Entwicklung webbasierter graphischer Benutzeroberflächen für Simulationskerne

Svenja Hilbrich[1*], Katharina Gerdes[1], Johannes Hinckeldeyn[1], Jochen Kreutzfeldt[1]

[1] Institut für Technische Logistik, Technische Universität Hamburg, Theodor-Yorck-Straße 8, 21079 Hamburg, Deutschland; [*] *svenja.hilbrich@tuhh.de*

Abstract. Simulation spielt eine wichtige Rolle in vielen Disziplinen der Wissenschaft und Praxis. Die webbasierte Simulation bietet dabei Vorzüge gegenüber Desktopanwendungen und erlaubt es die Dauer zur Durchführung einer Simulationsstudie zu verringern. Häufig werden damit Client-Server-Strukturen bezeichnet, bei denen der Simulationskern Dienste auf einem Server bereitstellt und die Benutzeroberfläche durch eine Webseite realisiert wird. Solche Systeme werden allerdings nur selten entwickelt. Daher wird mit diesem Beitrag eine Anleitung zur Entwicklung einer webbasierten Benutzeroberfläche für Simulationskerne bereitgestellt. Anhand von Designfragen und Entscheidungsbäumen wird dabei schrittweise durch den Entwicklungsprozess geleitet, welcher zusätzlich durch ein Beispiel aus der Materialflusssimulation illustriert wird.

Einleitung

Simulation ist das „Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierbaren Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind“[1, S. 3]. Damit bietet eine Simulation deutliche Vorteile gegenüber Experimenten in einem realen System, zum Beispiel [2]:

- Die Erstellung eines Simulationsmodells ist kostengünstiger als die eines realen Systems.
- Das Verhalten des Systems kann ohne Auswirkungen auf das Realsystem untersucht werden.
- Die Dauer des Beobachtungszeitraums kann verlängert oder verkürzt werden.

Für Materialflusssysteme gilt zusätzlich, dass nachträgliche Veränderungen mit sehr hohen Kosten verbunden sein können. Daher ist die Simulation auch in der Planung und Dimensionierung von Materialflusssystemen ein weit verbreitetes Hilfsmittel [3].

Als webbasierte Simulation werden Simulationswerkzeuge bezeichnet, die über einen Webbrowser zugänglich sind. Vorteile dieser Form der Simulation liegen zum Beispiel in der Unabhängigkeit vom Betriebssystem und Rechenleistung des Anwenders, der Möglichkeit zur gemeinsamen Modellierung durch mehrere Anwender und der parallelen Ausführung von mehreren Simulationsläufen [4]. Diese Vorteile ermöglichen es den zeitlichen Aufwand für die Simulation zu verringern, sodass es sich bei der webbasierten Simulation um ein relevantes Forschungsthema handelt.

1 Stand der Technik

Die Verwendung eines Simulationswerkzeuges wird entscheidend durch dessen Bedienbarkeit bestimmt. Besonders für Anwender, welche keine Simulationsexperten sind bietet eine graphische Benutzeroberfläche die benötigte Unterstützung zur Erstellung und Auswertung eines Simulationsmodells. Ohne eine solche Schnittstelle würde der Modellierungsprozess vertiefende Kenntnis der für die Simulation verwendeten Programmiersprache erfordern, dies wiederum schließt Benutzer ohne entsprechendes Fachwissen aus [5].

Für Simulationswerkzeuge sind Benutzerschnittstellen traditionell in Form von Desktopanwendungen realisiert [6]. Das Thema webbasierte Simulation begann 1996 im Rahmen der Winter Simulation Conference das wissenschaftliche Interesse zu wecken [7]. Kuljis und Paul identifizierten bereits im Jahr 2000 eine Reihe an Anwendungsfällen, welche von einem webbasierten Simulationsansatz profitieren würden [8]. Syberfeldt, Karlsson et al. benennen mit den Stichworten Zugänglichkeit, Skalierbarkeit, Portabilität, Wartung, kontrolliertem Zugriff und Lizenzierung Vorteile der webbasierten Architektur gegenüber eines klassischen Ansatzes [6]. Trotz des seit Mitte der Neunzigerjahre bestehenden Interesses und der bestehenden Vorteile existieren nur wenige webbasierte Simulations-

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 23-32, DOI: 10.11128/arep.59.a59005

anwendungen [4]. In [9] werden unflexible Standard-Simulationssoftware und teure Lizenzen als wichtige Gründe, die einen Einsatz von Simulation verhindern, genannt. In ihrem Lösungsansatz im Rahmen des Projektes DREAM ("simulation based application Decision support in Real-time for Efficient Agile Manufacturing") wählen sie eine webbasierte open source Implementierung mit einem auf python aufbauenden Simulationskern. Neben dem Ansatz zur Verwendung und Adaption einer open source Simulationssoftware ist die Entwicklung eines auf allgemeinen Programmiersprachen aufbauenden individuellen Simulationskerns eine weitere Möglichkeit zur Überwindung der aufgezeigten Barrieren beim Einsatz von Simulation. In [10] wird die Entwicklung von Simulationssoftware, welche auf einer allgemeinen Programmiersprache aufbaut als von gleichbleibend signifikanter Bedeutung identifiziert. Für den Kreis der Anwender, die diesen Weg wählen stellt der vorliegende Beitrag eine strukturierte Anleitung zur Erstellung einer webbasierten Benutzeroberfläche zur Verfügung. Diese soll Entwickler individueller Simulationssoftware dabei unterstützen die Vorteile einer Benutzeroberfläche mittels einer webbasierten Implementierung für sich nutzbar zu machen. Die vorhandene Literatur beschränkt sich in diesem Bereich bisher auf die Dokumentation von individuellen prototypischen Ansätzen zur Entwicklung einer webbasierten Benutzeroberfläche. Dies zeigt sich auch in den Beiträgen von [11],[12] und [13]. Ziel dieses Beitrages ist es, die Verwendung von webbasierten Simulationsanwendungen in Wissenschaft und Praxis, durch die Beschreibung eines generischen Vorgehens zur Entwicklung einer webbasierten Benutzeroberfläche für einen Simulationskern, zu fördern.

2 Entwickelter Simulationskern für die Materialflusssimulation

Die Anleitung zur Entwicklung der Benutzeroberfläche wird in diesem Beitrag zum besseren Verständnis exemplarisch an einem Beispiel illustriert. Dazu wird ein bereits entwickelter Simulationskern verwendet, welcher in diesem Abschnitt beschrieben wird. In [14] wurde eine Simulationsmethodik zur Dimensionierung von Materialflusssystemen vorgestellt. Diese basiert auf der Modellierung logistischer Systeme mithilfe von generischen Modulen. Der Ablauf der Simulation ist dabei ereignisorientiert. Für diese Methodik wurde ein Simulationskern entwickelt, der die Simulation implementiert. Der Simulationskern wurde in der Programmiersprache python (Version 3.6) implementiert und zur Datenspeicherung wird eine sqlite-Datenbank verwendet. Für die Simulation müssen die folgenden Daten vom Anwender in der Datenbank gespeichert werden:

- Auftragsdaten, dienen der Erzeugung von bewegten logistischen Elementen (z. B. Paletten)
- Simulationsmodell, bestehend aus mehreren miteinander verknüpften Modulen
- Moduleigenschaften und Regeln, definieren das Verhalten der Module

Bisher wurden die Eingangsdaten manuell in der Datenbank abgespeichert. Die Ansteuerung des Simulationskerns erfolgte über das PC-Terminal.

3 Anforderungen an eine webbasierte Benutzeroberfläche für Simulationskerne

Eine Webanwendung besteht grundlegend aus drei Schichten: Die Präsentations-, Verarbeitungs- und Datenhaltungsschicht [15]. Die Präsentationsschicht entspricht in diesem Fall der Benutzeroberfläche, die Verarbeitungsschicht ist der Simulationskern zusammen mit einem Webserver und die Datenhaltungsschicht kann eine Datenbank sein. Für die Benutzeroberfläche wird eine Webseite (Client) verwendet. Der Webserver stellt Dienste für den Client zur Verfügung und realisiert somit auch die Schnittstelle zwischen dem Simulationskern und der Benutzeroberfläche.

Laut VDI 3633 ermöglichen Benutzeroberflächen den Modellaufbau, die Dateneingabe und die Ergebnisdarstellung. Der Modellaufbau kann dabei sowohl mithilfe von graphischen Elementen als auch mit Simulationssprachen erfolgen [1]. Außerdem sollte die Möglichkeit bestehen einzelne Modellbestandteile also Subsysteme zu entwickeln, zu speichern und später wiederzuverwenden. Dies kann den Prozess des Modellaufbaus verkürzen.

Die Dateneingabe umfasst sowohl Modelleigenschaften als auch Experimentdaten [1]. Diese Daten müssen entweder interaktiv in der Benutzeroberfläche eingegeben werden oder durch das Einlesen ganzer Dateien auf dem Server gespeichert werden, sodass durch den Simulationskern darauf zugegriffen werden kann.

Die Ergebnisse eines Simulationslaufes können statisch oder dynamisch dargestellt werden [1]. Die dynamische Darstellung zielt meist auf die Animation der Ereignisse während eines Simulationslaufes ab. Dies dient der Unterstützung der Ablaufkontrolle und der Validierung des Simulationsmodells. Die statische Ergebnisdarstellung dient vor Allem der Darstellung von Kennzahlen, die sich aus Analysen der Simulationsergebnisse ergeben. Die Ergebnisdarstellung kann sowohl parallel zum Simulationslauf als auch im Anschluss daran erfolgen. Außerdem sollte die Benutzeroberfläche dem Benutzer die Möglichkeit bieten die Rohdaten der Simulationsergebnisse zu beziehen.

Zusätzlich zu den genannten Punkten muss der Benutzer mithilfe der Benutzeroberfläche den Simulationskern steuern. Dies umfasst beispielsweise das Starten eines Simulationslaufes oder das Löschen von gespeicherten Daten. Insgesamt ergeben sich somit die folgenden Anforderungen für die Implementierung der Benutzeroberfläche:
Technologische Anforderungen:

- Entwicklungsmöglichkeit für eine Webseite
- Entwicklungsmöglichkeit für einen Webserver
- Realisierung des Datentransfers zwischen Webseite und Server bzw. Simulationskern

Funktionale Anforderungen:

- Steuerung des Simulationskerns
- Hochladen von Daten auf den Server
- Download von Daten auf den PC des Benutzers
- Entwicklung und Speicherung eines Simulationsmodells
- Entwicklung und Speicherung von Modellbestandteilen
- Animation des Simulationslaufes
- Visualisierung der Simulationsergebnisse

4 Anleitung zur Entwicklung einer webbasierten Benutzeroberfläche

Basierend auf den ermittelten Anforderungen, wird im Folgenden die Entwicklung einer webbasierten Benutzeroberfläche für einen bereits existierenden Simulationskern beschrieben. Dabei werden die allgemeinen Hinweise anhand konkreter Beispiele verdeutlicht. Zunächst müssen die verwendeten Technologien ermittelt werden. Anschließend können die gewünschten Funktionalitäten entwickelt werden.

In [23] wird auszugsweise der zur Implementierung der Benutzeroberfläche entwickelte Quellcode zur Verfügung gestellt.

4.1 Auswahl eines Frameworks zur GUI-Entwicklung

Für die Entwicklung von Webseiten können so genannte Webframeworks oder Bibliotheken als Unterstützung verwendet werden. Diese geben eine gewisse Struktur für die Entwicklung vor und verringern den Implementierungsaufwand, da sie Standardfunktionalitäten bereits anbieten. Beliebt für die Entwicklung von Webseiten sind zum Beispiel das Framework Angular [16] oder die die Javascript-Bibliothek React [17]. Ausführliche Vergleiche können in [18] und [19] gefunden werden. Im Folgenden werden zur besseren Lesbarkeit beide Alternativen als Framework bezeichnet.

Bei der Entscheidung zwischen den verfügbaren Frameworks sollten in einem ersten Schritt die folgenden Fragen berücksichtigt werden:

- **Rechtfertigt die Unterstützung durch das Framework den gegebenenfalls entstehenden Overhead?** Teilweise wird durch Verwendung eines Frameworks die Implementierung von sonst nicht benötigtem Code notwendig. Hier ist eine Abwägung sinnvoll, welches Framework am geeignetsten ist. Da es sich bei der Entwicklung einer Benutzeroberfläche für einen Simulationskern allerdings nicht um ein triviales Problem handelt, kann diese Frage in der Regel mit ja beantwortet werden.
- **Welcher Lizenz unterliegt das Framework?** Es ist grundsätzlich zu prüfen ob die Lizenz des Frameworks zu den Anforderungen an das Simuationswerkzeug passt.
- **Können alle erforderlichen Funktionalitäten mit dem Framework realisiert werden?** Ist dies nicht der Fall, muss ein anderes Framework gewählt werden. Die genannten Frameworks sind in der Regel aber geeignet, um alle hier benötigten Funktionalitäten zu implementieren.
- **Gibt es Vorkenntnisse in einem der Tools?** Vorkenntnisse können den zeitlichen Aufwand für die

Entwicklung der Webseite deutlich verkürzen, da die Einarbeitungszeit entfällt.

- **Wie groß ist die unterstützende Community?** Bei einer großen Community können mehr Problemlösungen online gefunden oder in Foren geteilt werden.
- **Welches Framework entspricht dem persönlichen Geschmack?** Bei der Wahl zwischen den etablierten Frameworks kann keine falsche Entscheidung getroffen werden, sodass persönliche Vorlieben berücksichtigt werden sollten.

Das Framework Angular bringt viele vordefinierte Lösungen für die Webseitenentwicklung mit, wie zum Beispiel ein Modul für den Datentransfer zwischen Client und Server. Außerdem wird durch die Aufteilung in Komponenten eine klare Struktur vorgegeben und es werden nur wenige Erweiterungen für die geplanten Funktionalitäten der Benutzeroberfläche benötigt. Da das Framework zusätzlich einer MIT-Lizenz unterliegt wurde sich in diesem Kontext für Angular als Framework für die Webseitenentwicklung entschieden. Für das Beispiel wurde Angular Version 5.2.11 verwendet.

Mit Angular können Single Page Anwendungen (SPA) mit HTML, CSS und Typescript entwickelt werden. Typescript ist ein Superset von Javascript und erlaubt die Implementierung der Funktionalität der Webseite. Angular basiert auf dem Konzept der Komponenten und der Dependency Injection. Komponenten beschreiben gekapselte wiederverwendbare Funktionalitäten bzw. Sichten (Views). Diese Komponenten können wiederum so genannte Services verwenden, die Funktionalitäten anbieten, die unabhängig von einer speziellen Sicht sind [20]. Das Angular-Tutorial „Tour of Heroes“ [21] bietet bei Bedarf einen guten Einstieg in das Framework.

Nach der Initialisierung eines neuen Angular-Projektes stehen in diesem die folgenden Dateien bereit:

- app.module.ts: Definition von global verwendeten Bibliotheken und Komponenten.
- app.component.html: Inhalt der obersten Komponente.
- app.component.css: Design der obersten Komponente.
- app.component.ts: Funktionalitäten der obersten Komponente.
- app.component.spec.ts: Test der obersten Komponente.

Da eine Benutzeroberfläche diverse Funktionalitäten erfüllen soll, muss diese für eine gute Übersichtlichkeit strukturierbar sein. Bei Webseiten ist die Verteilung von Inhalten auf mehrere Ansichten ein gängiges Hilfsmittel. Für die Benutzeroberfläche der Materialflusssimulation wurden der Datenaustausch, die Steuerung des Simulationskerns und die Modellierung voneinander getrennt. Bei Angular entspricht jede dieser Sichten einer eigenen Komponente, die je nach Bedarf in der Hauptkomponente angezeigt wird. Angular bietet für die Navigation zwischen den Sichten ein so genanntes `RouterModule` an (vgl. [22]). Um eine Navigationsleiste wie in Abb. 1 (oben) zu realisieren, wurde der in [23] gezeigte Code implementiert.

4.2 Aufsetzen eines Webservers

Um den Simulationskern für den Benutzer über die Webseite erreichbar zu machen, muss ein Webserver aufgesetzt werden. Analog zur Webseitenentwicklung gibt es Frameworks und Bibliotheken, die die Entwicklung des Webservers unterstützen. Für die Auswahl können die gleichen Fragen als Unterstützung beantwortet werden. Frameworks und Bibliotheken sind für alle gängigen Programmiersprachen verfügbar. Für python sind bekannte Beispiele Flask [24] und Django [25]. Ein ausführlicher Vergleich ist in [26] zu finden. Flask ermöglicht die Implementierung aller notwendigen Funktionalitäten für die hier entwickelte Anwendung. Dies ist vor Allem die Ansteuerung der Funktionen des Simulationskerns und die Abwicklung des Datentransfers zwischen Webseite und Server. Zudem wird auf Overhead im Vergleich zu Django verzichtet und unterliegt der BSD 3-Lizenz. Daher wurde Flask für die Entwicklung des Webservers verwendet.

Der Webserver wird mit dem in [23] gezeigten Code implementiert. Dabei ist die Standardeinstellung der Adresse unter der der Server erreichbar ist ’localhost’, also ein lokaler Server auf dem genutzten Rechner.

4.3 Datentransfer

Die Kommunikation und somit der Datentransfer zwischen Server und Client bzw. Simulationskern und graphischer Benutzeroberfläche wird mithilfe des HyperText Transfer Protokolls (HTTP) durchgeführt. Dieses basiert auf dem Anfrage-Antwort-Prinzip, wobei der

Abbildung 1: Screenshot des graphischen Modelleditors. Im oberen Bereich der Ansicht befindet sich die Navigationsleiste.

Client eine Anfrage sendet und der Server die Antwort zurück gibt. Primär werden für den Datentransfer zwei Methoden verwendet. Dies sind die GET- und die POST-Methode. Mit der GET-Methode werden Daten vom Server angefordert und mit der POST-Methode werden Daten an den Server übermittelt [27]. Nach der Bearbeitung der Anfrage des Clients sendet der Server eine Antwort an den Client. Dies können zum Beispiel vom Client angeforderte Daten sein. Werden bei der Kommunikation Anfragen vollständig nacheinander abgearbeitet, wird also auf die Antwort des Servers gewartet bevor die nächste Anfrage gesendet wird, spricht man von einer synchronen Kommunikation. Das Framework Angular verarbeitet die Anfragen asynchron, also mehrere Anfragen parallel. Bei Bedarf muss eine synchrone Verarbeitung also erzwungen werden.

Bei der reinen Kommunikation mittels HTTP is zu beachten, dass der Server nur auf Anfragen des Clients antworten kann und die Kommunikation nicht selbstständig steuert. Ein bidirektionaler Datenaustausch ist beispielsweise mit Websockets möglich ist. Dies wird im Folgenden allerdings nicht weiter betrachtet, da diese Kommunikation für die Grundfunktionalität der Benutzeroberfläche nicht notwendig ist. Bei Bedarf finden sich weitere Informationen zu Websockets in [15]. Typischerweise wird für den Datentransfer zwischen Client und Server das JavaScript Object Notation (JSON)-Datenformat verwendet [27].

Übermittlung von Daten zum Server

Für die Übermittlung und Anforderung von Daten von bzw. auf den Server werden bei Angular Services implementiert. Diese können anschließend von allen Komponenten verwendet werden. In [23] ist die Implementierung einer Post-Methode in Angular verfügbar. Neben Daten können auch ganze Dateien an den Server gesendet werden. Dies ist zum Beispiel notwendig, wenn der Benutzer eine ganze csv-Datei an den Server übermitteln möchte. Der Service dient auch dazu, die vom Server zurückgesendete Erfolgs- oder Fehlermeldung zu empfangen und gegebenefalls ein entsprechendes Feedback an den Nutzer zu initialisieren.

Eine POST-Methode kann zusätzlich zum reinen Datentransfer auch als Steuerungsmethode für den Simulationskern verwendet werden. Dazu wird die Aktions-Codezeile durch den gewünschten Aufruf der Funktion des Simulationskerns ersetzt.

Anforderung von Daten

Die Anforderung von Daten vom Server erfolgt analog zur Übermittlung von Daten. Eine Besonderheit stellt aber der Download von Dateien auf den Anwender-PC dar. Aus Sicherheitsgründen ist dies nur mithilfe eines Download-Dialogs möglich. Im Kontext der Benutzeroberfläche ist der Download beispielsweise notwendig wenn dem Anwender die Rohdaten der Simulationsergebnisse als .csv-Datei zur Verfügung gestellt werden sollen. In [23] ist der entsprechende Code abrufbar.

4.4 Graphischer Modelleditor

Die Entwicklung und Speicherung eines Simulationsmodells ist ein zentraler Bestandteil einer Benutzeroberfläche für einen Simulationskern. Zur Unterstützung

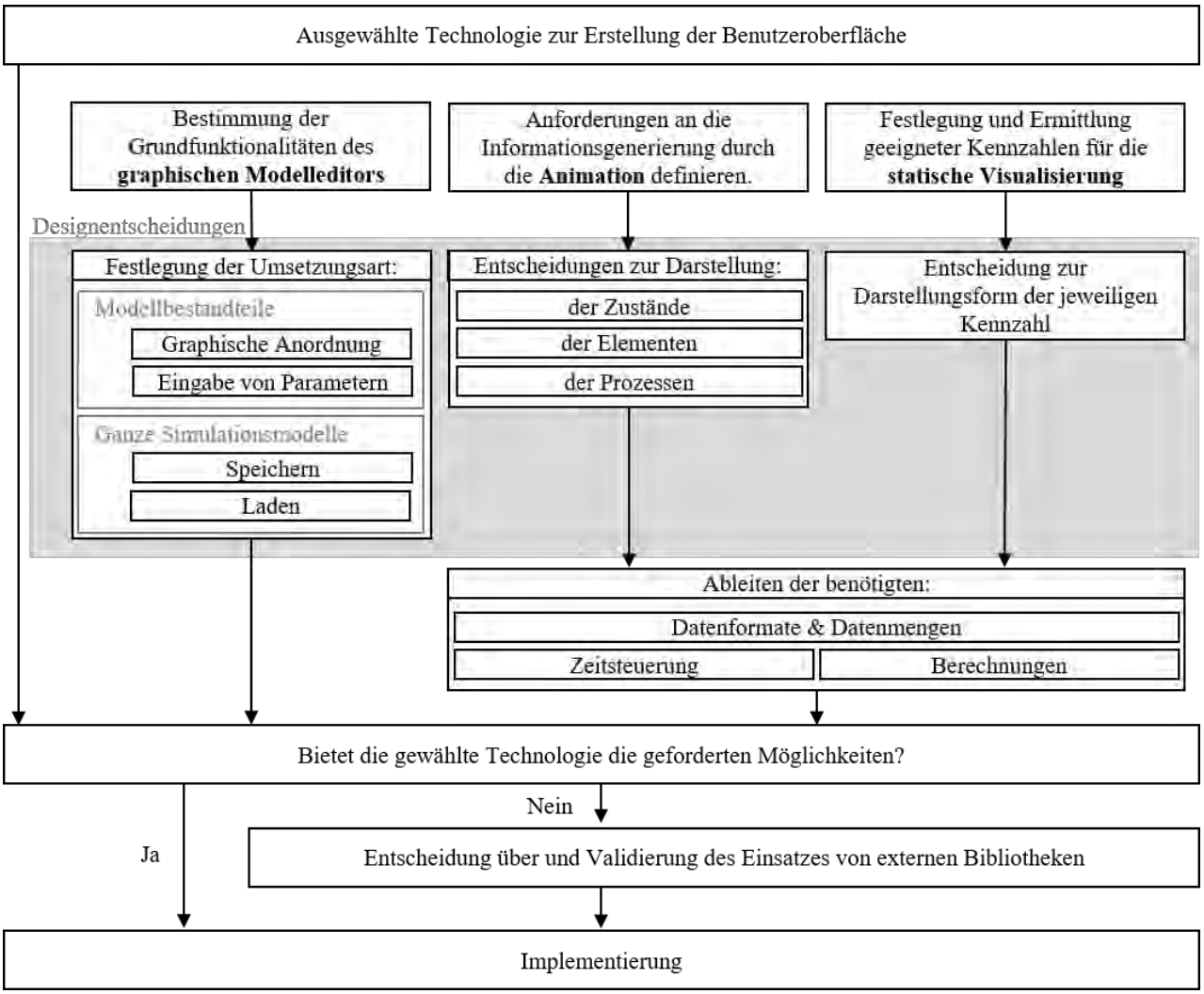

Abbildung 2: Entscheidungsbaum zur Entwicklung des Modelleditors (links), der Animation (mitte) und der statischen Visualisierung von Simulationsergebnissen (rechts).

der Modellierung haben sich graphische Modelleditoren als hilfreich erwiesen. Eine Übersicht über die Entscheidungen zur Entwicklung des Modelleditors ist in Abb. 2 dargestellt. Die benötigten Grundfunktionalitäten eines solchen Modelleditors sind

- Graphische Anordnung von Modellbestandteilen
- Eingabe von Parametern der Modellbestandteile
- Speichern von ganzen Simulationsmodellen
- Laden von gespeicherten Simulationsmodellen.

In vielen Bereichen in denen Simulationswerkzeuge eingesetzt werden, werden sehr große und komplexe Simulationsmodelle entwickelt. Um die Verständlichkeit und Übersichtlichkeit der Darstellung zu erhöhen, sollte zum Beispiel die Möglichkeit geschaffen werden, Bereiche des Simulationsmodells farblich hervorzuheben (vgl. den roten Bereich in Abb. 1). Außerdem wird empfohlen eine Zoom-Funktionalität zu implementieren, um den Gesamtmodellüberblick auch bei Modellen, welche die Größe des Editorfensters überschreiten, zu gewährleisten.

Für den in Abschnitt 2 beschriebenen Simulationskern wurde ein Modelleditor entwickelt, der in Abb. 1 dargestellt ist. Auf der rechten Seite der Seitenansicht kann die gewünschte Funktionalität ausgewählt werden,

in der Mitte werden die Simulationsmodelle zusammengestellt und rechts können die Moduleigenschaften eingestellt werden. Zusätzlich zu den genannten Funktionalitäten können so genannte Bausteine, welche aus mehreren Modulen bestehen, abgespeichert und wiederverwendet werden, rein graphische Elemente wie Rechtecke können verwendet werden, um das Modell visuell zu strukturieren und übersichtlicher zu gestalten und es besteht die Möglichkeit ein Bild des Simulationsmodells herunterzuladen. Bei der Aufteilung der Seite sollte die Modellierungsfläche möglichst groß gestaltet werden, um auch die Modelle möglichst großflächig darstellen zu können. Weitere Hinweise zur Gestaltung von Benutzeroberflächen finden sich zum Beispiel in [29] und [28].

Anordnung von Modellbestandteilen mit Drag & Drop

Modelle des entwickelten Simulationskerns bestehen aus generischen Modulen, welche verbunden werden, um ganze Materialflusssysteme abzubilden. Für die graphische Modellierung wurden daher Quadrate als Abbildung eines Moduls und gerichtete Verbindungslinien für die Darstellung der Verbindung zwischen diesem Modulen gewählt (vgl. Abb. 1). Drag & Drop erlaubt ein intuitives und visuell direkt erfassbares Anordnen von Modellbestandteilen und kann dadurch die Modellierung erleichtern. Die Verbindungen können durch Anklicken eines Moduls und Ziehen der Maus zu einem anderen Modul geschaffen werden. Für die Implementierung der Drag & Drop-Funktionalität wurde die Bibliothek jsplumb community edition [30] in Kombination mit jQuery [31] verwendet (vgl. [23]).

Die aktuelle Version Angular 9.0.0 bietet selbst ein Modul um drag & drop zu implementieren. Für zukünftige Entwicklungen sollte daher die Eignung dieser Bibliothek für die Implementierung der Drag & Drop-Funktionalität untersucht werden.

Parametereingabe

Für die Parametereingabe können Input-, Dropdown- und autofill-Felder verwendet werden. Soll der Benutzer nur aus einer vordefinierten Menge an Attributen auswählen können, bieten sich besonders Dropdown-Felder an, da diese nur vorgegebene Attribute anzeigen und zur Auswahl stellen. Diese Attribute können entweder vom Server angefragt oder in der Benutzeroberfläche hinterlegt werden. Inputfelder sollten nur verwendet werden, wenn dem Benutzer keine Vorgaben bezüglich der Eingabe gemacht werden. Autofill-Felder können in beiden Fällen verwendet werden. Dabei dient die Autovervollständigung zum einen als Unterstützung, wenn lange Eingaben notwendig sind.

Im betrachteten Beispiel werden die Eingabedaten direkt als Attribute der Modul-Elemente auf der Zeichenfläche gespeichert. Für die Ansicht der Moduleigenschaften auf der rechten Seite der Benutzeroberfläche wird eine `ComponentFactory` verwendet, die bei einem Klick auf ein Modul die Eigenschaften-Komponente erzeugt (siehe [23]).

Speichern & Laden des Simulationsmodells

Beim Speichern des Simulationsmodells wird dieses zunächst in ein json-Format überführt und anschließend an den Server übermittelt. Dort kann das Modell direkt in der für den Simulationskern benötigten Form abgespeichert werden. In diesem Fall wird es in der sqlite-Datenbank abgelegt. Unabhängig davon ob der Simulationskern die graphischen Informationen, also Position der einzelnen Modellbestandteile im Editor benötigt, sollte auf dem Server eine json-Datei abgelegt werden, die diese Informationen enthält. Damit reicht es bei dem Ladevorgang eines Modells aus, diese Datei an die Webseite zu übermitteln und einer aufwändige Datenverarbeitung bzw. Informationsrückgewinnung wird vorgebeugt. Das Speichern und Laden von Bausteinen funktioniert analog zum Speichern eines Modells.

Bild des Simulationsmodells speichern

Häufig ist es erwünscht ein Bild eines Simulationsmodells abzuspeichern. Hierzu stehen Bibliotheken zur Verfügung, die die Aufnahme eines Screenshots unterstützen. Im Beispiel wurde die Bibliothek `domtoimage` [32] verwendet (vgl. [23]). Mithilfe der Bibliothek werden alle Elemente, die einer ausgewählten Komponente zugeordnet sind, in einem Bild gespeichert.

4.5 Animation eines Simulationslaufs

Die Animation dient im Rahmen der Materialflusssimulation dem Verständnis von komplexen Zusammenhängen [1]. Es gilt zu entscheiden wie dies im konkreten Anwendungsfall bestmöglich umgesetzt werden soll. Dazu sollten grundsätzliche Entscheidungen über die Darstellung der Elemente, der Zustände und der Prozesse des Modells getroffen werden. Aus diesen leiten sich die Anforderungen an die Datenformate und -mengen sowie die Zeitsteuerung ab. Bietet die ausge-

wählte Technologie zur Erstellung der Benutzeroberfläche bereits die Möglichkeit zur Animationserstellung so wird diese implementiert. Anderenfalls geht diesem Schritt die Entscheidung und die Validierung zum Einsatz von externen Bibliotheken zur Animationsrealisierung voraus (vgl. Abb. 2). Innerhalb der betrachteten Materialflusssimulation s ind n ach d em geschilderten Vorgehen zur Umsetzung der Animation folgende Entscheidungen getroffen worden. Die Form und Anordnung der Modellelemente wird dem Modellaufbau entsprechend übernommen. Zustände werden zum einen durch „Füllstände“ und zum anderen durch sich auf den Modulen befindliche Elemente angezeigt. Die Materialflüsse z wischen d en M odulen w erden i hrer Richtung und ihrem mengenmäßigen Umfang gemäß durch Pfeile in unterschiedlicher Stärke angezeigt. In Abb. 3 ist eine schematische Übersicht zu den Darstellungsformen der Module gegeben. Bei der technischen Umsetzung der graphischen Darstellung wurden SVG (Scalable Vector Graphics) Elemente verwendet, da diese sich gut in das Angular Framework einfügen und manipulieren lassen, um somit die dynamischen Veränderungen abzubilden. Zur Steuerung der Systemzeit wurde die JavaScript Bibliothek `RxJS` (Reactive Extensions Library for JavaScript) [33], welche in das Angular Framework eingegliedert ist, verwendet.

4.6 Visualisierung von Simulationsergebnissen

Zur Unterstützung der Ergebnissinterpretation eines oder mehrerer Simulationsläufe sind diese aufzubereiten und graphisch darzustellen [1]. Die hierfür notwendigen Entscheidungen erfolgen analog zu denen für die Animation und sind in angepasster Form Abb. 2 zu entnehmen. Die Ermittlung geeigneter Kennzahlen und die Form ihrer graphischen Darstellung bilden die Grundlage für die abzuleitenden Datenformate und -mengen sowie die benötigten Berechnungen. Für die Umsetzung der Diagramme wurde das open source Framework ngx-charts eingebunden, da dieses speziell für die Verwendung in Angular-Projekten entwickelt wurde [34]. Eine exemplarische Darstellung ausgewählter Kennzahlen ist in Abb. 4 gezeigt.

5 Fazit

Jeder Entscheidung zur verwendeten Technologie oder Gestaltung einer Funktionalität der Benutzeroberfläche

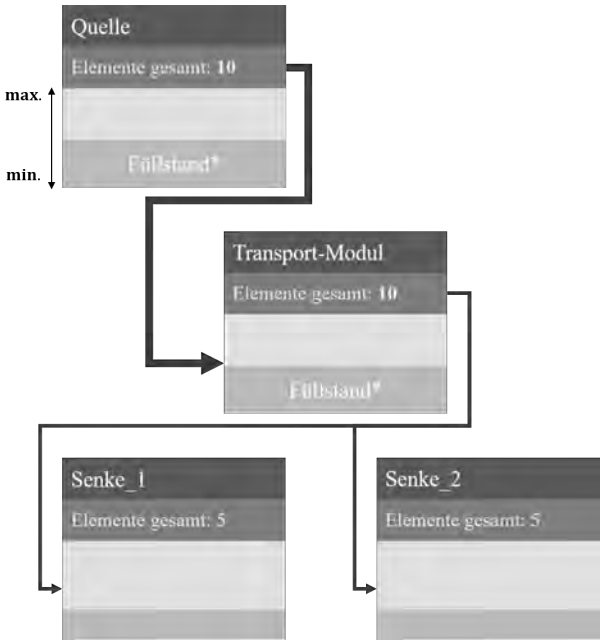

Abbildung 3: Übersicht zur gewählten Darstellungsform der Module in der Animation. * Der Füllstand zeigt den Grad der Auslastung des Moduls an.

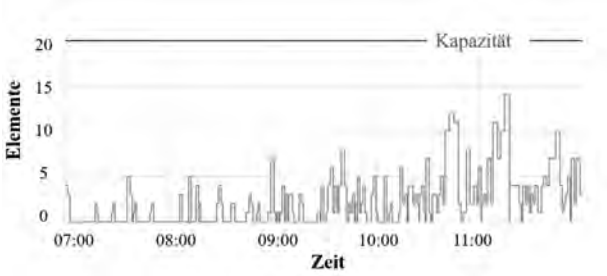

Abbildung 4: Exemplarische Ansicht der graphischen Darstellung ausgewählter Kennzahlen.

sollte diese Frage zu Grunde liegen: Wird dadurch die Bedienung des Simulationswerkzeugs erleichtert und der Aufwand für die Durchführung der Simulation verringert? Zusätzlich ergibt sich der in Abb. 5 dargestellte Entscheidungsbaum, welcher verwendet werden kann, um zu überprüfen ob eine Technologie für die Realisierung einer Funktionalität geeignet ist. Hierbei ist zu beachten, dass jeweils die Frage der Sinnhaftigkeit des Einsatzes einer Technologie vorher zu klären ist, ob also beispielsweise ein Webentwicklungsframework wie Angular überhaupt notwendig ist. Insgesamt ergibt sich damit für die Entwicklung der webbasierten Benutzeroberfläche der in Abb. 6 dargestellte Ablauf, wo-

bei Technologien für alle Funktionalitäten ausgewählt werden müssen. Der Entscheidungsbaum findet bei der Eignungsprüfung der Technologien Anwendung.

Bei der Entwicklung der Benutzeroberfläche für den in Abschnitt 2 vorgestellten Simulationskern, hat sich besonders die Wahl des Frameworks für die Entwicklung der Webseite als Entscheidung mit großer Tragweite herausgestellt. Dieses gibt die grundlegenden Rahmenbedingungen wie zum Beispiel die Programmiersprache und strukturelle Konzepte vor. Auf der einen Seite bedeutet das Framework eine Verringerung des Implementierungsaufwands, kann aber auf der anderen Seite auch die Implementierungsfreiheit einschränken, sodass dieses mit besonderer Sorgfalt ausgewählt werden sollte.

Weiterhin hat sich gezeigt, dass durch die Implementierung der webbasierten Benutzeroberfläche teilweise Mehraufwände im Vergleich zu Desktopanwendungen entstehen können. Zum Beispiel ist der Zugriff auf Daten und das zur Verfügung stellen von Dateien für den Anwender mit einem höheren Implementierungsaufwand verbunden. Allerdings übersteigen die Vorteile der WBS und die damit verbundene Zeiteinsparung diesen einmaligen zusätzlichen Aufwand, sodass webbasierte Benutzeroberflächen deutlich häufiger Anwendung finden sollten. Die hier beschriebene Anleitung kann dazu beitragen, dass zukünftig mehr webbasierte Benutzeroberflächen entwickelt werden.

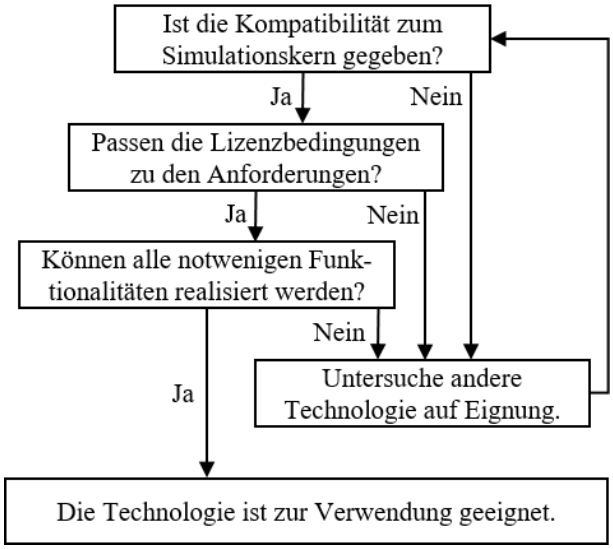

Abbildung 5: Entscheidungsbaum zur Auswahl geeigneter Technologien.

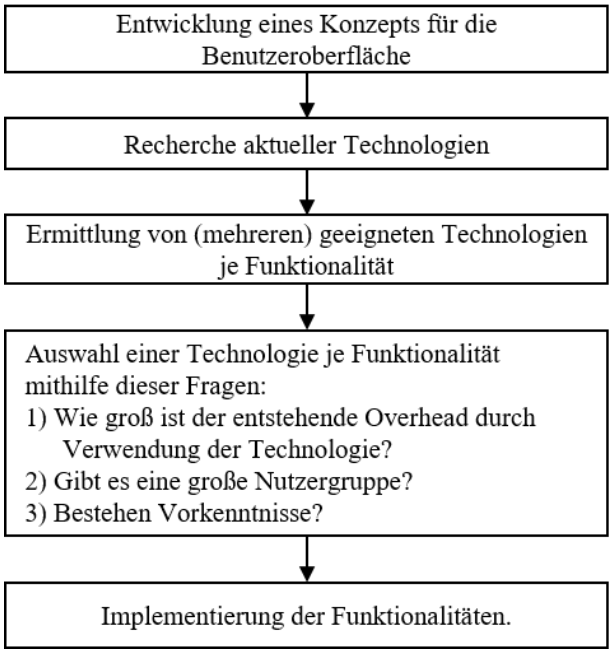

Abbildung 6: Ablauf der Entwicklung webbasierter Benutzeroberflächen für Simulationskerne.

Literatur

[1] Verein Deutscher Ingenieure e.V. *VDI 3633: Simulation von Logistik-, Materialfluss- und Produktionssystemen - Grundlagen*. VDI-Richtlinien. 2014.

[2] Bossel, H. *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*. 1st ed. Norderstedt: Books on Demand GmbH; 2004. 400 p.

[3] Wenzel, S. Simulation logistischer Systeme. In: Tempelmeier, H. editors. *Modellierung logistischer Systeme. Fachwissen Logistik*. 1st ed. Berlin, Heidelberg: Springer Vieweg;2018. p 1 -34.

[4] Byrne, J., Heavey, C., Byrne, P.J. A review of web-based simulation and supporting tools. *Simulation Modelling Practice and Theory*. 2010; 18(3): 253–276. doi: 10.1016/j.simpat.2009.09.013.

[5] Odhabi, H.I., Paul, R.J. Macredie, R.D. Developing a graphical user interface for discrete event simulation. In Medeiros, D.J., Watson, E.F., Carson, J.S., Manivannan, M.S., editors. Proceedings of the 1998 Winter Simulation Conference. *Winter Simulation Conference*; 1998 Dec; Washington, DC. p. 429–436.

[6] Syberfeldt, A., Karlsson, I., Ng, A., Svantesson, J., Almgren, T. A web-based platform for the simulation –optimization of industrial problems. *Computers*

Industrial Engineering. 2013; 64(4): p. 987–998

[7] Page, E.H., Buss, A., Fishwick, P.A., Healy, K.J., Nance, R.E., Paul, R.J. Web-Based Simulation: Revolution or Evolution? *ACM Transactions on Modeling and Computer Simulation*. 2000; 10(1): p. 3–17.

[8] Kuljis, J., Paul, R.J. A review of web based simulation: whither we wander? In Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A., editors. Proceedings of the 2000 Winter Simulation Conference. *Winter Simulation Conference*. 2000 Dec; Orlando, Florida. p. 1872–1881.

[9] Heavey, C., Dagkakis, G., Barlas, P., Papagiannopoulos, I., Robin, S., Mariani, M., Perrin, J. Development of an open-source discrete event simulation cloud enabled platform. In Tolk, A., Diallo, S.Y., Ryzhov, I.O., Yilmaz, L., Buckley, S., Miller, J.A., editors. Proceedings of the 2014 Winter Simulation Conference. *Winter Simulation Conference*; 2014 Dec; Savannah, Georgia. p. 2824–2835.

[10] Nance, R.E., Overstreet, C.M. History of computer simulation software: an initial perspective. In Chan, W.K., D'Ambrogio, A., Zacharewicz, G., Mustafee, N., Wainer, G., Page, E.H., editors. Proceedings of the 2017 Winter Simulation Conference.*Winter Simulation Conference*. 2017 Dec; Las Vegas, Nevada. p. 243–261.

[11] Gan, B.P., Liu, L., Ji, Z., Turner, S.J., Cai, W. Managing event traces for a web front-end to a parallel simulation. In Peters, B.A., Smith, J.S., Medeiros, D.J., Rohrer, M.W., editors. Proceedings of the 2001 Winter Simulation Conference. *Winter Simulation Conference*. 2001 Dec; Arlington, Virginia. p. 637–644.

[12] März, L., Interaktives Montageplanungssystem zur Online- Leistungssteuerung. In Dangelmaier, W., Laroque, C., Klaas, A., editors. Simulation in Produktion und Logistik. Entscheidungsunterstützung von der Planung bis zur Steuerung. *15. ASIM Fachtagung*. 2013 Oct; Paderborn. p. 661–668.

[13] Bergmann, S., Parzefall, F., Straßburger ,S. Webbasierte Animation von Simulationsläufen auf Basis des Core Manufacturing Simulation Data (CMSD) Standards. In Wittmann, J., Deatcu, C., editors. Tagungsband ASIM 2014. *22. Symposium Simulationstechnik*. 2014 Sep; Berlin. p. 63–70.

[14] Hilbrich, S., Köck, H., Hinckeldeyn, J., Kreutzfeldt, J. Entwicklung einer Simulationsmethodik zur schnellen Dimensionierung komplexer Materialflusssysteme. *Logistics Journal: Proceedings*. 2017; Vol 2017. doi: 10.2195/lj_Proc_hilbrich_de_201710_01.

[15] Abts, D. *Masterkurs Client/Server-Programmierung mit Java*. 5th ed. Wiesbaden: Springer Vieweg; 2019. 389 p.

[16] Google. Angular Docs. https://angular.io/docs. 2020.

[17] Facebook Inc. Getting Started. https://reactjs.org/docs/getting-started.html. 2020.

[18] Isaak Tappert. JavaScript Frameworks im Vergleich: Vue vs. Angular vs. React. https://entwickler.de/online/javascript/angular-vs-react-vs-vue-js-579921249.html. 2020.

[19] X-Company Pty Ltd. React vs angular: Their Biggest differences. https://x-team.com/blog/react-vs-angular/. 2019.

[20] Google. Introduction to Angular concepts. https://angular.io/guide/architecture. 2020.

[21] Google. Tour of Heroes app and tutorial. https://angular.io/tutorial. 2020.

[22] Google. Add in-app navigation with routing. https://angular.io/tutorial/toh-pt5. 2020.

[23] Hilbrich, S., Gerdes, K., Hinckeldeyn, J., Kreutzfeldt, J. WBS-GUI-Entwicklung. https://collaborating.tuhh.de/w-6/forschung/wbs-gui-entwicklung.2020.

[24] Pallets. Flask. https://flask.palletsprojects.com/en/1.1.x/. 2010.

[25] Django Software Foundation. Django. https://djangoproject.com. 2020.

[26] TestDriven Labs. Django vs. Flask in 2019: Which Framework to Choose. https://testdriven.io/blog/django-vs-flask/. 2020.

[27] Clow, M. *Angular 5 Projects: Learn to Build Single Page Web Applications Using 70+ Projects*. 1st ed. New York City: Apress; 2018. 458 p.

[28] MacDonald, D. *Practical UI Patterns for Design Systems*. 1st ed. New York City: Apress; 2019. 293 p.

[29] Shneiderman, B. et al. *Designing the user interface: strategies for effective human-computer intercation*. 6th ed. London: Pearson; 2017. 616 p.

[30] JSPLUMB UK LTD. jsplumb. http://jsplumb.github.io/jsplumb/home.html. 2020.

[31] The jQuery Foundation. jQueryAPI. https://api.jquery.com/. 2020.

[32] Saienko, A. DOM to image. https://github.com/tsayen/dom-to-image. 2017.

[33] Lesh, B. et al. RxJS. https://rxjs-dev.firebaseapp.com/. 2020.

[34] Swimlane. ngx-charts. https://swimlane.gitbook.io/ngx-charts/. 2020.