Webbasierte Animation von Simulationsläufen auf Basis des Core Manufacturing Simulation Data (CMSD) Standards

Sören Bergmann[1], Florian Parzefall[1], Steffen Straßburger[1]
[1]Technische Universität Ilmenau
soeren.bergmann@tu-ilmenau.de

Animation von Simulationsläufen ist für viele Anwendungen ein nicht zu unterschätzendes Hilfsmittel. Die Nutzungsmöglichkeiten sind hierbei mannigfaltig, sie reichen von der Validierung der Modelle bis hin zur Ergebnispräsentation von Simulationsstudien. Dem Nutzen steht mitunter aber auch ein nicht zu unterschätzender Aufwand gegenüber, gerade im Kontext der automatischen webbasierten Simulation sind oft geeignete Animationen nicht verfügbar. Im Rahmen dieses Papers wird ein Ansatz vorgestellt, welcher ein bestehendes Framework zur automatischen Modellgenerierung, -initialisierung und Simulationsdurchführung inklusive Ergebnisauswertung auf Basis des Core Manufacturing Simulation Data (CMSD) Standards um die Möglichkeit der vollständig automatischen webbasierten Animation erweitert. Hierzu wird neben der Diskussion der Grundlagen der Animation das bestehende Framework und der dem Framework zugrunde liegende CMSD-Standard vorgestellt. Des Weiteren werden verschiedene Implementierungstechnologien vom Streamen von Videos über das Nutzen von Plug-Ins wie Flash oder Java Applets bis hin zu modernen Techniken wie HTML 5, CSS3 und JavaScript kritisch beleuchtet. Abschließend wird eine prototypische Implementierung mittels HTML 5 Canvas und den JavaScript Frameworks JQuery und KineticJS vorgestellt.

1 Einleitung

Die diskret-ereignisgesteuerte Simulation wird in verschiedensten Disziplinen und Anwendungsgebieten eingesetzt. Insbesondere im Bereich der Produktion und Logistik ist Simulation eine akzeptierte Methode zur Planung, Evaluierung und Verbesserung bzw. Steuerung von Prozessen [1]. Eine in den letzten Jahren immer wieder genannte Herausforderung im Kontext der Simulation ist in der Reduktion des Aufwands von Simulationsstudien bzw. in der verbesserten Nutzbarkeit der Simulation vor allem auch durch Nicht-Simulationsexperten zu sehen [2].

Ein hierzu in den letzten Jahren verfolgter Ansatz ist die Automatisierung der Modellgenerierung und -initialisierung. In Vorarbeiten konnte gezeigt werden, dass unter Nutzung des Core Manufacturing Simulation Data (CMSD) Standards u.a. eine Aufwandsreduktion bei der Simulationsmodellgenerierung [3-5], -initialisierung [3,6] und Ergebnisauswertung [3,7] möglich ist.

Ein weiterer wichtiger Trend der letzten Jahre ist die zunehmende Realisierung von webbasierten Anwendungen, hierbei sind sowohl im Browser laufende Anwendungen bzw. Nutzerschnittstellen als auch Cloud-basierte bzw. "Software as a Service" Lösungen zu subsumieren [8].

"Simulation as a Service"-Lösungen und webbasierte Nutzerschnittstellen sind hierbei eine mögliche Chance zur weiteren Reduktion von Aufwänden, sowie zur Verbesserung der Benutzerfreundlichkeit der Simulation, auch dies konnte in Vorarbeiten [3,7] gezeigt werden.

Zudem konnte gezeigt werden, dass eine weitere Reduktion durch Verteilung von Experimentläufen auf mehrere Simulatorinstanzen realisierbar ist [3,7].

Ein bisher in den meisten Ansätzen zur automatischen Modellgenerierung, -initialisierung und automatisierten Durchführung von Experimenten vernachlässigter Punkt ist die Animation von Simulationsläufen. In einigen Fällen, insbesondere bei Nutzung von bausteinbasierten Simulationswerkzeugen, ist zwar meist eine rudimentäre Animation per se gegeben, aber insbesondere in webbasierten Simulationsszenarien ist diese für den Nutzer nicht ohne weiteres nutzbar.

Zu bemerken ist aber, dass die Animation von Simulationsläufen von vielen Simulationsnutzern als wichtiges Hilfsmittel in allen Phasen der Simulation, von der Validierung bis hin zu Ergebnispräsentation bzw. Wissensvermittlung angesehen wird [9].

Im Rahmen dieses Papers wird ein Ansatz vorgestellt, der ein Framework zur webbasierten Simulation auf Basis des CMSD-Standards um die Möglichkeit der 2D-Animation ergänzt. Hierbei wird wie in allen

Komponenten des Frameworks auf eine rein webbasierte, d.h. für den Nutzer allein im Browser laufende intuitiv nutzbare Benutzeroberfläche gesetzt. Eine Forderung ist hierbei, dass keine zusätzlichen Aufwände zur Erzeugung von Layout- und Tracedaten anfallen. Insbesondere Aufwände einer zusätzlichen manuellen Layouterzeugung sind zu vermeiden.

Alle für die Animation benötigten Informationen liegen allein im CMSD Datenformat vor. Die CMSD Daten sind hierbei mit denen identisch, welche bereits erfolgreich zur Modellgenerierung, Initialisierung sowie der Ergebnisdatenanalyse mittels eines webbasierten Statistikmonitors genutzt werden [3-7]. Weitere optionale Ergänzungen zur Verbesserung der Animation (z.B. Definition spezifischer Icons usw.) sind ebenso im CMSD Format zu hinterlegen.

Im folgenden Abschnitt werden kurz die benötigten Grundlagen bzgl. der Animation von Simulationsläufen gelegt, bevor der CMSD-Standard und dessen grundlegende Interpretation thematisiert werden. Den Grundlagenteil schließen Ausführungen bzgl. der Auswahl der verwendbaren Technologien zur Realisierung der Animation als webbasierte Anwendung ab. Im Abschnitt 3 wird das Konzept zu webbasierten Animation auf Basis des CMSD-Standards sowie dessen prototypische Umsetzung vorgestellt.

Abschließend werden ein kurzes Fazit sowie ein Ausblick auf weitere Forschungs- bzw. Entwicklungsmöglichkeiten gegeben.

2 Grundlagen

Im folgenden Abschnitt werden zunächst Grundlagen der Animation diskutiert, der CMSD Standard, nötige Interpretationen und Erweiterungen vorgestellt sowie die Auswahl der für die im Prototypen genutzten (Web-)Technologien begründet.

2.1 Animation von Simulationsläufen

Die VDI Richtlinie 3633 Simulation von Logistik-, Materialfluss- und Produktionssystemen Blatt 11 definiert Animation als "die Erzeugung und Präsentation von Bildfolgen, in denen Änderungen (z.B. Zustandsänderungen und Bewegungen von Modellelementen) einen visuellen Effekt bedingen.

Unter visuellen Effekten kann eine über die Zeit variierende Position, die Änderung von Form, Farbe, Transparenz, Struktur und Musterung eines Objektes, die Änderung der Beleuchtung sowie der Position, Orientierung und Brennweite der Kamera verstanden werden" [9].

Bei der Gestaltung der Animation ist besonders auf Expressivität, Effektivität und Angemessenheit zu achten [10]. Unter Expressivität wird hierbei die unverfälschte Wiedergabe der Informationen und unter Effektivität die geeignet intuitive Darstellung verstanden [9]. Angemessenheit betrachtet zusätzlich den Aufwand zur Realisierung der Animation in Relation zum Nutzen [10].

Im Kontext kommerzieller Simulationswerkzeuge sind verschiedene Animationsansätze anzutreffen. Zum einen kann zwischen vollintegrierten und entkoppelten Ansätzen unterschieden werden. Bei ersteren wird direkt bei jeder Simulationsstudie eine Kombination aus dem eigentlichen Simulations- und dem Animationsmodell erstellt, typisch ist dieses Vorgehen in bausteinorientieren Simulationswerkezeugen, z.B. Plant Simulation [11]. Die zweite Variante ist eine entkoppelte Animation, d.h. das Animationsmodell wird getrennt bzw. sogar nur bei Bedarf erstellt. Diese zusätzliche Modellierung erfolgt meist in einem eigenständigen Softwarewerkzeug, z.B. Proof Animation [12]. Im Fall der entkoppelten Animation muss diese durch die Simulation mit Tracedaten versorgt werden. Tracedaten können hierbei sowohl komplett nach Ablauf eines Simulationslaufes am "Stück" (Offline bzw. Post Simulation Animation) oder laufend (Online Animation) übertragen werden [5].

Im Gegensatz zum Grundprinzip der diskret-ereignisgesteuerten Simulation, in welcher von Ereigniszeitpunkt zu Ereigniszeitpunkt "gesprungen" wird, ist bei der Animation eine zeitproportionale Darstellung der Ereignisse im Modell einzuhalten, wobei neben des Ablaufs in "Echtzeit" auch Abläufe in Zeitraffer und Zeitlupe möglich sind [9, 13].

Bei der Animation muss zunächst bzgl. der Anzahl der Darstellungsdimensionen unterschieden werden, mögliche Ausprägungen sind 1D, 2D, 2,5D und 3D Visualisierungen. Für die Animation sind 1D Darstellungen allein nicht relevant, allein Zusatzinformationen lassen sich z.B. in Balkendiagrammen darstellen. Obwohl 3D-Visualisierungen im Kontext der Digitalen Fabrik an Bedeutung gewinnen [14], ist 2D die vorherrschende Darstellungsform im Kontext der Animation von Simulationsläufen in Produktion und Logistik. In Ausnahmen sind auch perspektivische Ansichten in 2,5D sinnvoll [13].

Ein weiteres Kriterium, welches im Kontext der Animation relevant ist, sind die Interaktionsmöglichkeiten, d.h. insbesondere der Navigation (Start, Stopp, Sprung zu einen Zeitpunkt usw.) und Anzeige (Zoom usw.) in der Animation. Weiterhin ist meist auch das Lesen von Zusatzinformationen zum aktuellen Animationszeitpunkt, z.B. aktueller Auslastungen von Ressourcen, Details zu Aufträgen usw. möglich. Interaktionsmöglichkeiten im weitesten Sinn können zudem Möglichkeiten der Manipulation der Visualisierung bzw. der grundlegenden Simulation über die grafische Darstellung umfassen [9, 13]. Diese erweiterten Möglichkeiten gehen allerdings über den Anspruch der klassischen Animation weit hinaus.

In den folgenden Ausführungen wird ein interaktiver entkoppelter, offline 2D Animationsansatz verfolgt.

2.2 Simulation auf Basis des CMSD-Standards

Das Core Manufacturing Simulation Data (CMSD) Information Model ist ein unter der Federführung der Simulation Interoperability Standards Organization (SISO) entwickelter Standard, der vor allem das Ziel verfolgt, Möglichkeiten des Datenaustauschs und die Interoperabilität zwischen Simulationen und anderen Anwendungen der Fertigung (z.B. ERP, MES) zu erleichtern [15, 16].

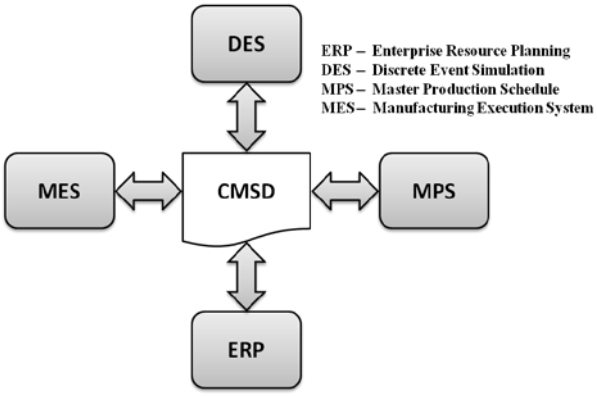

Abbildung 1. CMSD als neutrales Datenaustauschformat verschiedener Systeme der Fertigung [16]

Die CMSD Spezifikation umfasst zwei Teile. In Teil eins [15] werden UML Klassen- und Paketdiagramme zur Definition aller benötigten Entitäten und ihrer Eigenschaften (vgl. Abbildung 2) genutzt. Teil zwei [17] setzt diese Definitionen in technisch nutzbarer Art in Form von XML Schemabeschreibungen um. Hierbei kommen die sich ergänzenden Schemasprachen RelaxNG und Schematron zum Einsatz.

Versuche haben nachgewiesen, dass der Standard zwar im Detail interpretiert und ggf. mittels des im Standard verankerten Property-Konzepts erweitert werden muss, aber er geeignet ist, alle für eine Simulation nötigen Daten adäquat abzubilden und somit (webbasierte) automatische Modellgenerierung und -initialisierung aber auch die Abbildung von Ergebnissen von Simulationsläufen zu ermöglichen [4, 6-8].

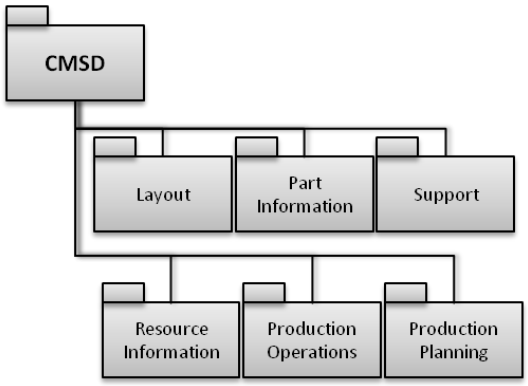

Abbildung 2. Pakete des CMSD-Standards [15]

Im Kontext der Animation von Simulationsläufen werden, wie bereits andiskutiert, zum einen Layoutinformationen und zum anderen Tracedaten benötigt.

Layoutinformationen lassen sich gut durch Entitäten des CMSD-Layout Paketes (z.B. LayoutObject), die meist eng mit Entitäten des Ressource Paketes (z.B. Ressource) verknüpft sind, abbilden. Ergänzend können organisatorische oder den Arbeitsablauf betreffende Informationen weiterer Pakete genutzt werden [3]. Grundlegende Layoutinformationen, z.B. zur Anzahl, Position und Fläche von Maschinenressourcen oder Hintergründe lassen sich direkt im CMSD Datenformat z.B. um die Angabe von Bildern oder komplexe geometrische Muster (Shapes) ergänzen.

Die Tracedaten werden mittels Events (Klasse des Support Paketes; vgl. Abbildung 3) abgebildet, welche Fertigungsaufträge (Jobs) des Production Operation Paketes erweitern können. Diese Mimik ist in Anlehnung an die Betriebsdatenerfassung (BDE) entwickelt worden und kam bereits zur Ergebnisdatenbeschreibung bzw. Kennzahlermittlung und statischen Darstellung von Simulationsläufen (z.B. Gantt Diagramme) im WebStatistikMonitor zum Einsatz und ermöglicht u.a. die zeitliche Rekonstruktion eines speziellen Simulationslaufs [3, 7].

Jedes Event umfasst hierbei zumindest eine laufende Nummer, welche die Erzeugungsreihenfolge abbildet,

einen Zeitstempel, welcher kennzeichnet zu welcher Simulationszeit das Event auftrat, einen Namen, der den Eventtyp (start setup, start work, broken usw.) kennzeichnet, sowie ggf. Referenzen auf beteiligte Ressourcen wie Werker und/oder Maschinen [3,7].

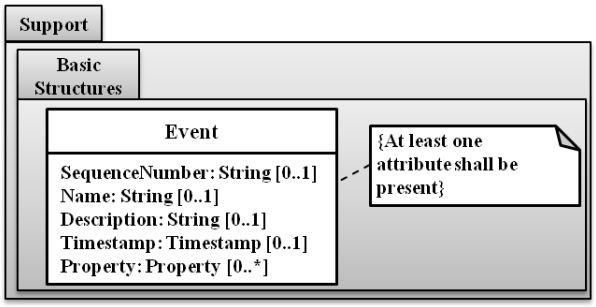

Abbildung 3. Die CMSD Event Klasse [15]

2.3 Auswahl der Webtechnologie

Im Rahmen der Umsetzung einer webbasierten Animationslösung stellt sich die Frage, welche Technologien geeignet sind.

Grundsätzlich lässt sich eine webbasierte Animation auf unterschiedliche Arten basierend auf verschiedenen Technologien implementieren. Im Folgenden soll eine Auswahl möglicher Ansätze bezüglich ihrer Stärken und Schwächen sowie möglicher Ausschlusskriterien untersucht werden. Eine umfängliche Diskussion aller Möglichkeiten würde den Rahmen dieses Papers jedoch sprengen.

Basierend auf der Annahme, dass serverseitig bereits eine Animation lauffähig vorliegt, entweder da der Simulator diese per se bei der Simulationsmodellerstellung mitgeneriert z.B. in Plant Simulation (vgl. [4]) oder diese in einem externen Werkzeug z.B. Proof Animation generierbar ist (vgl. [5]), ist der erste Lösungsansatz in der Bereitstellung von Videodaten zu sehen, welche als (Live-) Stream oder Download verfügbar gemacht werden. Neben der ggf. hohen Belastung des Servers sowie der Tool- und Plattformabhängigkeit ist vor allem die beschränkte Interaktionsmöglichkeit als Ausschlusskriterium zu sehen. So ist ggf. eine Navigation zwar noch entsprechend zu realisieren, eine Präsentation von interaktiven Zusatzinhalten hingehen ist kaum möglich.

Die zweite Gruppe von Lösungsmöglichkeiten basiert auf der Nutzung von Technologien, welche ggf. im Browser des Nutzers lauffähig sind, jedoch entsprechende Zusatzkomponenten (Plug-Ins) benötigt. Typische und verbreitete Vertreter dieser Gruppe sind Java-Applets [18], Flash [19] und Microsoft Silverlight [20], aber auch ActiveX und JavaFX, welche hier nicht explizit betrachtet werden sollen.

Allen Technologien dieser Gruppe ist der Bedarf an zusätzlich zum Browser installierten Softwarekomponenten gemein, welche zwar meist eine hohe aber nicht vollständige Verbreitung aufweisen. Nachteilig ist, dass entsprechende Plug-Ins ggf. nicht für alle Plattformen, Betriebssysteme und Browser verfügbar sind, unterschiedlich gute Sicherheitskonzepte aufweisen und nicht standardisiert sind, zudem werden einige der Technologien kaum oder gar nicht weiterentwickelt. Im Folgenden werden einzelne Technologien andiskutiert.

Java Applets ist die erste Technologie dieser Gruppe. Applets als Teil der Java SDK sind eine mittlerweile in die Jahre gekommene Technologie, die vor allem durch die Möglichkeit der Nutzung vieler existierender Java Klassen und Konzepte wie z.B. umfangreiche Kommunikationsmöglichkeiten mittels verschiedener Protokolle von Socket, RMI bis CORBA besticht. Des Weiteren fällt JAVA-Entwicklern die Entwicklung von Applets leicht, zudem sind die bestehenden Java Entwicklungsumgebungen (IDE) nutzbar. Des Weiteren verfügt diese Technologie über ein umfangreiches Sicherheitskonzept, z.B. Sandboxkonzept, Same-Origin-Policy etc. Dem gegenüber stehen beschränkte Möglichkeiten bzgl. der Gestaltung des Nutzerinterfaces (UI), hohe Aufwände bei der Entwicklung und Wartung, schlechte Trennung zwischen Logik und Design sowie eine ggf. schlechte Performance [21].

Das Java Applet-basierte Skopeo war eines der wenigen webbasierten Animationsframeworks für Simulationen. Skopeo kann hierbei Proof Animation Daten (Layout und Trace) interpretieren und in einem Java Applet im Browser darstellen [22].

Eine zweite Implementierungsmöglichkeit ist in der Nutzung von Adobe Flash zu sehen. Flash war und ist (mit stark sinkender Bedeutung) der de facto Standard zur Darstellung von Videos und Animationen (ggf. auch interaktiv, z.B. Spiele) im Web. Flash ist ein vektorbasiertes Format, welches durch eine eigene Skriptsprache (Actionscript) ergänzt wird. Problematisch und in Summe zum Ausschluss führend ist neben den allgemeinen, für alle Plug-In-basierte Techniken geltenden Einschränkungen, besonders das proprietäre Datenformat, die weitegehende Bindung an die kostenpflichtige Entwicklungsumgebung von Adobe sowie die gerade auf mobilen Endgeräten

teilweise Nichtverfügbarkeit bzw. die hohen Ressourcenanforderungen und -verbrauch zu nennen [19, 23].

Eine weitere ebenso wie Flash proprietäre Technik ist Microsoft Silverlight. Das Ausschlusskriterium ist besonders das mangelnde bzw. in den letzten Versionen ausgehebelte Sicherheitskonzept sowie die enge Bindung an Windows als die Zielplattform [20, 24].

Die dritte und letzte Gruppe von Möglichkeiten zur Realisierung einer webbasierten Animation kann mit der Überschrift „Nutzen von aktuellen standardisierten Webtechniken“ beschrieben werden. Bei den Standards handelt es sich vor allem um den vom W3C kurz vor der Verabschiedung stehenden HTML5 Standard [25, 26] in Verbindung mit JavaScript (standardisiert als ECMAScript 262 [27]) sowie CSS3 [28].

Diese Standards ermöglichen eine Plug-in freie Realisierung komplexer interaktiver Webanwendungen, welche in allen aktuellen Browsern inkl. Browsern mobiler Endgeräte lauffähig sind.

Im Detail sind verschiedene Spielarten möglich. So können alle benötigten Daten auf verschiedene Arten zur Verfügung gestellt werden, vom "klassischen" Laden mit der Webseite bis hin zum (nach-)Laden der Daten bei Bedarf bzw. Verfügbarkeit mittels Techniken wie AJAX oder Websockets.

Auch bei der Realisierung der eigentlichen Anzeige sind diverse Möglichkeiten mit unterschiedlichen Vor- und Nachteilen möglich. So sind in HTML5 und bei Nutzung von JavaScript Animationen mittels Canvas Elementen oder als SVG (Scalable Vector Graphics) möglich. Canvas bietet hierbei die Möglichkeit, dynamische Bitmaps zu erstellen, SVG hingegen ist ein vektorbasiertes Zeichenformat. Beide Technologien sind prinzipiell ähnlich gut geeignet, unterscheiden sich aber in einigen Details (vgl. Tabelle 1).

Neben den beiden beschrieben Möglichkeiten zur Realisierung ist auch eine sehr performante aber derzeit noch mit erheblichem Entwicklungsaufwand verbundene Realisierung auf Basis von CSS3 Transformation bzw. Animation denkbar [28].

Zusammenfassend kann gesagt werden, dass einige verschiedene Techniken theoretisch möglich sind und sich ein einzelner klarer Sieger nicht abzeichnet. Der Verzicht auf Plug-Ins, die zu erwartende Zukunftssicherheit, die Plattformunabhängigkeit (Server und Client) bei moderatem Entwicklungsaufwand, sowie die in den letzten Jahren enorm gestiegene Performance von Java Script Engines sprechen jedoch für HTML 5, JavaScript und CSS3 basierte Lösungen.

Canvas	**SVG**
Pixelbasiert (dyn. PNG)	Formbasiert
Ein HTML-Element	Mehrere grafische Elemente (Teil Dokumentobjektmodells - DOM)
Nur durch Skript änderbar	Durch Skript und CSS änderbar
Ereignismodell / Benutzerinteraktion ist präzise (x,y)	Ereignismodell / Benutzerinteraktion ist allgemeiner ("rect", "path")
Die Leistung ist bei kleineren Oberflächen, einer größeren Objektanzahl (>10T) oder bei beiden besser.	Die Leistung ist bei einer kleineren Objektanzahl (<10T), größeren Oberflächen oder bei beiden besser.

Tabelle 1. Canvas vs. SVG (in Anlehnung an [29])

CSS 3 wird aus Gründen des Aufwands und der Flexibilität für diesen Anwendungsfall als schlechteste der genannten Alternativen ausgeschlossen, Canvas und SVG sind gleichwertig. Im Folgenden wird aber allein eine auf Canvas Elementen basierte Lösung vorgestellt.

3 Prototypische Implementierung eines webbasierten Animationstools

Eine Implementierung wurde mittels HTML 5 Canvas Elementen und JavaScript prototypisch realisiert. Dieser Prototyp ergänzt ein auf dem CMSD-Standard aufbauendes Framework, welches bereits webbasierte Simulation ermöglicht. Zur Vereinfachung des Entwicklungsprozesses, zur Verbesserung der Browserkompatibilität, zum Senken des Testaufwandes usw. wurde auf entsprechende JavaScript-Frameworks zurückgegriffen. Konkret wurde JQuery [30] für die Realisierung der AJAX Anfragen, mittels welcher sämtliche Datenübertragungen realisiert wurden, sowie das speziell für Canvas-basierte Animation optimierte Framework KineticJS [31] verwendet.

KineticJS ermöglicht eine abstraktere Definition von Bildern und deren Animation. Zudem werden Schwächen des Canvas-Ansatzes geschickt gemildert. So ist bspw. durch das Nutzen mehrerer sich dynamisch

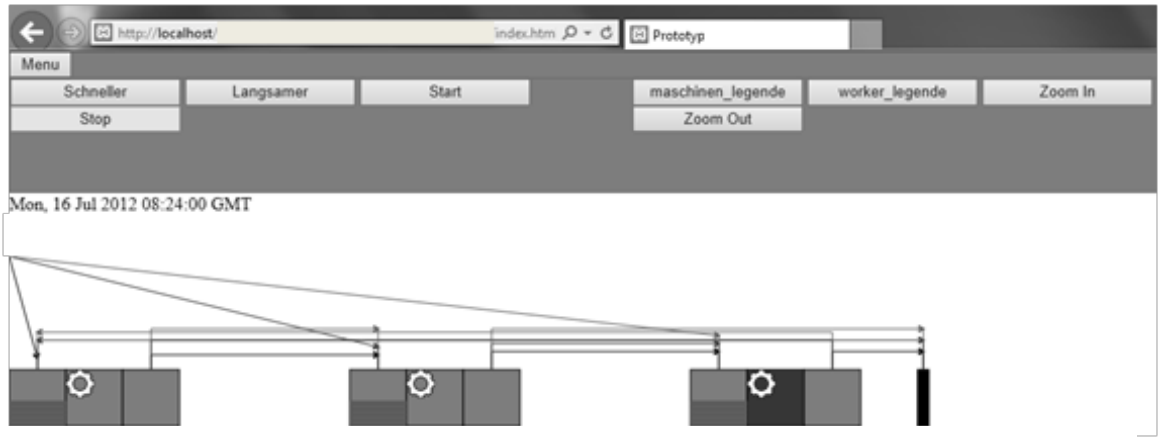

Abbildung 4. Screenshot des Prototyps

ergänzender Canvas-Elemente eine Manipulation über den DOM-Baum oder auch Zoomen ähnlich SVG möglich. Dies ermöglicht vor allem eine leichte Implementierung von zusätzlichen Interaktionen, z.B. der Anzeige von aktuellen Parametern, Kennziffern und Status von Aufträgen oder Maschinen bei Mausklick mittels eines entsprechenden Tooltips sowie das Zoomen zu jeder Zeit.

Grundlegend werden im Prototyp sämtliche Daten aus Dateien, welche serverseitig vorliegen, mittels AJAX Aufrufen in ein JavaScript Array geladen (vgl. Abbildung 5). Zusätzlich werden zunächst nicht direkt abgebildete Elemente heuristisch ermittelt und ergänzt, z.B. Quellen und Senken.

Alle statischen Elemente der Animation werden sofort nach erfolgreichem Laden der Layoutdatei gezeichnet. Das Erzeugen der Anzeige im Browser inklusive Bewegungen von Elementen wird über Funktionalitäten des KineticJS Frameworks realisiert. Hierzu werden für Maschinenressourcen deren Eigenschaften wie Position, Größe und die ggf. angegebene Grafik und Form ausgewertet und entsprechend in ein HMTL5 Canvas gezeichnet. Zudem werden Funktionen z.B. zur Erzeugung eines Popups mit Statusinformationen bei Klick auf das Element hinterlegt.

Nach Laden des Trace werden alle möglichen Arbeitspläne ermittelt. Aus diesen werden die möglichen Pfade der Aufträge automatisch generiert und dargestellt. Danach wird, unter Berücksichtigung ggf. hinterlegter Schichtkalender, die Animationsuhr initialisiert und die Darstellung aller Elemente ihren aktuellen Zuständen angepasst, z.B. Markierungen von gestörten Maschinen (vgl. Abbildung 4).

Ab diesen Zeitpunkt kann die Animation jederzeit ablaufen. Während der eigentlichen Animation werden basierend auf den Stand der "Animationsuhr" die Tracedaten kontinuierlich auf anstehende Events hin untersucht. Die Suche erfolgt in einem während der Initialisierung angelegten JavaScript Array, welches alle Events in zeitlich geordneter Reihenfolge enthält. Events führen hierbei je nach Typ zur direkten Änderung der Anzeige einzelner Elemente und / oder zur Erzeugung von Tweens (abgeschlossen laufender Animationsteile), welche u.a. für die Bewegung von Aufträgen zwischen Elementen genutzt werden.

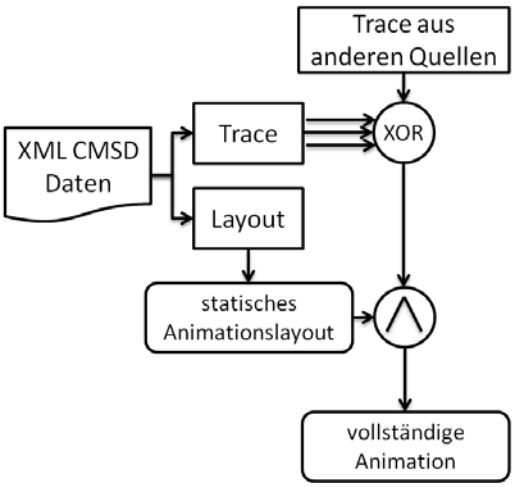

Abbildung 5. Datenfluss im Prototyp (in Anlehnung an [32])

Eine Veränderung der Tickrate der Animationsuhr (Buttons in der Oberfläche vorhanden) ermöglicht Zeitraffer und Zeitlupe Funktionalitäten, hierbei sind aber ggf. aktive Tweens bzgl. der Geschwindigkeit anzupassen. Das Ende der Animation wird durch das manuelle Stoppen der Animationsuhr oder die Verarbeitung aller Tracedaten bewirkt.

4 Fazit und Ausblick

Es konnte gezeigt werden, dass auf Basis vorhandener Daten im CMSD Format eine Animation von Simulationsläufen möglich ist. Des Weiteren konnte ebenfalls prototypisch die Machbarkeit von webbasierter Animation bei Verzicht auf zusätzliche Plug-Ins basierend auf modernesten Webtechnologien bewiesen werden. Alle Versuche zeigten, dass die Animation eine geeignete Expressivität und Effektivität ermöglicht. Eine Anpassung auf den konkreten Anwendungsfall ist mit angemessenem Aufwand zu realisieren. Ebenfalls zu bemerken ist, dass die Performance in den ersten Tests in allen Phasen gut war. So sind auf handelsüblicher Hardware und bei üblichen Netzwerkanbindungen nur geringe Ladezeiten und auch bei schnellem Vorlauf und Zoomen keinerlei negativen Effekte (z.B. Ruckeln) während der Animation aufgetreten. Umfangreiche Praxistests des Animationsansatzes stehen jedoch noch aus.

Über den bestehenden Prototypen hinaus sind weiterführende Forschungs- und Entwicklungspakete möglich. So sind mittels moderner Webtechnologien, z.B. WebGL [33] und entsprechenden JavaScript-Frameworks ebenso 3D-Animationen in sehr hoher Qualität und dank Hardwarebeschleunigung in guter Performance direkt im Browser möglich.

Ein weiterer Ansatzpunkt ist die Erweiterung des Prototyps um Möglichkeiten der Online Animation bzw. der onlinenahen Animation, d.h. der Animation noch laufender Simulationsexperimente. Hierzu ist zum einen eine geeignete (Trace-)Datenversorgung z.B. über Websockets zu realisieren. Zudem sind wie bei jedem Streamen von Livedaten oder auch in der verteilten Simulation Fragen wie das geeignete Zwischenpuffern der Daten und das Handling von verspäteten Events usw. zu betrachten.

5 References

[1] VDI - Verein Deutscher Ingenieure. *VDI Richtlinie 3633-1 Simulation von Logistik-, Materialfluss- und Produktionssystemen - Blatt 1 Grundlagen.* Beuth, Deutschland, 2008.

[2] S. Bergmann und S. Strassburger. *Challenges for the Automatic Generation of Simulation Models for Production Systems.* In Proceedings of the 2010 Summer Simulation Multiconference, in Ottawa (Kanada), S. 545-549, 2010.

[3] S. Bergmann. *Automatische Generierung adaptiver Modelle zur Simulation von Produktionssystemen.* Dissertation, TU Ilmenau, Deutschland, 2013.

[4] S. Bergmann und S. Straßburger. *Generierung und Integration von Simulationsmodellen unter Verwendung des Core Manufacturing Simulation Data (CMSD) Information Model.* In Proceedings der 14. ASIM-Fachtagung Simulation in Produktion und Logistik. In Karlsruhe, S. 461-468, 2010.

[5] S. Bergmann, S. Stelzer, S. Wüstemann und S. Strassburger. *Model generation in SLX using CMSD and XML Stylesheet Transformations.* In Proceedings of the 2012 Winter Simulation Conference, in Berlin, 2012.

[6] S. Bergmann, S. Stelzer und S. Strassburger. *Initialization of simulation models using CMSD.* In Proceedings of the 2011 Winter Simulation Conference, in Phoenix (USA), S. 2228-2239, 2011.

[7] S. Bergmann, S. Stelzer und S. Strassburger. *A new web based method for distribution of simulation experiments based on the CMSD standard.* In Proceedings of the 2012 Winter Simulation Conference, in Berlin, S. 3057-3068, 2012.

[8] Gartner Inc. *Gartner Executive Programs' Worldwide Survey of More Than 2,300 CIOs Shows Flat IT Budgets in 2012, but IT Organizations Must Deliver on Multiple Priorities.* Abruf am 8. Januar, 2014. www.gartner.com /it/page.jsp?id=1897514.

[9] VDI - Verein Deutscher Ingenieure. *VDI Richtlinie 3633-11 Simulation von Logistik-, Materialfluss- und Produktionssystemen - Blatt 11 Simulation und Visualisierung.* Beuth, Deutschland, 2003.

[10] R. Schuhmann und W. Müller. *Visualisierung: Grundlagen und allgemeine Methoden.* Springer, Deutschland, 2000.

[11] Siemens. *Plant Simulation.* Abruf am 15. Januar, 2014. www.plm.automation.siemens. com/de_de/products/tecnomatix/plant_design/ plant_simulation.shtml.

[12] J. O. Henriksen. *The Power and Performance of Proof Animation.* In Proceedings of the

1997 Winter Simulation Conference, in Atlanta (USA), S. 574-580, 1997.

[13] S. Wenzel, J. Bernhard und U. Jessen. *A taxonomy of visualization techniques for simulation in production and logistics*. In Proceedings of the 2003 Winter Simulation Conference, in New Orleans (USA), S. 729-736, 2003.

[14] S. Straßburger, H. Seidel, R. Schady und S. Masik. *Werkzeuge und Trends der digitalen Fabrikplanung - Analyse der Ergebnisse einer Onlinebefragung*. In: Proceedings der 12. ASIM-Fachtagung "Simulation in Produktion und Logistik", in Kassel, S. 391-402. 2006.

[15] SISO Core Manufacturing Simulation Data Product Development Group. *Standard for: Core Manufacturing Simulation Data – UML Model*. Abruf am 05. März, 2014. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=31457.

[16] M. Johansson, B. Johansson, A. Skoogh, S. Leong, F. Riddick, Y.T. Lee, G. Shao und P. Klingstam. *A test implementation of the core manufacturing simulation data specification.* In: Proceedings of the 2007 Winter Simulation Conference, in Washington (USA), S. 1673-1681, 2007.

[17] SISO Core Manufacturing Simulation Data Product Development Group. *Standard for: Core Manufacturing Simulation Data – XML Representation.* Abruf am 05. März, 2014. http://www.sisostds.org/DigitalLibrary.aspx?Command=Core_Download&EntryId=36239.

[18] Wikipedia. *Java-Applet*. Abruf am 05. März, 2014. http://de.wikipedia.org/wiki/Java-Applet.

[19] N. Weschkalnies und S. Gasser. *Adobe Flash CS5: Das umfassende Handbuch*. Galileo Design, Deutschland, 2010.

[20] Microsoft Inc. *Silverlight Developer Center.* Abruf am 05. März, 2014. http://msdn.microsoft.com/de-de/silverlight/.

[21] C. Ullenboom. *Java 7 – Mehr als eine Insel: Handbuch zu den Java SE-Bibliotheken*. Galileo Computing, Deutschland, 2011.

[22] P. Lorenz und K.C. Ritter. *Skopeo: Platform-Independent System Animation for the W3*. In Proceedings of Simulation and Animation ‘97, in Magdeburg, S.12-23, 1997.

[23] S. Jobs. *Thoughts on Flash.* Abruf am 05. März, 2014. http://www.apple.com/hotnews/thoughts-on-flash/

[24] W. Möhle. *Der Preis ist (zu) hoch*. iX 2/2010, S. 3.

[25] W3C Working Group. *HTML 5- A vocabulary and associated APIs for HTML and XHTML W3C Candidate Recommendation 04 February 2014*. Abruf am 10. März, 2014, http://www.w3.org/TR/html5/.

[26] P. Kröner. *HTML5 – Webseiten innovativ und zukunftssicher.* Open Source Press, Deutschland 2011.

[27] Ecma International. *ECMAScript Language Specification 2011*. Abruf am 10. März, 2014, http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf

[28] F. Maurice. *CSS3 - Leitfaden für Webdesigner*. Addison-Wesley, Deutschland, 2011.

[29] MSDN. *So wird's gemacht: Auswählen zwischen SVG und Canvas*. Abruf am 15. März, 2014, http://msdn.microsoft.com/de-de/library/ie/gg193983(v=vs.85).aspx.

[30] JQuery Foundation. *jQuery*. Abruf am 14. Februar, 2014, http://jquery.com/.

[31] E. Rowell. *KineticJS*. Abruf am 14. Februar, 2014, http://kineticjs.com/

[32] F. Parzefall. *Konzeption und prototypische Implementierung eines webbasierten Pos-Simulation-Animationstools zur Verarbeitung von CMSD Daten*. Masterarbeit, TU Ilmenau, 2013.

[33] Wikipedia. *WebGL*. Abruf am 19. März, 2014. http://de.wikipedia.org/wiki/WebGL.