sie geladen wurde. Dies ist durch die Einführung des Ladebildschirms[4.3.5] geschehen, der in der ersten Version die Elemente des Viewers nicht überlagert hat, sondern sie außerhalb des Sichtbereichs der Seite gedrängt hat.

Als Viewport (Sichtbereich) wird der Bereich des Browser bezeichnet, der für den Benutzer sichtbar ist[Koc]. Meistens ist von einer Webseite nur ein bestimmer Teil sichtbar, die restlichen Elemente liegen außerhalb des „Sichtfensters“ des Browsers. Durch Scrollen verschiebt man demnach nur den Sichtbereich des Browsers, was die anderen Elemente sichtbar macht (s. Abb. 7).

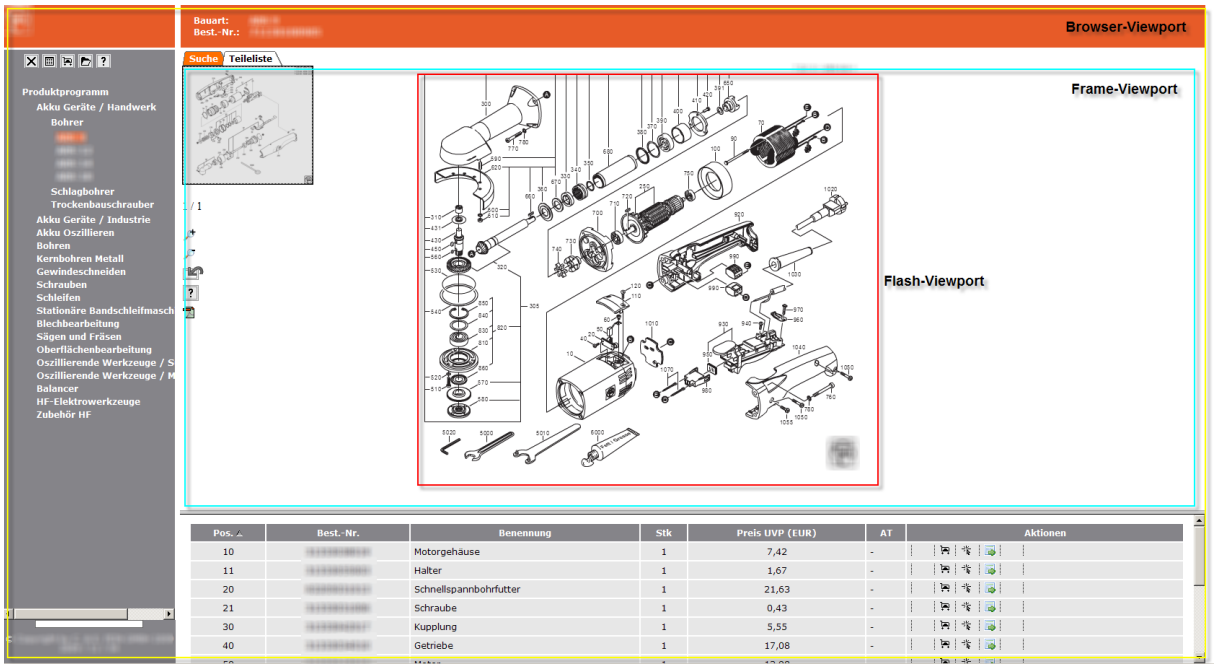


Abbildung 7: Viewport des SVG-Viewers und der Flash-Komponente

Im Fall des ETKs von ACME und die darin enthaltene Framestruktur[4.1] wird dies durch die Tatsache verkompliziert, dass jeder Frame für sich genommen einen eigenen Viewport hat. Dieser Viewport wird für jeden Frame einzeln verwaltet, sodass man in jedem Frame zu jedem Zeitpunkt verschiedene Viewport-Ausmaße und -Positionen, abhängig von der Größe des Frames im „Browserviewport“, zu beachten hat.

Der Viewport einer Seite (bzw. in diesem Fall eines Frames) ist für die Ausführung von Flash äußerst wichtig. Befinden sich Flash-Element außerhalb dieses Bereichs, wird automatisch die Ausführungsgeschwindigkeit sowie die Ressourcenzuordnung zum Flash-Plugin auf ein Minimum reduziert, um die Geschwindigkeit und Antwortzeiten des Browsers nicht durch ohnehin nicht sichtbare Elemente zu verlangsamen.

Dies hatte jedoch auch zur Folge, dass die Ressourcen des für den Viewer verwendeten Flash-Renderer reduziert wurden, was die Dauer des Renders deutlich verlängerte.

Gelöst werden konnte das Problem durch die Anpassung des Ladebildschirms, der nun durch geeignete *CSS*-Angaben über den Elementen des Viewers liegt, anstatt sie aus dem Sichtbereich zu drängen. Dadurch erkennt das Flash-Plugin nicht, dass die Flash-Komponente eigentlich nicht sichtbar ist und reduziert so die Ressourcen nicht, wodurch das Rendern wieder auf die normale Geschwindigkeit reduziert werden konnte.

5.3.2 Ladezeiten

Die Komplexität der verwendeten Grafiken hatte deutliche Auswirkungen auf die Ladezeit des Flash-Renderers. Trotz der in 5.3.1 beschriebenen Optimierungen konnte je nach Größe (die ein gutes Zeichen für die Komplexität der enthaltenen Grafik ist) eine Ladezeit von 1-2 Sekunden (1-2 MB Dateigrößte) bis zu 30 Sekunden (ugf. 10 MB Dateigröße) beobachtet werden.

Durch den Vergleich mit der Verwendung der in den Browser integrierten SVG-Renderern (z. B. Chrome oder Firefox 4+) wurde klar, dass die Dauer unmittelbar dem Flash-Renderer selbst zuzuschreiben ist. Für die Usability des ETKs wurden Ladezeiten ab 15 Sekunden als unzumutbar eingestuft.

Dieses Problem konnte in der Zeit des Projektablaufs nicht behoben werden. Die Möglichkeiten, die Ladezeiten drastisch zu verringern, sind;

- Verringerung der Komplexität der Grafiken selbst. Dies könnte z. B. durch bessere Verwendung von Gruppen und bereits vordefinierten Formen erreicht werden.
- Verwendung der nativen SVG-Darstellung. Dies ist die optimale Lösung, verhindert jedoch die einheitliche Verwendung und Darstellung der Grafiken auf den für das Projekt benötigten Browsern[2.3.2].

Angesichts des Projektumfangs und der Planung musste dieses Problem als nicht lösbar eingestuft werden.

5.3.3 Netzwerkumgebungen hinter Proxy-Servern

Um der Flash-Komponente die Grafiken zur Verfügung zu stellen, können diese nicht wie gewohnt einfach in den Quellcode der HTML-Seite eingebettet werden, sondern müssen über einen AJAX[4]-Aufruf vom Server angefragt werden. Diese Anfrage erlaubt es, aus Javascript heraus eine HTTP-Anfrage an den Server zu stellen und dessen Antwort zu verarbeiten.

Dieses Verfahren verursachte durch die in den meisten Firmen vorhandene Proxy-Server-Infrastruktur der Internetverbindung ein Problem, das dazu führte, dass die Grafiken nicht korrekt geladen werden konnten. Hierbei trat zu Tage, dass die Proxy-Konfiguration von ACME fehlerhaft war und daher der SVG-Viewer innerhalb des Firmennetzes nicht korrekt funktionierte.

Das Problem konnte innerhalb der Projektzeit nicht gelöst werden, ist aber aufgrund der Natur des Problems nicht der Entwicklung des SVG-Viewers zuzuschreiben, sondern einer fehlerhaften Konfiguration auf Seiten des Kunden.

5.3.4 Herauszoomen des Bildbereichs

Der SVG-Viewer verfügte bereits in der prototypischen Entwicklung über die Funktion, den Bildbereich durch Zoomen zu vergrößern und zu verkleinern. Hierbei zeigte sich bereits das Problem, dass der Ausschnitt des Miniaturbildes, welches den aktuellen Bildbereich anzeigt, beim vergrößern des Bildbereiches („Herauszoomen“) in bestimmten Situationen aus der ihn enthaltenden Box läuft (s. Abb. 8).

Dieses Verhalten ist nicht wünschenswert und wurde durch geeignete Abfragen in der Methode des Herauszoomens verhindert. Die Methoden prüfen *vor* dem Zoomen das Ergebnis der Berechnungen. Wird dabei festgestellt, dass der Miniaturbildbereich außerhalb der Grenzen des Miniaturbildes liegt oder sie in einer anderen Weise sprengt, wird der Zoomvorgang abgebrochen.

Listing 10: Neue Herauszoomen-Methode

```
1  // Verhindert, dass der Mover rechts aus dem Bild herausläuft
2  if (nextWidth + left >= image.thumbnail.width) {
3    nextLeft -= nextWidth - width;
4    nextLeft = nextLeft < 0 ? 0 : nextLeft;
5  }
6
7  // Verhindert, dass der Mover unten aus dem Bild herausläuft
8  if (nextHeight + top >= image.thumbnail.height) {
9    nextTop -= nextHeight - height;
10   nextTop = nextTop < 0 ? 0 : nextTop;
11 }
```

[4] Asynchronous Javascript And XML

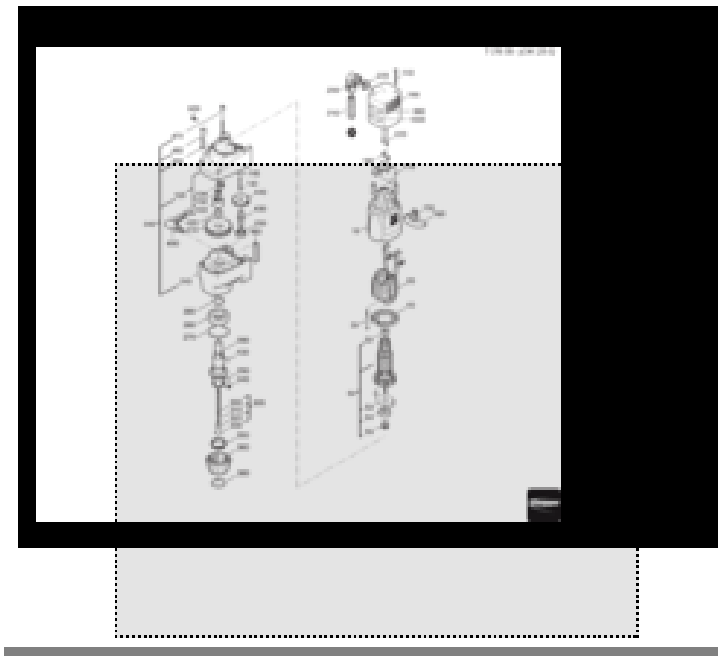


Abbildung 8: Grenzen des Miniaturbildes werden nicht eingehalten

5.3.5 Hervorheben nach Suche von Bauteilen

Die Suche des ETK-Systems ermöglicht das direkte Springen in eine Maschine, die das gesuchte Bauteil enthält. Dieses Springen hebt das entsprechende Bauteil in der Liste hervor, jedoch aufgrund des neuen SVG-Viewers nicht auch automatisch in der Grafik. Um das Verhalten anzupassen, wurde beim initialen Laden des Viewers eine Prüfung eingebaut, welche auf bereits hervorgehobene Bauteile in der Liste prüft.

Werden solche gefunden, werden die bereits bestehenden Methoden verwendet, um das Bauteil in der Grafik hervorzuheben und die Liste auf den entsprechenden Eintrag des Bauteils zu zentrieren.

Listing 11: Methode zum Hervorheben von bereits hervorgehobenen Bauteilen

```
1 var highlight = parseInt($(".spclistrowmarked .spcposnr", window.parent.infoTop.
    document).text());
2 if (highlight > 0) {
3     tetkenv.highlight(highlight);
4     tetkenv.scrollTo(highlight);
5 }
```

6 Ausblick

6.1 Verbesserung der Hervorhebung

Um die Qualität der Hervorhebungen von Bauteilen stark zu verbessern, sollten alle Pfade der Grafik, die zu einem Bauteil gehören, in eine generische Gruppe im Sinne des SVG-Standards zusammengefasst werden. Dies würde das Hervorheben gesamter Bauteile ermöglichen, anstatt nur die teilweise schwer sichtbaren Bauteilenummern.

Ein Beispiel, wie eine solche Hervorhebung aussehen kann, wurde bereits in der prototypischen Entwicklung aufgezeigt[Pes11, Abb. 9]. Das Hervoheben gesamter Bauteile ist aus Usability-Sicht deutlich angenehmer und leichter verständlich für Benutzer.

Außerdem könnte so eine automatisierte Zoom-Funktion erstellt werden, die beim Hervorheben nicht nur das Bauteil markiert, sondern den Bildbereich automatisch auf das Bauteil in seiner Gesamtheit zentriert. Das wäre möglich, da die Gruppendefinition des Bauteils exakt die äußeren Grenzen des Bauteils darstellt und sich damit berechnen lässt, wo im Bild sich die Gruppe des Bauteils befindet. Diese Position könnte dann ausgelesen und als neuer Bildbereich definiert werden.

Dazu müssten die Gruppen auch eine eindeutige Identifizierung erhalten, anhand derer sie sich an die Bauteile der Liste koppeln lassen, ähnlich wie das bereits bestehende Mapping[4.2.3].

6.2 Besseres Ausnutzen der Möglichkeiten von SVG

Viele der Funktionen, die der SVG-Viewer unterstützt, lassen sich nativ in SVG einbetten. Dies würde den SVG-Viewer wesentlich vereinfachen, da er nur die in SVG definierten Methoden aufrufen müsste. Beispiele könnten hier das Hervorheben von Bauteilen sein. Durch die Definition einer geeigneten Schnittstelle zwischen der SVG-Grafik und der JavaScript-Bibliothek des SVG-Viewers könnte so jede SVG-Grafik selbst definieren, wie sie die in ihr enthaltenen Bauteile hervorhebt, wenn die Methode von außen aufgerufen wird.

Dies könnte die Erweiterungsmöglichkeiten des SVG-Viewers deutlich verbessern. So ließe sich ein Framework entwickeln, welches mit den definierten Schnittstellen für jeden Kunden an wenigen Stellen angepasst werden muss, damit die von ihm gelieferten SVG-Grafiken in den Viewer eingebettet werden können.

Voraussetzung hierfür ist, dass die Flash-Komponente mittelfristig entfernt wird. Die Unterstützung der JavaScript-Flash-SVG-Bridge[5] ist gerade beim Verwenden von JavaScript-Frameworks wie *jQuery* ungenügend, sodass auf natives JavaScript zurückgegriffen werden muss. Dies ist insoweit ein Problem, als dass dadurch die Vorteile der Browserunabhängigkeit sowie der höheren Stabilität und Sicherheit verloren gehen.

Zudem eröglicht der Umstieg auf natives SVG eine deutlich schnellere Ladezeit beim Anzeigen der Grafiken.

[5] Dies bezeichnet die Methoden des Flash-Renderers, die JavaScript-Aufrufe in Flash „übersetzen“ und umgekehrt.

Literaturverzeichnis

[Con03] World Wide Web Consortium. Scalable Vector Graphics 1.1. Recommendation, W3C, http://www.w3.org/TR/2003/REC-SVG11-20030114/, Januar 2003. Zugriff: 13. März 2012.

[Cor07] Microsoft Corporation. PNG Files Do Not Show Transparency in Internet Explorer, Juli 2007. http://support.microsoft.com/kb/294714.

[DLR06] Pierre Delisle, Jan Luehe, and Mark Roth. JavaServer Pages 2.1 Specification. Technical report, Sun Microsystems, Mai 2006. http://download.oracle.com/otndocs/jcp/jsp-2.1-fr-eval-spec-oth-JSpec/.

[Ham11] David Hammond. Web Browser ECMAScript Support, 2011. http://www.webdevout.net/browser-support-ecmascript.

[Inc12] Adobe Systems Incorporated. Vorteile der Flash-Plattform, 2012. http://www.adobe.com/de/flashplatform/benefits/.

[Int11] ECMA International. ECMAScript Language Specification 5.1. Technical report, ECMA International, Juni 2011. http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf.

[Koc] Peter-Paul Koch. A tale of two viewports - part two. http://www.quirksmode.org/mobile/viewports2.html.

[Pes11] Peschka. *Darstellung von Vektorgrafiken in modernen Browserumgebungen*. PhD thesis, DHBW Ravensburg, 2011.

[RHJ99] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.01 Specification. Technical report, World Wide Web Consortium, Dezember 1999. http://www.w3.org/TR/1999/REC-html401-19991224/.

[uP06] VDI-Fachbereich Technischer Vertrieb und Produktmanagement. 4500 Blatt 3 - Technische Dokumentation - Erstellen und Verteilen von elektronischen Ersatzteilinformationen. Technical report, Verein Deutscher Ingenieure, Juni 2006. http://www.vdi.de/401.0.html?&no_cache=1&tx_vdirili_pi2%5BshowUID%5D=91410.

Abbildungsverzeichnis

Listings