Fachhochschule Südwestfalen
University of Applied Sciences
Standort Hagen
Fachbereich Technische Betriebswirtschaft

Webtechnologie

Die Lehre von den Techniken des World Wide Web und von ihren Auswirkungen auf Wirtschaft und Gesellschaft

— Teil II: Fortgeschrittene Webtechniken und digitale Ökonomie —

Vorlesungsskript

Andreas de Vries

Version: 3. April 2023

Inhaltsverzeichnis

Teil IV

Fortgeschrittene Webtechniken und Konzepte

17 jQuery

Kapitelübersicht

JavaScript spielt in der professionellen Webprogrammierung eine ganz zentrale Rolle, insbesondere aufgrund seiner Fähigkeiten, DOM-Manipulationen und asynchrone HTTP-Requests dynamisch auszulösen. Da allerdings die dazu notwendigen Anweisungen in JavaScript recht aufwendig sind — mit Funktionen wie `getElementById` zur DOM-Manipulation oder dem im wesentlichen immer gleichen Dreischritt zur Implementierung eines asynchronen HTTP-Requests — wurden immer wieder spezielle JavaScript-Bibliotheken entwickelt, um die Programmierung zu vereinfachen. Eine der wichtigsten und verbreitesten dieser Bibliotheken ist jQuery, das zu einem De-Facto-Standard der Webprogrammierung geworden ist: Nach W3Techs (`http://w3techs.com/`) wird jQuery auf fast 70 % aller Webauftritte eingesetzt.

17.1 Entstehungsgeschichte, Lizenz, Dokumentation

jQuery wurde 2006 von John Resig veröffentlicht und ist seit April 2013 in der Version 2 verfügbar. Da diese Version nur noch mit modernen Browserversionen kompatibel ist (also insbesondere nicht mit dem Internet Explorer bis einschließlich Version 8), wird die Version 1 parallel weitergeführt. jQuery ist als freie Software unter der MIT-Lizenz (`https://tldrlegal.com/license/mit-license`) verfügbar, darf also auch für kommerzielle Zwecke verwendet werden, wobei jede Kopie ihr Copyright beinhalten muss.

Die API-Dokumentation von jQuery ist unter

`http://api.jquery.com/`

verfügbar, ein sehr gutes Tutorial unter `http://www.w3schools.com/jquery/`.

17.2 Einbindung der Bibliothek

Da jQuery eine JavaScript-Bibliothek ist, benötigt man zur dessen Verwendung keine besondere Installation, sondern bindet es per Hyperlink in das HTML-Dokument ein. Die aktuellste Variante findet man unter

`http://code.jquery.com/`

Dort ist jede Version in einer „unkomprimierten“ und einer „minifizierten“ Fassung verfügbar. Die erste ist kommentiert und formatiert und somit gut lesbar, die andere ist unformatiert und ohne Kommentare, aber mit deutlich weniger Speicherbedarf. Die minifizierte Fassung eignet sich also zur reinen Nutzung besser als die unkomprimierte. Man kann nun die gewünschte Version der Bibliothek einfach auf den eigenen Server herunterladen und sie dann per Link einbinden, man kann aber auch die Originalbibliothek einfach verlinken. Die minifizierte Version 3.3.1 beispielsweise kann man im Head des HTML-Dokuments mit

```
<script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
```

einbinden. Empfohlen ist die Einbindung mit einer Überbrüfung der „Subresource Integrity“ des W3C:

```
<script
  src="http://code.jquery.com/jquery-3.3.1.min.js"
  integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
  crossorigin="anonymous"
></script>
```

17.3 Syntax

Basis einer jQuery-Anweisung ist der Aufruf der Funktion `$`, oder synonym `jQuery`, mit einem *Selektor* und einer *Aktion*:

`$(`*Selektor*`).`*Aktion*`(...);` oder synonym `jQuery(`*Selektor*`).`*Aktion*`(...);`

(Gebräuchlicher ist die Verwendung von `$`.) Der *Selektor* ist hierbei ein CSS-Selektor zur Auswahl eines oder mehrerer HTML-Elemente, die *Aktion* ist eine auszuführende Funktion aus der jQuery-Bibliothek. Für den Selektor gibt es im Wesentlichen vier Varianten, beispielhaft mit der Aktion `hide()` aufgelistet, die die ausgewählten HTML-Elemente per CSS verstecken lässt:

`$("#test").hide()`	versteckt alle HTML-Elemente mit der ID `"test"`.
`$(".test").hide()`	versteckt alle HTML-Elemente der Klasse `"test"`.
`$("p").hide()`	versteckt alle `<p>`-Elemente.
`$(this).hide()`	versteckt das aktuelle Element (funktioniert normalerweise nur innerhalb eines selektierten Elements);

Diese Anweisungen müssen in eine anonyme Funktion (auch „Callback-Funktion“ oder „Lambda-Ausdruck“ genannt) eingepackt werden:

```
$(function(){
    ...
});
```

oder äquivalent

```
$(document).ready(function(){
    ...
});
```

und sollten im `<head>` des HTML-Dokuments programmiert (bzw. als externe JS-Datei dort eingelesen) werden. Auf diese Weise wird garantiert, dass das gesamte Dokument vollständig geladen ist, bevor jQuery-Code ausgeführt wird. Ein Aufruf dieser Art entspricht dem in JavaScript oft verwendeten Programmiermuster IIFE (*immediatley invoked function expression*).

Neben diesen „Selektor-Aktion“-Anweisungen kann man in jQuery auch direkt vordefinierte Nutzfunktionen durch

`$.`*Nutzfunktion*`(...);` oder synonym `jQuery.`*Nutzfunktion*`(...);`

aufrufen. In der API-Dokumention (`http://api.jquery.com/`) sind die vorhandenen Nutzfunktionen mit

`jQuery.`*Nutzfunktion* (...)

aufgeführt. Die auf selektierte Elemente möglichen Aktionen sind dagegen einfach mit

.*Aktion* (...)

beschrieben.

17.4 DOM-Manipulation mit jQuery

Als erstes Programmbeispiel mit jQuery betrachten wir eine Version des Programms aus Kapitel 16 (Listing 16.2), das nach Klicken der Schaltfläche ein neues `<li>`-Element an die geordnete Liste `<ol>` hängt:

Listing 17.1: DOM Manipulation mit jQuery

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script>
   $(function() {
      $("button").click( function () { // Event-Handler für alle Buttons
         if ($(this).val() === "hinzufuegen") {
            $("#liste")
               .append($("<li>Die Zeichen an der Wand</li>"));
         } else if ($(this).val() === "loeschen") {
            $("#liste")
               .children()
               .last()
               .remove();
         }
      });
   });
  </script>
</head>
<body>
  <button value="hinzufuegen">Neuer Text</button>
  <button value="loeschen">Löschen</button>
```

```
  <ol id="liste"></ol>
</body>
</html>
```

Ein direkter Vergleich der jQuery-Anweisungen mit einem äquivalenten Programm in „reinen“ JavaScript zeigt die Philosophie und die funktional orientierte Sicht von jQuery. Das jQuery Programm

```
$(function() {
   $("button").click(function(){ // Event-Handler für alle Buttons
      console.log("$(this).val(): " + $(this).val());
      if ($(this).val() === "hinzufuegen") {
         $("#liste")
            .append($("<li>Die Zeichen an der Wand</li>"));
      } else if ($(this).val() === "loeschen") {
         $("#liste")
            .children()
            .last()
            .remove();
      }
   });
});
```

würde in JavaScript (fast) äquivalent wie folgt lauten:

```
var modifizieren = function(aktion) {
   var wurzel = document.getElementById('liste');

   if (aktion === "hinzufuegen") {
      var neu = document.createElement('li');
      wurzel.appendChild(neu);
      var neuerText = document.createTextNode("Die Zeichen an der Wand.");
      neu.appendChild(neuerText);
   } else if (aktion === "loeschen") {
      if (wurzel.hasChildNodes()) {
         wurzel.removeChild(wurzel.lastChild);
      }
   }
};
```

mit den Ereignisbehandlern

```
<button onclick="modifizieren('hinzufuegen');">Neuer Text</button>
<button onclick="modifizieren('loeschen');">Löschen</button>
```

Im jQuery-Quelltext wird die objektorientierte API des DOM derart gekapselt, dass sie eine „fluente“ („fließende“) Programmierung hintereinandergeschalteter Funktionen erlaubt, wie in Zeile 15 bis 17 in Listing 17.1. Außerdem brauchen die Ereignisbehandler nicht mehr im HTML-Quelltext in den betreffenden Elementen implementiert zu werden, sondern über die Selektoren ausschließlich in JavaScript. Damit ist eine saubere Trennung zwischen der Ablaufdynamik und den HTML-Inhalten möglich.

17.5 CSS-Formatierungen

Etwas einfacher und auch durchweg gebräuchlicher als direkte DOM-Manipulationen, wie im vorigen Abschhnitt besprochen, sind dynamische Veränderungen von Formatierungen über CSS. Hier stehen in jQuery mit `hide`, `show` und `toggle` (umschalten) drei wirkmächtige Funktionen zur Verfügung, die hier stellvertretend für dynamische Effekte aufgelistet seien. Die Funktionen modifizieren dabei gar nicht die Struktur des DOM, sondern ändern lediglich die `style`-Attribute der HTML-Elemente. Als Beispiel sei das folgende jQuery-Programm gelistet, das durch Klicken der Schaltfläche „Umschalten“ bewirkt, dass alle Abätze des HTML-Dokuments angezeigt werden oder nicht:

Listing 17.2: Zeigen und Verbergen von HTML-Elementen mit jQuery

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script>
   $(function() {
      $("button").click(function(){
          $("p").toggle();
      });
   });
  </script>
</head>
<body>
  <button>Umschalten</button>
  <p>Dies ist ein erster Absatz.</p>
  <p>Dies ist ein anderer kleiner Absatz.</p>

  <div>
  Das ist ein weiterer Text.
  </div>
</body>
</html>
```

Untersucht man die HTML-Elemente mit den Entwicklertools des Browsers, so erkennt man, dass lediglich das `style`-Attribut aller `<p>`-Elemente verändert wird, entweder von `<p style="display: block;"` (sichtbar) auf `<p style="display: none;"` (unsichtbar), oder umgekehrt.

Mit jQuery stehen aber auch Funktionen für Effekte mit CSS zur Verfügung, die lange und komplexe JavaScript-Anweisungen bündeln. Im folgenden Beispielprogramm wird ein angeklickter Listeneintrag zunächst rot gefärbt und verblasst dann langsam, bevor er nach 3 Sekunden (scheinbar) verschwindet.

Listing 17.3: Animationsartiger CSS-Effekt mit jQuery

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script>
```

```
   $(function() {
      $("li").click( function() {
         $(this).css("color", "red").fadeOut(3000);
      });
   });
  </script>
</head>
<body>
<ul>
  <li>Clyde</li>
  <li>Fred</li>
  <li>Bonny</li>
</ul>
</body>
</html>
```

Lassen Sie das Programm ablaufen und beobachten Sie dabei im Entwicklertool des Firefox („Inspector“) oder des Chrome Browsers („Elements“) die sich dynamisch verändernden Einstellungen der CSS-Anweisungen!

17.6 AJAX mit jQuery

Die einfachste Möglichkeit zur Implementierung von AJAX-Ereignissen ist über die Nutzfunktion `$.ajax` (bzw. `jQuery.ajax`). Es gibt zwei Versionen (siehe `http://api.jquery.com/jQuery.ajax/`), die gebräuchlichere erwartet als Arguent ein JSON-Objekt `setting` mit den Einstellungen der asynchronen HTML-Anfrage:

- `type`: Die Methode der HTML-Anfrage, also `"GET"` oder `"POST"`
- `url`: Der URL der HTML-Anfrage
- `data`: Die zu übermittelnden Daten der HTML-Anfrage,
- `success`: Der Ereignisbehandler bei Erfolg der HTML-Anfrage.

Beispielhaft sei hier ein einfacher asynchroner Aufruf des Spiegelskripts auf Hägar beschrieben:

Listing 17.4: AJAX mit jQuery

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8"/>
  <script src="http://code.jquery.com/jquery-3.3.1.min.js"></script>
  <script>
$.ajax({
   type: "POST",
   url: "http://haegar.fh-swf.de/AJAX/spiegel.php",
   data: "y=x^2-1&x=1",
   success: (response) => {
      $("#ausgabe").html("<h2>Serverantwort</h2>" + response);
   }
```

```
});
  </script>
</head>
<body>
  <p id="ausgabe"></p>
</body>
</html>
```

Während also mit jQuery ein einziger Funktionsaufruf zur Erzeugung der asynchronen HTTP-Anfrage genügt,

```
$.ajax({
   type: "POST",
   url: "http://haegar.fh-swf.de/AJAX/spiegel.php",
   data: "y=x^2-1&x=1",
   success: function(response){
      $("#ausgabe").html("<h2>Serverantwort</h2>" + response);
   }
});
```

muss man in JavaScript für denselben Zweck 7 Anweisungen programmieren:

```
var request = new XMLHttpRequest();
request.onreadystatechange = function() { // Event Handler
  if (request.readyState === 4 && request.status === 200) {
    document.getElementById("ausgabe").innerHTML =
       "<h2>Serverantwort</h2>" + request.response;
  }
};
request.open("POST", "http://haegar.fh-swf.de/AJAX/spiegel.php");
var data = new FormData();
data.append("y","x^2 - 1");
data.append("x",1);
request.send(data);
```

Zu beachten ist dabei, dass in jQuery die AJAX-Anweisungen aufgrund der verzögerten Ausführung im `<head>` des Dokuments programmiert werden können, während die reinen JavaScript-Anweisungen *nach* dem `<p>`-Element geschrieben werden sollten, also wenn es zum Ausführungszeitpunkt bekannt ist, oder als Funktionsaufruf des Ereignisbehandlers `onload` im `<body>`-Element.

Eine genauere Beschreibung der für die Funktion `$.ajax` möglichen Parameter sind in der API-Dokumentation unter

`http://api.jquery.com/jquery.ajax/`

beschrieben. Die wichtigsten Parameter sind bereits in obigem Listing 17.4 aufgeführt. Bemerkenswert ist außerdem der Parameter `dataType`, der als Standard per *Intelligent Guess* eines der Formate `xml`, `json`, `script` (für JavaScript-Quelltext) oder `html` auswählt.

17.7 jQuery Mobile

jQuery Mobile ist eine für Wischgesten optimierte JavaScript-Bibliothek, die auf jQuery und auf eine spezifische CSS-Bibliothek aufbaut. Sie ist kompatibel mit den gängigen Smartphones

und Tablets. Sie erlaubt es durch gekapseltes JavaScript, vorwiegend mit HTML-Elementen und geeigneten Attributen dynamische Apps zu programmieren. Als ein einführendes Beispiel sei die folgende App, die zwei miteinander verlinkte Seiten mit einfachen Effekten darstellt:

Listing 17.5: Eine zweiseitige App für mobile Endgeräte

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>jQuery Mobile Example</title>
  <!-- Include meta tag to ensure proper rendering and touch zooming: -->
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <!-- Include jQuery Mobile stylesheets: -->
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css"/>
  <!-- Include the jQuery library: -->
  <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
  <!-- Include the jQuery Mobile library: -->
  <script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
</head>
<body>
<div data-role="page" id="pageone">
  <div data-role="header">
    <h1>Welcome To My Homepage</h1>
  </div>

  <div data-role="main" class="ui-content">
    <h1>Welcome!</h1>
    <p>I'm glad to be a mobile app developer.</p>
    <p>If you click on the link below, it will take you to Page Two.</p>
    <a href="#pagetwo">Go to Page Two</a>
  </div>

  <div data-role="footer">
    <h1>Footer Text</h1>
  </div>
</div>

<div data-role="page" id="pagetwo">
  <div data-role="header">
    <h1>Welcome To My Homepage</h1>
  </div>

  <div data-role="main" class="ui-content">
    <p>Now you're on Page Two.</p>

    <div data-role="collapsible" data-collapsed="true">
      <h1>Click me - I'm collapsible!</h1>
      <p>I'm not expanded by default.</p>
    </div>

    <p>If you click on the link below, it will take you to Page Two.</p>
    <a href="#pageone">Go to Page One</a>
  </div>

  <div data-role="footer">
    <h1>Footer Text</h1>
  </div>
</div>
</body>
</html>
```

Die App ist verfügbar unter

`http://haegar.fh-swf.de/Webtechnologie/jQuery-mobile.html`

Testen Sie sie mit Ihrem mobilen Endgerät oder mit den Entwicklertools des Google Chrome!

Im `<head>` des Dokuments werden die notwendigen CSS- und JavaScript-Bibliotheken geladen, bevor mit `<div>`-Elementen über das Attribut `data-role` die Struktur einer Seite festgelegt wird:

```
<div data-role="page" id="pageone">
  <div data-role="header">
    ...
  </div>

  <div data-role="main" class="ui-content">
    ...
  </div>

  <div data-role="footer">
    ...
  </div>
</div>
```

Jedes `<div>`-Element mit der `data-role="page"` sollte dabei eine eigene ID bekommen. Die Rollen von `"header"`, `"main"` und `"footer"` sind (hoffentlich) selbsterklärend und können bei Betrachtung der App nötigenfalls erschlossen werden.

Weitere Informationen: Die Dokumentation zur API befindet sich unter

`http://api.jquerymobile.com/`,

ein gutes Tutorial unter `http://www.w3schools.com/jquerymobile/`.

18 Angular

Kapitelübersicht

Angular ist ein Webframework zur Entwicklung clientseitiger JavaScript-Anwendungen, insbesondere *Einzelseiten-Webanwendungen* nach dem Entwurfsmuster *Model View ViewModel (MVVM)*. Angular wird seit 2009 von Google als quelloffenes Projekt entwickelt, anfangs unter dem Namen AngularJS. In diesem Kapitel werden die grundlegenden Konzepte und eine Einführung in die Webentwicklung mit Angular behandelt.

18.1 Grundlegende Konzepte

18.1.1 Einzelseiten-Webanwendungen (SPA)

Eine *Einzelseiten-* oder *Single-Page-Webanwendung* (englisch *Single-page Web Application, SPA*) ist eine Webanwendung, die aus einem einzigen HTML-Dokument besteht und deren Inhalte dynamisch nachgeladen werden. Diese Art von Web-Architektur steht im Gegensatz zu klassischen Webanwendungen, welche aus mehreren, untereinander verlinkten HTML-Dokumenten bestehen. Eine Einzelseiten-Webanwendung ermöglicht eine Reduzierung der Serverlast, gibt dem Anwender das Gefühl einer klassischen Desktopanwendung und erlaubt die Umsetzung von Webclients, die auch temporär offline verwendbar sind. Insbesondere aus letzterem Grund eignen sich SPAs für den Einsatz bei störanfälligen und oft unterbrochenen Verbindungen mobiler Endgeräte.

18.1.2 Model-View-ViewModel (MVVM)

Eines der wichtigsten Architekturmuster des Software Engineering ist das *Model-View-Controller (MVC)*. Es zählt zu den Mustern der interaktiven Systeme, die allgemein Mensch-Maschine-Interaktion strukturieren. In den 1980er Jahren von Trygve Reenskaug für die Program-

miersprache Smalltalk entwickelt, trennt es die Verantwortlichkeiten von Datenhaltung (Model), Darstellung (View) und Ablauflogik (Controller) und lagert sie in separate Schichten aus (Abbildung 18.1 rechts).

In einer dynamischen Webanwendung kann die strikte Trennung von Ablauflogik (Controller) und Darstellung (View) allerdings nicht gehalten werden, da der Controller HTML-Code kennen und teilweise auch generieren muss, der aber grundsätzlich Teil der View-Komponente ist. Da der Controller einer Webanwendung klassischerweise auf dem Server läufr, trägt dieser am Ende die Hauptlast der Anwendung. Darunter können die Interaktivität und das Antwortverhalten bei hoher Nutzung leiden, während der Browser vergleichsweise wenig zu tun hat. Er ist in diesem Falle nur noch ein sogenannter *Thin Client*.

Mit dem Muster *Model-View-ViewModel (MVVM)* wird der Controller durch eine Proxy-Schicht auf dem Client (!) ersetzt, die *ViewModel* genannt wird. Das ViewModel ist gewisser-

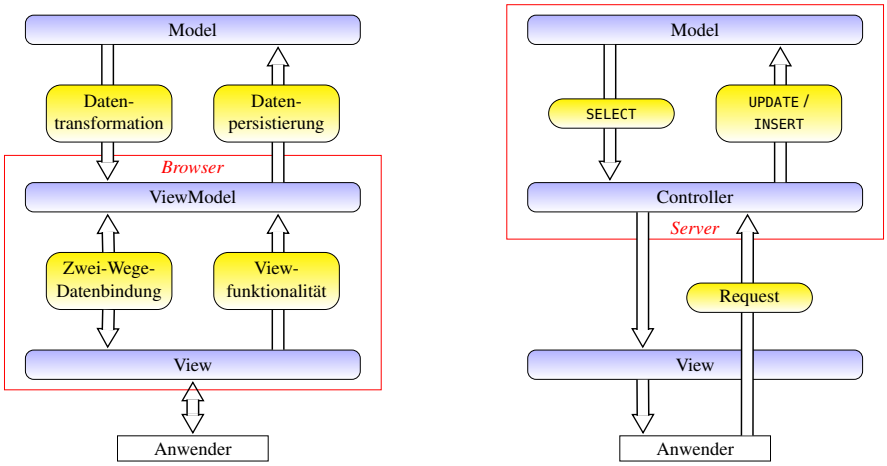


Abbildung 18.1: Das MVVM-Muster (links) für eine Webanwendung, im Vergleich dazu das MVC-Muster (rechts). Da bei MVC die Hauptlast der Webanwendung auf dem Server läuft, ist der Browser in diesem Fall ein Thin Client. Bei MVVM dagegen hat er mit der Poxy-Schicht MVVM einen großen Teil der Arbeitslast und ist daher in diesem Fall ein Fat Client.

maßen das Datenmodell der View und verwaltet lediglich diejenigen Daten, die tatsächlich für die Darstellung gebraucht werden. Oft müssen die Daten auch auf bestimmte Art und Weise transformiert werden, bevor sie zur Anzeige kommen; auch hierfür ist das ViewModel zuständig. Außerdem definiert es die innerhalb der Anzeige benötigte Funktionalität: Dort müsste beispielsweise ein Ereignisbehandler definiert werden, der auf einen Button-Klick in der View reagiert.

So plausibel das MVVM-Muster auch auf den ersten Blick erscheint, wirft es sofort ein großes Problem auf: die Datenredundanz zwischen Model, ViewModel und View. Wie kann dieses Problem gelöst werden? Betrachten wir dazu im nächsten Abschnitt einen Ansatz, der sich als großenteils erfolgreich realisierbar herausgestellt hat.

18.1.3 Datenbindung (*data binding*)

Aufgrund der engen Verzahnung von Model, ViewModel und Model und dadurch implizierten grundsätzlichen Datenredundanz ergibt sich sofort das Problem der Synchronisation der Schichten. D.h. jede Datenaktualisierung, sei sie im Model durch die Datenbank oder im View durch eine Anwendereingabe, muss in den jeweils anderen Schichten nachvollzogen werden. Ansonsten kann es sehr schnell zu Inkonsistenzen der Daten kommen, beispielsweise weil mittlerweile ein Anwender einen Dateninhalt geändert hat, den etwas später ein anderer verändern

will: Hat zum Beispiel ein Anwender den letzten vorhanden Artikel im Bestand gekauft, so muss das ViewModel jedes anderen Anwenders sofort angepasst werden, sonst wird dieser Artikel möglicherweise mehrfach verkauft.

Als ein Konzept zur Lösung dieses Synchronisationsproblems des MVVM-Musters hat sich die Datenbindung, vor allem die „Zwei-Wege-Datenbindung“, etabliert. Wir werden dieses gleich im nächsten Programmierbeispiel betrachten. Die *Zwei-Wege-Datenbindung* ist im MVVM ein Mechanismus zur automatischen Synchronisation von Viewmodell und View bei Änderung von Daten, sei es durch den Nutzer in der View oder durch andere Nutzer oder Prozesse im mit dem Model synchronisierten Viewmodell.

Normalerweise muss man eine solche Datenbindung zwischen den Schichten mühsam programmieren. Beispielsweise müssen für alle eingaberelevanten DOM-Elemente jeweils extra ein Ereignisbehandler registriert werden, der die betroffenen Variablen des Models ändert. Entsprechend muss auch die Logik des umgekehrten Weges programmiert werden, also die Aktualisierung der DOM-Elemente nach einer Änderung des Daten des Models. Auf diese Weise entsteht bereits für relativ einfache Anwendungen jede Menge sogenannter „Boilerplate-Code“ (etwa „Textbaustein-“ oder „Klebe-Code“), der die eigentliche Anwendungslogik verschmutzt.

18.1.4 Dependency Injection

Ein grundlegendes Problem komponentenbasierter Softwaresysteme, die wartbar, testbar und komponentenweise austauschbar sein sollen, ist die Auflösung von Abhängigkeiten der Komponenten untereinander. Eine einfache Abhängigkeit ist bereits gegeben, wenn eine Komponente eine andere für ihren Ablauf benötigt. In einer objektorientierten Programmiersprache beispielsweise wäre dies der Fall, wenn ein Objekt zu seiner Erzeugung ein anderes verwendet. Durch solche Abhängigkeiten kann ganz schnell ein verschlungenes Netz von Abhängigkeiten entstehen, das eine Modifikation des Softwaresystems oder den Austausch einzelner Komponenten praktisch unmöglich macht.

Im Software Engineering hat sich daher in den 2000er Jahren das auf Martin Fowler[1] zurück gehende Entwurfsmuster der *Dependency Injection (DI)* etabliert. Ihr liegt das abstrakte Prinzip der Umkehrung des Kontrollflusses (*Inversion of Control (IoC)*) zugrunde, also zum Beispiel eine Umkehrung der Aufrufe von Funktionen oder der Ablaufsteuerung eines Programms. (Getreu dem Hollywood-Prinzip, “Don’t call us, we’ll call you!”[2]) Eine typische Realisierung dieses Prinzips sind beispielsweise Event Handler, da sie Funktionen implementieren, deren Aufruf nicht in der Anwendung selbst geschieht, sondern aus einer der Programme der API-Bibliothek erfolgt, initiiert von einem spontanen Event. Ermöglicht wird eine solche Umkehrung des Kontrollflusses beispielsweise durch Interfaces, Plug-Ins oder Callback-Funktionen. Ein anderes

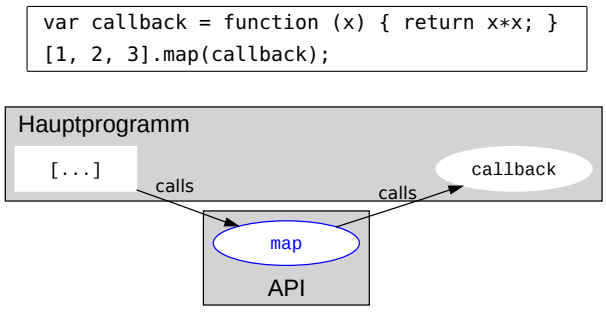


Abbildung 18.2: Inversion of Control durch eine Callbackfunktion.

[1] Fowler (2004).
[2] Fowler (2005).

Beispiel für eine Kontrollflussumkehrung ist eine serverseitig programmierte Webanwendung, wie etwa mit PHP, da die Ausführungsverantwortung beim Server liegt, die Initiierung und das Ergebnis des Kontrollflusses jedoch beim Client[3].

Das Entwurfsmuster der *Dependency Injection* oder der *automatischen Übergabe von Abhängigkeiten* nun ist eine spezielle Form der Kontrollflussumkehrung und besagt, dass eine Komponente ihre Abhängigkeiten von anderen Komponenten nicht selbst erzeugt, sondern von einer zentralen Komponente des Systems, dem „Assembler“, übergeben bekommt[4]. Dadurch können Abhängigkeiten zwischen Komponenten also ausschließlich durch die zentrale Komponente erzeugt und damit zentral aufgelöst werden. Eine Möglichkeit dafür ist die zentrale Bereitstellung einer Klasse für die Abhängigkeit sowie einer Schnittstelle, die von abhängigen Klassen implementiert werden muss, um zur Laufzeit die Abhängigkeiten zur Verfügung gestellt zu bekommen:

Listing 18.1: Einfache Implementierung einer Dependency Injection in Java

```
interface Injizierbar {
   public void injiziere(Abhängigkeit abhängigkeit);
}

class Abhängiges implements Injizierbar {
   private Abhängigkeit abhängigkeit;

   public void injiziere(Abhängigkeit abhängigkeit)  {
      this.abhängigkeit = abhängigkeit;
   }
}

public class Injizierer {
   public static void main(...) {
      Abhängigkeit abhängigkeit = ...;
      Injizierbar abhängiges = new Abhängiges(...);
      abhängiges.injiziere(abhängigkeit);
   }
}
```

Die Klasse `Injizierer` kann hier die Abhängigkeit über deren Konstruktor spezifizieren und zusätzlich aus möglicherweise mehreren injizierbaren Klassen auswählen (solange sie die Schnittstelle implementieren).

Mit der automatischen Übergabe von Abhängigkeiten weist ein Softwaresystem eine *lose Kopplung* der Komponenten auf. Lose Kopplung führt dazu, dass Änderungen in einer Komponente keine Änderungen in einer anderen Komponente bedingen, wenn die Schnittstelle unverändert bleibt. Lose Kopplung ist daher eine hinreichende Bedingung für maximale Wartbarkeit und Testbarkeit eines Systems gegebener Komplexität.

[3] Tarasiewicz und Böhm (2014):§2.1.3.
[4] Fowler (2004).

18.2 Einführung in Angular

18.2.1 Entwicklungsumgebung für Angular

Es gibt zwei Möglichkeiten, Angular-Anwendungen zu installieren. Die erste ist die Verwendung der Plattform `https://stackblitz.com`, die man mit einem GitHub-Account verknüpfen kann und die prototypische Programmierung ermöglicht. Die zweite Möglichkeit ist die Installation der Entwicklungsplattform Angular CLI (CLI: "command line interface"). Sie ermöglicht es, die Anwendung mit der Programiersprache TypeScript zu erstellen und in eine lauffähige Website auf Basis von JavaScript zu übertragen. Benötigt werden dazu der Webserver `node.js` und das Paket `ng`. Die Installation ist auf

`https://nodejs.org`

beschrieben (*Empfohlen:* die Version LTS, "long term support"). Danach kann Angular CLI nach den Anweisungen unter

`https://angular.io/guide/setup-local`

installiert werden.

18.2.2 TypeScript

Die Entwicklung einer Angular-App basiert auf der Programmiersprache TypeScript. TypeScript umfasst JavaScript vollständig, erweitert es aber wesentlich, z. B. um folgende Programmierkonstrukte:

- statische Typisierung: `let x: number = 5; let a: string = 'Hallo!';`
- klassenbasierte objektorientierte Programmierung und Interfaces; die Attribute heißen *Properties* oder Eigenschaften und können mit `!` als obligatorisch und mit `?` als optional markiert werden, z.B.:

```
class User {
  id!: number;       // muss obligatorisch einen Wert bekommen
  name: string;
  nickname?: string; // optional

  constructor()
}
```

- *Decorators* mit `@` zum Hinzufügen von Metadaten, z.B. `@Component`.
- *Nullish Coalescing* mit `??` erlaubt die Zuweisung von Defaultwerten:

```
let bar = foo ?? 'ufo' // ist foo undefined oder null, gilt bar = 'ufo'
```

Siehe auch `https://www.typescriptlang.org/docs/`, insbesondere die Cheat Sheets `https://www.typescriptlang.org/cheatsheets`.

18.2.3 Aufbau und Ablauf einer Angular-Anwendung

Um Webanwendungen mit TypeScript zu entwickeln, verwendet man üblicherweise die Entwicklungsplattform Angular CLI (CLI: "command line interface"). Sie ermöglicht es, mit TypeScript zu entwickeln und die Programme in eine lauffähige JavaScript-Anwendung umzuwandeln (zu „transpilieren"). Benötigt werden dazu der Webserver `node.js` und das Paket `ng`. Weitere Informationen, auch zur Installation, sind zu finden unter

`https://cli.angular.io/`.

Ein Projekt mit Angular hat eine wie in Abbildung 18.3 dargestellte Dateistruktur. Hier sind die

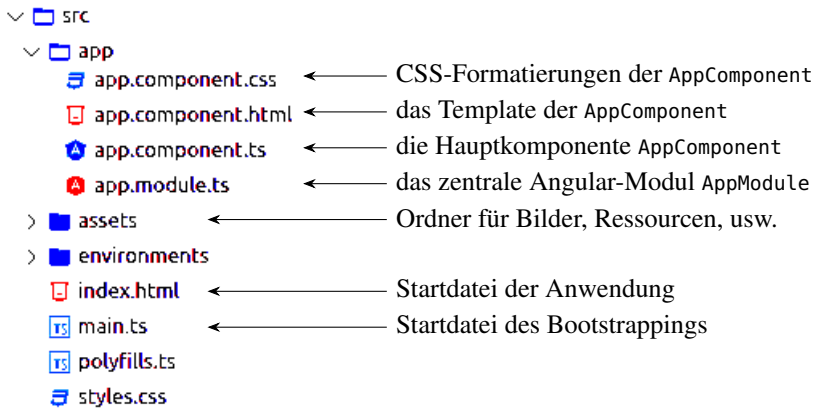


Abbildung 18.3: Dateistruktur einer Angular-Anwendung.

für den Start der Anwendung wesentlichen Dateien im Wurzelverzeichnis `src` gespeichert, die für die konkrete Darstellung und die Geschäftslogik im Verzeichnis `app`. Die HTML-Dateien

Templates

sind Templates, also Dateien, die das HTML-Grundgerüst der Seite oder Teilen davon liefert. Die Dateien mit der Endung `.ts` sind Programme in TypeScript und steuern die Daten und Abläufe der Anwendung. Die Datei `index.html` ist die Startdatei der Anwendung, die im Browser geladen wird. Sie kann verschiedene HTML-Anweisungen enthalten, trägt aber immer die folgende Besonderheit: das Element `<app-root>`, das den Anzeigebereich der Hauptkomponente

Komponente

markiert. Angular verarbeitet dieses Element und berechnet („rendert") seinen Inhalt mit der Hauptkomponente `AppComponent`. Sie enthält die eigentlichen Inhalte der Anwendung und wird in der Datei `app.component.ts` definiert; das Template `app.component.html` bestimmt die *View*, also wie die Inhalte angezeigt werden, die Klasse `app.component.ts` das *ViewModel*.

main.ts: Bootstrap

Eine wesentliche Rolle beim Start der Anwendung spielt die Datei `main.ts`. Sie startet das zentrale Modul `AppModule`, das wiederum in dem Programm `app.module.ts` im Verzeichnis `app` definiert ist und die zu importierenden Module und Klassen festlegt.

Der zeitliche Ablauf einer Angular-Anwendung ist in Abbildung 18.4 zusammengefasst. Ruft ein Browser die Anwendung auf, wird zunächst die Datei `index.html` dargestellt, während das Bootstrapping mit dem App-Modul durchgeführt wird und die Hauptkomponente AppComponent in die Index-Datei geladen wird. Diese Komponente besteht aus der Klasse `AppComponent`, implementiert in dem Programm `app.component.ts`, und ihrem Template `app.component.html`. Die eigentlichen Inhalte und ihre Darstellung werden in dieser Komponente festgelegt. Wir werden uns daher im Folgenden vor allem näher mit ihr beschäftigen.

18.2.4 Schritte zur Erstellung der ersten Angular App

Die Dateien für das Grundgerüst eines Angularprojekts werden automatisch angelegt, wenn wir über die Konsole den Befehl eingeben:

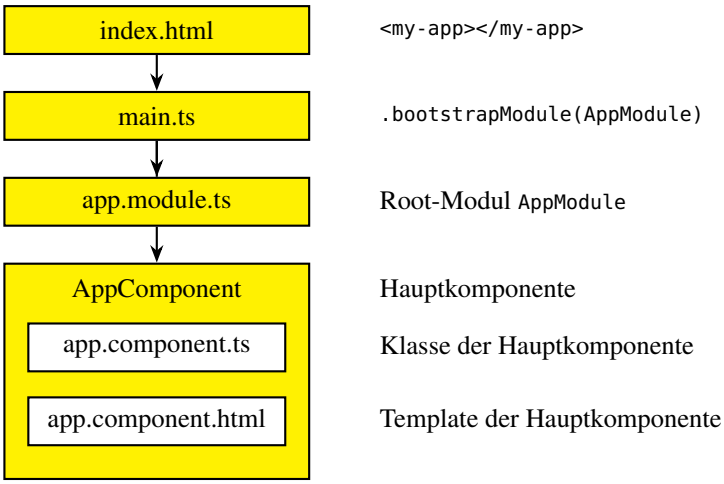


Abbildung 18.4: Ablauf einer Angular-Anwendung. Grafik nach Malcher et al. (2020:S. 6).

```
ng new <mein-projekt>
```

und danach mit `cd <mein-projekt>` in das Projektverzeichnis wechseln. (Hier stehen die spitzen Klammern nur für Platzhalter!). Wollen wir unser erstes Projekt also `erste-app` nennen, sollten wir über die Konsole einen Ordner `angular` erstellen, in dem wir auch alle künftigen Apps speichern:

Schritt 0: `mkdir angular` (Dieser Schritt ist künftig natürlich nicht mehr nötig.) Danach wechseln wir in dieses Verzeichnis:

Schritt 1: `cd angular`

und erstellen das eigentliche Projekt:

Schritt 2: `ng new erste-app`

Danach müssen wir mit zwei Prompts die wesentlichen Konfigurationseinstellungen des Projekts festlegen, wir bestätigen für unser erstes Projekt alle Defaultangaben mit Return.

Schritt 3: Import wichtiger Module. Da wir fast immer mit Formulardaten arbeiten werden, sollte als nächstes in unserer Hauptdatei `app.module.ts` das Modul `FormsModule` eingebunden werden:

```
// ...
import { FormsModule } from '@angular/forms';

@NgModule({
  // ...
  imports: [
  // ...
  FormsModule,
  ],
})
export class AppModule {}
```

Schritt 4: Löschen des Standardtexts in dem Template der Appkomponente `app.component.html`. Stattdessen Eingabe eines eigenen HTML-Textes, z. B. `<h1>Hallo Angular!</h1>`

Schritt 5: Wechsel in das Projektverzeichnis mit `cd erste-app` und Starten des Node-Servers mit `ng serve`.

Schritt 6: Aufruf der in der Konsole angegebenen URL im Browser: `http://localhost:4200`

Letzter Schritt: Ist das Projekt fertig entwickelt, kann es mit `ng build --base-href ./` als App in JavaScript transpiliert werden. Die Dateien dazu werden dann in dem Unterverzeichnis `dist` erstellt und können auf einen beliebigen Webserver deployed werden.

Anmerkung. *Kopieren wir die AppComponent aus Stackblitz in unser Projekt, so muss der Selektor darin nach* `'app-root'` *statt* `'my-app'` *geändert werden:*

```
// ...
@Component({
  selector: 'app-root',
  // ...
})
```

Dieser Selektor verweist auf der Anzeigenbereich der Hauptkomponente in der Startdatei `index.html`.

18.2.5 Komponenten und Templates

Komponenten als selbstdefinierte HTML-Tags

Komponenten sind die Grundbausteine einer Angular-Anwendung. Sie sind angelehnt an die *Web-Components*, das sind selbst definierbare HTML-Tags. In Angular ist jede Anwendung aus Komponenten zusammengesetzt, die jeweils eine bestimmte Aufgabe erfüllen und der Hauptkomponenten *(Root Component)* der Anwendung untergeordnet sind. Eine Komponente beschreibt immer nur einen kleinen Teil der Anwendung, z.B. eine Seite oder ein einzelnes HTML-Element. Sie sollte stets mit dem Konsolenbefehl

`ng generate component <komponente>` oder `ng g component <komponente>`

erzeugt werden, so dass alle notwendigen Dateien und Einträge automatisch erstellt werden. Das Grundgerüst einer Komponente sieht wie folgt aus: Eine Komponente besteht aus einer Klasse in TypeScript und wird mit einem Template verknüpft, der *View* der Komponente. Die Klasse wird immer mit einem *Decorator* `@Component` eingeleitet, der diese Verknüpfung über ien JSON-Objekt als Metainformation herstellt:

```
@Component({
  selector: 'my-component',
  templateUrl: './my.component.html'
})
export class MyComponent { ... }
```

Die dem Decorator übergebenen Metadaten enthalten also das HTML-Element als `selector`, das *Host-Element*, in dem die Daten der Klasse verfügbar sind und dargestellt werden können, sowie das Template, in dem es sich befindet. Das Template kann als externe Datei per `templateUrl` mit seinem Pfad verknüpft werden, aber auch mit `template` direkt („inline“) als String. Als Konvention wird das HTML-Element von dem Namen der Komponente (in CamelCase-Notation)

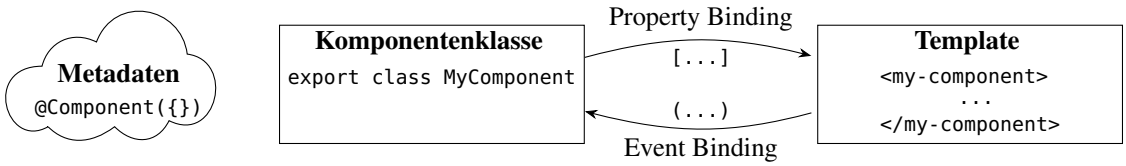


Abbildung 18.5: Bestandteile einer Komponente in Angular. Grafik nach Malcher et al. (2020:S. 77).

app.component.ts

```
import {Component} from '@angular/core';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})

export class AppComponent {
  eingabe : string;
}
```

app.component.html

```
<input [(ngModel)]="eingabe"
   type="text" placeholder="Name"/>
<p>Hallo {{ eingabe }}!</p>
```

Abbildung 18.6: Datenbindung in Angular: Links die Komponente, rechts das Template

abgeleitet, indem jeder Großbuchstabe im Innern durch einen Bindestrich und den entsprechenden Kleinbuchstaben ersetzt wird („dashed-case“). Eine Komponente `"adressenListe"` beispielsweise wird mit den Tags

```
<adressen-liste> ... </adressen-liste>
```

verwendet. Eine Komponente muss in Angular in dem zentralen `AppModule` registriert werden. Dazu dient die Eigenschaft `decoration` im Decorator `@NgModule()`. Mit dem Konsolenbefehl `ng g component ...` wird diese Registrierung automatisch ausgeführt.

Komponente im AppModule registrieren

Die Klasse der Komponente stellt Attribute als *Properties* zur Verfügung, deren Werte in dem Template angezeigt werden können. Dazu kann die Direktive `ngModel` in eckigen Klammern verwendet werden, also `[ngModel]`. Wird umgekehrt der Wert eines `<input>`-Elements in dem Template verändert und soll an die Properties der Komponente gebunden werden, so verwenden wir sie in runden Klammern, also `(ngModel)`. Wollen wir eine *Zwei-Wege-Bindung (two-way data binding)*, so müssen wir beide Klammern verwenden, also `[(ngModel)]`. Voraussetzung für diesen Mechanismus ist, dass im Hauptmodul Da wir fast immer mit Formulardaten arbeiten werden, sollte als nächstes in unserer Hauptdatei `app.module.ts` das Modul `FormsModule` eingebunden ist (s.o.).

Datenbindung mit ngModel

18.2.6 Erste Beispiele: Datenbindung

Genug der Theorie, wie kann man denn nun mit Angular programmieren? In diesem Abschnitt werden wir einführende Programmbeispiele vorstellen, die die Datenbindung verwenden, aber auch ein Schleifenkonstrukt und eine (Art) Selektionsstruktur kennenlernen. Betrachten wir zunächst ein Programm mit Angular, das das Prinzip der Datenbindung (*data binding*) illustriert. Mit der Direktive `[(ng-model)]` wird in dem Template die Verbindung zu einer oder den

Variablen in der zugehörigen Komponente festgelegt. Hier ist es die Variable `eingabe`, die den Wert des `<input>`-Elements erhält. In einer „Interpolation“ `{{...}}` mit dem *Template-Ausdruck* `eingabe`, also `{{ eingabe }}`, wird auf den aktuelle Wert der Variable zugegriffen. Hier wird er eingepackt in ein `<p>`-Element ausgegeben.

Was geschieht in diesem Programm? Gibt der Anwender in dem Eingabefeld einen Text ein, so wird nach jedem Tastendruck der aktualisierte Wert der Variable angezeigt. Man hat also in dem HTML-Dokument zu jedem Zeitpunkt in zwei verschiedenen Elementen denselben Wert stehen, es handelt sich um eine (hier noch einfache) *Datenbindung*.

Die Datenbindung ist eine mächtige Technik und ermöglicht mit relativ wenig Quelltext bemerkenswerte Effekte. So wird im nächsten Beispielprogramm ein Farbenwähler erstellt, der durch Schiebereglern festgestellte Farben nach dem RGBA darstellt.[5] Zunächst werden mit der Komponente vier numerische Variablen `r`, `g`, `b` und `a` deklariert und mit Startwerten initialisiert:

```
import { Component } from '@angular/core';
@Component({ ... })

export class AppComponent {
  r : number = 255;
  g : number = 0;
  b : number = 123;
  a : number = 0.7;
}
```

In em zugehörigen Template befinden sich vier Slider-Inputs (`type="range"`), deren Werte mit den Attributvariablen der Komponenten über `[(ngModel)]` doppelt gebunden sind:

```
<div>
R: <input type="range" min="0" max="255" step="1" [(ngModel)]="r"/><br/>
G: <input type="range" min="0" max="255" step="1" [(ngModel)]="g"/><br/>
B: <input type="range" min="0" max="255" step="1" [(ngModel)]="b"/><br/>
A: <input type="range" min="0" max="1" step="0.01" [(ngModel)]="a"/>
</div>

<div style="width: 300px; height: 100px;
     background-color: rgba({{r}},{{g}},{{b}},{{a}});">
</div>
```

Die mit den Slidern eingestellte Farbe wird in dem zweiten `<div>`-Element als Fläche dargestellt.

18.2.7 Direktiven: Schleifen und bedingte Anweisungen

In den nächsten Beispielprogramm dieses Abschnitts werden wir über Standarddirektiven in Angular Schleifen und bedingte Anweisungen kennenlernen. Es gibt drei Arten von Direktiven in Angular:[6]

1. *Komponenten* sind Direktiven mit einem Template; sie sind die Grundbausteine einer Angular-Anwendung. (Wir haben sie schon oben in 18.2.5 kennengelernt.)

[5] RGBA erweitert das RGB-Farbmodell mit den Werten für Rot, Grün und Blau jeweils ganzzahlig zwischen 0 und 255 um den Alphawert α zwischen 0 und 1, der die Transparenz der Farbe festlegt (0: transparent, also unsichtbar, 1: nicht transparent, also opak).

[6] `https://angular.io/guide/built-in-directives`

2. *Attributdirektiven* sind Direktiven, die das Verhalten eines Elements, einer Komponente oder einer anderen Direktive ändern. Wichtigstes Beispiel für eine Attributdirektive ist `ngModel`, das zur Datenbindung verwendet. Für das Event-Binding wird eine Attributdirektive in runde Klammern gesetzt, für das Property-Binding in eckigen Klammern, und für die Zwei-Wege-Datenbindung in eckige und runde Klammern. Daneben gibt es noch `ngClass` zum Hinzufügen oder Entfernen von CSS-Klassen und `ngStyle` zum Hinzufügen oder Entfernen von `style`-Attributen in HTML.

3. *Strukturdirektiven* fügen DOM-Elemente hinzu oder entfernen sie. Im Standard vorhandene Strukturdirektiven sind `*ngFor` für wiederholte DOM-Elemente, `*ngIf` zur bedingten Darstellung von Elementen eines Templates und `*ngSwitch` zum Selektieren aus einer Menge von optionalen Elementen.

Im nächsten Programm wird über die Strukturdirektive `*ngFor` eine Schleife zum Durchlaufen eines Arrays erzeugt, die alle Einträge ausgibt, sowie eine zweite Schleife, die bestimmte Elemente des Arrays filtert, also selektiert. Aber zunächst das Programm:

`app.component.ts`:

```
import {Component} from '@angular/core';
@Component({ ... })

export class AppComponent {
  woerter = ["Da", "steh", "ich", "nun",
     "ich", "armer","Tor"
  ];
}
```

`app.component.html`:

```
<p>
  <span *ngFor="let wort of woerter">
    {{ wort }}
  </span>
</p>
```

Es gibt die einzelnen Einträge des Arrays `woerter` aus, das in der Komponente initialisiert wurde. Wir können in der Anwendung durch die Direktive `*ngIf` auch filtern. Benötigen wir ferner den Index der Einträge, so geschieht dies mit dem Zusatz `index as ...` in der *ngFor-Direktive in dem Template:

```
<p>
<span *ngFor="let wort of woerter; index as i">
  <span *ngIf="wort == 'ich'">{{ i }}: {{ wort }}, </span>
</span>
</p>
```

Die Ausgabe ist hier: `2: ich, 4: ich`. Wichtig ist, dass die *ngIf-Direktive in einer inneren Element ist, hier ein weiteres `<span>`-Element, da die Variable `wort` nur innerhalb der Komponente mit der *ngFor-Direktive bekannt ist.

Wie die `*ngIf`-Direktive mit einem else-Zweig versehen werden kann, betrachten wir in dem folgenden Programm. Hier werden drei Radio-Buttons (rot, grün, blau) gezeigt und ein Text, der „Rot“ ausgibt, wenn der entsprechende Button angeklickt ist, und „Nicht rot“ sonst. Es handelt sich also um eine if-else-Konstruktion. Zunächst die Komponente:

```
import {Component} from '@angular/core';
@Component({ ... })

export class AppComponent {
  farbe : string = "blau";
}
```

Und hier das Template:

```
<div>
  <input type="radio" name="farbe" [(ngModel)]="farbe" value="rot" /> rot
  <input type="radio" name="farbe" [(ngModel)]="farbe" value="gruen" /> grün
  <input type="radio" name="farbe" [(ngModel)]="farbe" value="blau" /> blau
</div>
<div *ngIf="farbe == 'rot'; then rot; else nichtRot"></div>
<ng-template #rot>Rot</ng-template>
<ng-template #nichtRot>Nicht rot.</ng-template>
```

Hierbei wird der Initialwert „blau“ durch die Klasse der Komponente festgelegt. In der *ngIf-Direktive wird nach der Bedingung und einem Semikolon mit `rot` hinter `then` das ng-Template `#rot` referenziert, das bei Anklicken von „rot“ angezeigt wird. Hinter `else` wird entsprechend das ng-Template `#nichtRot` referenziert, das bei Anklicken einer anderen Farbe erscheint. Wichtig ist, dass in dem gesamten Template die Referenzen der `ng-Templates` eindeutig sind.

18.2.8 Eine Komponente hinzufügen

Betrachten wir als letztes Beispielprogramm eine Erweiterung unseres ersten Beipielprogramms um eine Komponente, die die pythagoreische Hypothenusengleichung

$$z = \sqrt{x^2 + y^2} \tag{18.1}$$

berechnet. Erweitern wir unsere Anweundung mit der Anweisung

```
ng generate component formel
```

in Angular CLI, ausgehend von dem Hauptverzeichnis, so wird ein Unterverzeichnis `formel` mit den Dateien `formel.component.ts`, `formel.component.html`, `formel.component.css` und `formel.component.spec.ts` angelegt, siehe Abbildung 18.7 links. (Die Datei `formel.component.spec.ts` ist hier nicht abgebildet, sie ist erst für das Testmanagement erforderlich.) Die schon

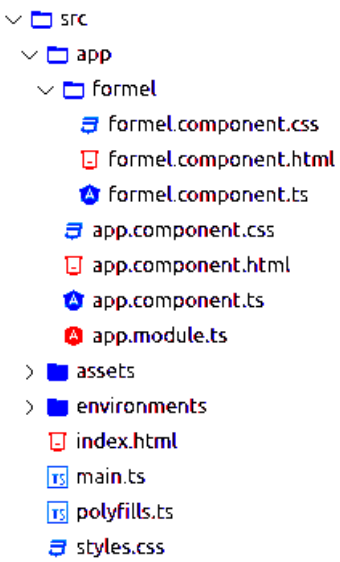


Abbildung 18.7: Dateistruktur mit der Komponente `formel`.

vorher vorhandenen Dateien im Verzeichnis `app` bleiben unverändert. Die Datei `formel.component.css` lautet:

```
p {text-align:center;}
```

Sie legt fest, dass Inhalte eines `<formel>`-Elements als zentrierter Absatz `<p>` formatiert wird. Die Komponente `formel.component.ts` lautet:

```
import { Component } from '@angular/core';
@Component({
  selector: 'formel',
  templateUrl: './formel.component.html',
  styleUrls: ['../app.component.css', './formel.component.css'],
})
export class FormelComponent {
  x = 3;
  y = 4;

  /** Funktion als Property */
  berechne = function (a: number, b: number): number {
    return Math.sqrt(a ** 2 + b ** 2);
  };
}
```

Es stellt mit `selector` die Verbindung zum Element `<formel>` her, mit `templateUrl` zum Template, und mit `styleUrls` die Formatierung der Hauptkomponente und den spezifischen Formatierungen in der Datei `formel.component.css`. Ferner stellt sie drei Properties bereit, die Attribute `x` und `y` sowie die Funktion `berechne`, die für zwei einzugebende Zahlen a und b die Hypothenusengleichung (18.1), also $\sqrt{a^2 + b^2}$ implementiert. Das Template `formel.component.html` schließlich lautet:

```
<div>
x = <input size="1" [(ngModel)]="x" />,
y = <input size="1" [(ngModel)]="y" />:
</div>
<p>
  {{ x }}<sup>2</sup> + {{ y }}<sup>2</sup> = {{ berechne(x,y) }}<sup>2</sup>
</p>
```

Es ermöglicht die Eingabe zweier Zahlen x und y, die sie dann mittels Zwei-Wege-Datenbindung über `[(ngModel)]`in die Funktion `berechne` der Komponente einsetzt und das Ergebnis z als Formel $x^2 + y^2 = z^2$ ausgibt.

18.2.9 Wichtige Grundelemente von Angular

Zunächst sei als Standardreferenz die Dokumentation der API von Angular erwähnt:

`https://angular.io/api`

Angular basiert auf den folgenden Grundelementen, siehe Abbildung 18.8. Das übergreifende Strukturelement in Angular bilden *Module*. Sie kapseln zusammenhängende Anwendungsbestandteile und dienen als Container für Komponenten, Direktiven, Services, Routings und Pipes (Abbildung 18.8). Siehe auch `https://angular.io/guide/architecture`. Komponenten und Direktiven haben wir bereits kennen gelernt. Services stellen Funktionen bereit, so dass sie von mehreren Komponenten und Klassen aus aufgerufen werden können. Dies funktioniert mit Dependency Injection, indem der Konstruktor der Klasse den Dienst als Parameter übergeben bekommt.

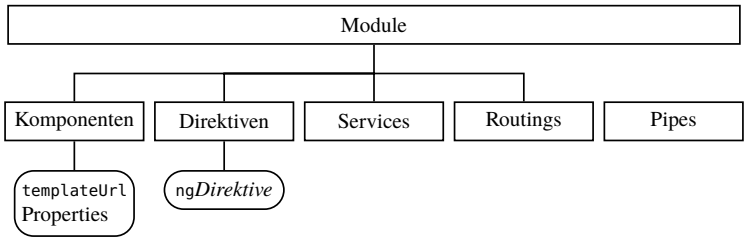


Abbildung 18.8: Struktur der Grundelemente von Angular.

Routings steuern und verwalten das Laden von Komponenten innerhalb einer Single-Page-Anwendung. Wir werden es weiter unten in Abschnitt 18.4 behandeln.

Pipes dienen zur Transformation von Strings, Währungsbeträgen oder Datumsangaben in der Anzeige: `https://angular.io/guide/pipes`

18.3 Erste Datenbankzugriffe mit Angular

An Programmbeispielen haben wir bislang Angular lediglich im Bereich des Clients gesehen, also der View und der ModelView mit der Zwei-Wege-Datenbindung. Ohne Zweifel ist das eine der Stärken von Angular und ermöglicht beeindruckende Effekte. Aber den serverseitigen Anteil, also die Datensynchronisation mit dem Model in Abbildung 18.1 (links) haben wir noch nicht betrachtet.

Wir werden jedoch gleich sehen, dass dies eine weitere Stärke des Ansatzes von Angular ist. Um das Prinzip dieses clientseitigen Ansatzes für den Zugriff auf zentrale Daten zu verstehen, betrachten wir in diesem Abschnitt eine einfache Webapplikation, die die Einträge der Adressdatenbank aus Abschnitt 12.2 aus Webtechnologie 1 in einer Angular-Applikation darstellt. Daraus werden wir erste wesentliche Schritte zur Entwicklung einer Anwendung mit Angular ableiten.

18.3.1 Die API des Webservers: das Modell

Um eine Webapplikation zu implementieren, die die Einträge der Adressdatenbank aus Abschnitt 12.2 aus Webtechnologie 1 in einer Angular-Applikation darstellt, muss zuerst die Schnittstelle zwischen Webserverver und dem Client spezifiziert werden. Standardmäßig wird dazu das JSON-Format verwendet. Die bereitgestellten Fuknktionen werden in der Webentwicklung oft als Web API des Servers bezeichnet.

In unserem ersten Beispiel sei die Web API durch das PHP-Skript in Listing 18.2 gegeben, das unter dem Namen `adress-db.php` auf dem Server gespeichert sei.

Listing 18.2: Das serverseitige PHP-Skript, das in Listing 18.3 aufgerufen wird.

```php
<?php
header("Access-Control-Allow-Origin: *"); //für JS-Zugriffe von anderen Hosts
require_once("db_login.php");

if (isset($_GET['action']) && $_GET['action'] === "list") {
   $mysqli = login("AdressDB");
   $result = $mysqli->query("SELECT * FROM adressen");

   $array = array();
   while ($adresse = $result->fetch_assoc()) {
      $array[] = $adresse;
```

```
    }
    echo json_encode($array);
}
?>
```

Wird es per GET mit dem Parameter `action=list` aufgerufen, so gibt es die Ergebnismenge eines SELECTs als JSON-Array zurück.

18.3.2 Implementierung mit Angular

Um den Aufruf des PHP-Skripts aus dem vorherigen Abschnitts im Browser über Angular auszuführen, müssen wir zunächst ein eigenes Projekt erzeugen, nennen wir es `datenbank`:

```
ng new datenbank
```

Danach wechseln wir in das Verzeichnis mit

```
cd datenbank
```

Zunächst erstellen wir das Modell unserer Anwendung. Da wir nur eine einzige Tabelle `adressen` in unserer Datenbank haben, benötigen wir auch nur eine Entität `adresse`. Wir erzeugen ihre TypeScript-Datei zunächst in einem Unterverzeichnis `model` mit

```
ng generate interface model/adresse
```

und ergänzen sie mit den Spaltennamen der Datenbanktabelle als Properties:

```
export interface Adresse {
  id: number;
  vorname: string;
  name: string;
  email: string;
  web: string;
  strasse: string;
  plz: string;
  ort: string;
}
```

Wichtig ist, dass Spalten- und Property-Namen exakt gleich sind – insbesondere in Groß- und Kleinschreibung – und die Datentypen kompatibel sind.

Als nächstes erzeugen wir den Service `datenbank` für die Zugriffe auf die Datenbank mit

```
ng generate service datenbank
```

Es entsteht die Datei `datenbank.ts` im App-Verzeichnis, die wir mit unseren spezifischen Anforderungen ergänzen:

Listing 18.3: Der Service, der den Aufruf des PHP-Skripts in Listing 18.2 bereit stellt.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

import { Adresse } from './model/adresse';

@Injectable({ providedIn: 'root' })
export class DatenbankService {
```

```
  constructor(private http: HttpClient) {}

  getAdressen(): Observable<Adresse[]> {
    const params = new HttpParams({
      fromObject: {action: 'list'}
    });
    return this.http.get<Adresse[]>(
      "http://haegar.fh-swf.de/Webtechnologie/angular/datenbank/address-db.php",
      { params }
    );
  }
}
```

Hier werden die Klassen `HttpClient` und `Observable` importiert, mit denen wir AJAX-Zugriffe auf den Server programmieren können. Dazu müssen wir aber vorher in unserer Hauptdatei `app.module.ts` das Modul `HttpClientModule` einbinden:

```
// ...
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  // ...
  imports: [
  // ...
  FormsModule,
  HttpClientModule,
  ],
})
export class AppModule {}
```

In dem Datenbankservice selber haben wir eine Methode `getAdressen()` implementiert, die den AJAX-Zugriff mit dem URL spezifiziert. Die Funktion gibt eine *Observable*[7] zurück, eine asynchron aufzurufende Funktion, die erst mit `subscribe` aufgerufen wird. Will man Eingabedaten verarbeiten, muss entsprechend das Modul `FormsModule` importiert werden.

Am Schluss erzeugen wir eine neue Komponente `adress-liste`:

```
ng generate component adress-liste
```

Sie soll die vom Server erhaltene Adressliste anzeigen. Dazu importieren wir in der Klasse `adress-liste.component.ts` unser Modell `adresse` und den DatenbankService, übergeben dem Konstruktur den Service und abonnieren den AJAX-Zugriff mit `subscribe`:

```
import { Component, OnInit } from '@angular/core';

import { Adresse } from '../model/adresse';
import { DatenbankService } from '../datenbank.service';

@Component({
  selector: 'app-adress-liste',
  templateUrl: './adress-liste.component.html',
```

[7] `https://angular.io/guide/observables`

```
  styleUrls: ['./adress-liste.component.css']
})
export class AdressListeComponent implements OnInit {
  adressen: Adresse[] = [];

  constructor(private dbs: DatenbankService) {}

  ngOnInit(): void {
    this.dbs.getAdressen().subscribe(res => this.adressen = res);
  }
}
```

Die Property `adressen` ist ein Array von Adressen und kann als Attribut der Komponente `<app-adress-liste>` eingebunden werden. Das Template soll dieses Array durchlaufen und jeden Eintrag in einer geordneten Liste `<ol>` anzeigen, was in der Datei `adress-liste.component.html` wie folgt geschieht:

```
<ul class="adressen">
  <li *ngFor="let adresse of adressen">{{adresse.id}}: {{adresse.vorname}} {{
      adresse.name}}
  </li>
</ul>
```

Um schließlich die Liste auch zu sehen, müssen wir die Komponente in das Template `app.component.html` der Hauptkomponente einbinden:

```
<h2>Meine Adressliste</h2>
<app-adress-liste></app-adress-liste>
```

Mit der Anweisung `ng generate component ...` wurde übrigens die erzeugte Komponente automatisch im Hauptmodul `app.module.ts` registriert.

Wenn wir es nicht bereits vorher gemacht haben, können wir nun unsere Anwendung mit `ng serve -open` starten. Ist das Projekt schließlich fertig entwickelt und lauffähig, so kann es mit dem Befehl

```
ng build --base-href ./
```

als App in JavaScript transpiliert werden. Die Dateien dazu werden damit in dem Unterverzeichnis `dist` erstellt und können auf einen beliebigen Webserver kopiert („deployed“) werden.

18.3.3 Zusammenfassung: Erstellung einer Angular-App

Fassen wir die Schritte zur Erstellung einer Angular-App zusammen, wie wir sie in diesem Abschnitt erkannt haben.

1. Erstellung der serverseitigen Schnittstelle für Queries als Web API zur zentralen Datenbank, z.B. über ein PHP-Skript.
2. Erzeugung eines Angular-Projekts mit `ng new ...`
3. Import der Module `FormsModule` und `HttpClientModule` in der Hauptdatei `app.module.ts`.
4. Spiegelung der Datenbanktabellen als Interfaces mit `ng generate interface ...`. Hier müssen die Properties der Interfaces den Spaltennamen der Tabellen exakt entsprechen und ihre Datentypen kompatibel sein.

5. Erzeugung von Services mit `ng generate service ...`, die die wesentlichen Funktionen der Anwendung ermöglichen, insbesondere die Zugriffe auf die Serverprogramme. Dazu muss das Modul `HttpClientModule` im Hauptmodul `app.module.ts` und die Module `HttpClient` und `Observable` im Service importiert werden. `HttpClient` stellt AJAX-Zugriffe auf die Datenbank als Observablen bereit, die mit `subscribe` abonniert werden müssen, damit sie tatsächlich ausgeführt werden.

6. Erzeugen der Komponenten mit `ng generate component ...`, die die implementierten Funktionen einbinden und die Ergebnisse im Browser darstellen. Mit diesem Konsolenbefehl werden nicht nur die notwendigen Dateien erzeugt, sondern die Komponente auch im Hauptmodul `app-module.ts` registriert.

7. Ist das Projekt fertig entwickelt und lauffähig, so kann es mit dem Befehl

   ```
   ng build --base-href ./
   ```

   als App in JavaScript erstellt werden. Die Dateien dazu werden dann in dem Unterverzeichnis `dist` erstellt und können auf einen beliebigen Webserver deployed werden.

18.4 Single Page App mit Datenbankzugriffen

In diesem Abschnitt erstellen wir eine kleine Single-Page App (SPA). Eine solche „Einzelseitenanwendung“ ist eine Webanwendung, die aus einem einzigen HTML-Dokument besteht, dessen Inhalte dynamisch über AJAX nachgeladen werden, ohne dass die Seite neu geladen wird, wie schon kurz in Abschnitt 18.1.1 beschrieben. In Angular wird dies ermöglicht, indem eine spezielle leere Komponente als Platzhalter in der Hauptkomponente eingerichtet wird, die abhängig von dem URL-Pfad entsprechende Inhalte dort darstellen kann. Die dazu notwendige Verknüpfung von Pfaden und Inhalten liefert das *Routing*. Die Grundidee des Routings

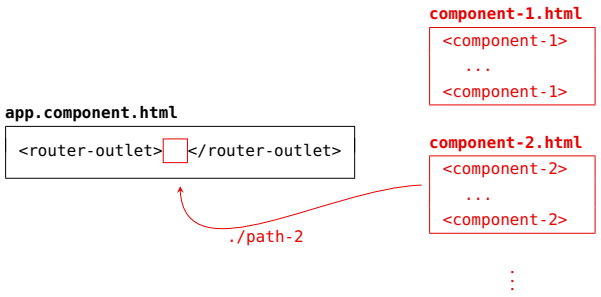


Abbildung 18.9: Grundidee des Routings in Angular: dynamisches Einsetzen von Komponenten in den Platzhalter `<router-outlet>` per Pfad.

ist folgende: Das Template `app-component.html` der Hauptkomponente enthält die Direktive `<router-outlet>`, die als Platzhalter für weitere Komponenten dient. Welche Komponente angezeigt wird, entscheidet der Pfad, der für sie konfiguriert wurde.

Die interne Verwaltung der Komponenten und ihrer Pfade übernimmt dabei das Routenmodul `app-routing.module.ts`. Es wird automatisch erstellt und eingebunden, wenn wir das Projekt mit der Anweisung

```
ng new <my-project> --routing
```

erzeugen. In dieser Datei importieren wir die zu verlinkenden Komponenten und tragen wir in das anfangs leere Array `routes` seine Routendefinitionen ein, also beipielsweise:

```
import { Component-1 } from '...';
import { Component-2 } from '...';

const routes: Routes = [
  { path: 'path-1', component: Component-1 },
  { path: 'path-2', component: Component-2 },
  ...
]
```

Diese Pfade können nun auch innerhalb des Projekts verlinkt werden. Allerdings muss dazu die Direktive `routerLink` verwendet werden, nicht das Attribut `href`! Also:

```
<a routerLink="./path-1">Link zu Pfad 1</a>
```

Routen können auch Parameter dynamisch übergeben werden. Bei der Routendefinition wird ein solcher Parameter mit einem Doppelpunkt eingeleitet:

```
  { path: 'path-x/:id', component: Component-x },
```

Die aufgerufene Komponente muss dazu im Konstruktor ein Objekt der Klasse `ActivatedRoute` übergeben werden, beispielsweise `route` genannt:

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
...
@Component({ ... })
export class ComponentX implements OnInit {
  ...
  constructor( private route: ActivatedRoute, ... ) {}

  ngOnInit(): void {
    const id = Number(this.route.snapshot.paramMap.get('id'));
  }
  ...
```

Mit dessen Eigenschaft `snapshot.paramMap` können per `get` alle übergebenen Parameterwerte abgerufen werden.[8] Es sind aber stets Strings. Wird eine Zahl als Parameterwert erwartet, so muss er mit `Number(...)` konvertiert werden.[9]

Als nächstes löschen wir (wie immer) den Inhalt des Templates `app.component.html` der Hauptkomponente und fügen den Platzhalter `<router-outlet>` für das Routing ein:

```
<h1>Projekt 2</h1>
<router-outlet></router-outlet>
```

Alles, was außerhalb des Platzhalters steht, bleibt in der Anzeige der App konstant.

18.4.1 Entwurf einer SPA mit Angular

Wie geht man vor, um eine SPA zu erstellen? Bevor die Implementierung beginnt, müssen zunächst das Modell mit der zugrundeliegenden Datenbank beschrieben und der Navigationsfluss der beabsichtigten App entworfen werden. Betrachten wir dazu beispielhaft eine SPA, die den Inhalt einer Datenbank „Hall of Fame“ von Helden und ihren Eigenschaften darstellt und

[8] vgl. Malcher et al. (2020):§8.2.6.

[9] vgl. `https://angular.io/tutorial/toh-pt5#extract-the-id-route-parameter`

verwaltet. Das Projekt ist angelehnt an das berühmte Tutorial *Tour of Heroes* des Angular-Projekts, `https://angular.io/tutorial#tour-of-heroes-app-and-tutorial`.

1. Das Modell und die Web API. Zunächst muss das Modell gegeben sein. Üblicherweise wird es durch ein Entity-Relationship-Diagramm oder ein Klassendiagramm dargestellt. Nehmen wir

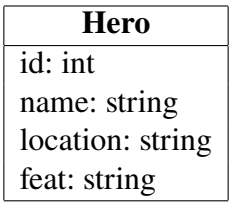


Abbildung 18.10: Das Modell der Datenbank `Hall of Fame`.

für unserr Beispielanwendung an, das Modell besteht nur aus einer Tabelle der Struktur gemäß Abbildung 18.10. Daneben muss serverseitig der Zugriff auf Datenbankfunktionen feststehen, also die API. Hier nehmen wir an, dass es ein einziges PHP-Skript `heroes.php` ist, das anhand des Requestparameters `action` die Datenbankzugriffe steuert:

```
<?php
... // Login zur Datenbank halloffame

if ($_GET['action'] == "list") {
   ... SQL "SELECT * FROM heroes" und speichern in $array
   echo json_encode($array);
} else if ($_GET['action'] == "insert") {
   $data = json_decode($_GET['data']);
   ... SQL-INSERT von $data und Rückgabe des Eintrags:
   echo $json;
} else if ($_GET['action'] == "update") {
   $data = json_decode($_GET['data']);
   ... SQL UPDATE mit $data WHERE id=$_GET[id]
} else if ($_GET['action'] == "search") {
   ... "SELECT * FROM heroes WHERE name LIKE '%$_GET[name]%'"
   ... Ergebnismenge in $array speichern
   echo json_encode($array);
} else if ($_GET['action'] == "select") {
   $sql = "SELECT * FROM $table WHERE id=$_GET[id]";
   ... Ergebnismenge in $json speichern
   echo $json;
} else if ($_GET['action'] == "delete") {
   $sql = "DELETE FROM $table WHERE id=$_GET[id]";
   $mysqli->query($sql);
} else if ($_POST['action'] == "insert") {
   $data = json_decode($_POST['data']);
   ... SQL-INSERT von $data und Ausgabe des Eintrags als JSON-Objekt:
   echo $json;
} else if ($_POST['action'] == "update") {
   $data = json_decode($_POST['data']);
   ... SQL UPDATE mit $data WHERE id=$_POST[id]
```

```
}
?>
```

2. Entwurf des Navigationsflusses der Views. Als nächstes muss der Navigationsfluss der Views entworfen werden. Hierfür muss klar sein, wieviel Views es überhaupt geben soll und wie man von dem einen zu einem anderen kommt. Üblicherweise wird ein Wechsel von Views durch das Anklicken eines Links oder eines Buttons verursacht. Daraus entsteht ein Navigationsfluss wie in Abbildung 18.11 zwischen drei Views. Für den Entwurf kann dieser Navigationsfluss

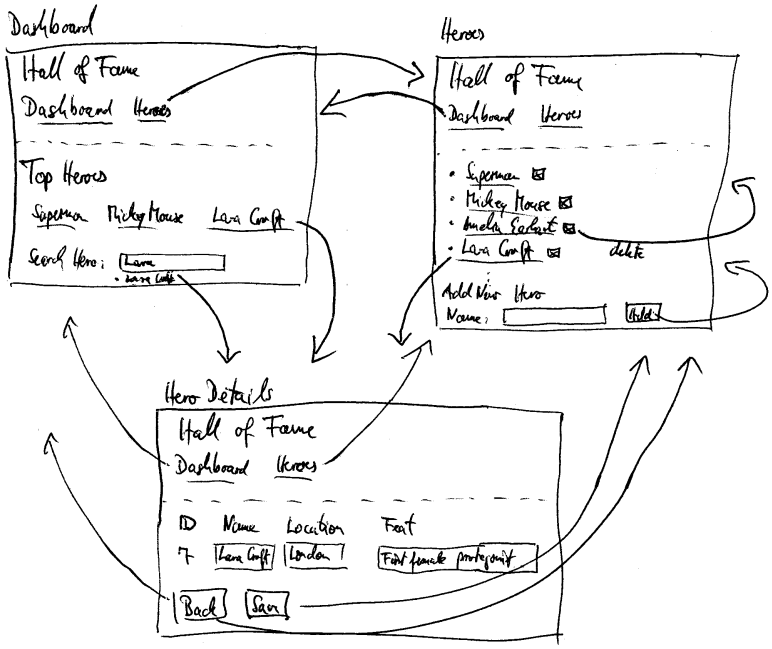


Abbildung 18.11: Der Navigationsfluss der drei Views `Dashboard`, `Heroes` und `Hero Details`.

wie in Abbildung 18.11 einfach skizziert werden. Aus ihm ergeben sich das Routing und die Linkstruktur der Anwendung. Jede View entspricht üblicherweise einer Komponente.

18.4.2 Implementierung des Programms

1. Projekterzeugung Zunächst wird ein Projekt `heroes` eingerichtet:

```
ng new heroes --routing
```

und mit `cd heroes` in das Projektverzeichnis gewechselt.

2. Erstellung des Modells. Wie in Abschnitt 18.3.3 beschrieben, wird dann zunächst das zentrale Modell mit der Anweisung

```
ng generate interface ./model/hero
```

erstellt und die Datei `hero.ts` in dem Unterverzeichnis `model` gemäß dem Modell in Abbildung 18.10 wie folgt modifiziert:

```
export interface Hero {
  id: number;
  name: string;
```

```
  location: string;
  feat: string;
}
```

3. Import des FormsModule und des HttpClientModule Um eine Web API mit Eingabedaten einzubinden, müssen die Module `FormsModule` und `HttpClientModule` in der Hauptdatei `app.module.ts` importiert werden:

```
// ...
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  // ...
  imports: [
  // ...
  FormsModule,
  HttpClientModule,
  ],
})
export class AppModule {}
```

4. Erstellung des Services. Aus der Server-API ergibt sich wiederum der Datenbankservice `hero.service.ts`, den wir mit

```
ng generate service hero
```

einrichten und danach die Datenbankzugriffe in Angular wie folgt bereitstellen:

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders, HttpParams } from '@angular/common/http';

import { Observable, of } from 'rxjs';

import { Hero } from './model/hero';

@Injectable({ providedIn: 'root' })
export class HeroService {
  // URL to web api:
  private heroesUrl='http://haegar.fh-swf.de/Webtechnologie/angular/heroes.php';

  constructor( private http: HttpClient ) { }

  /** GET heroes from the server */
  getHeroes(): Observable<Hero[]> {
    const params = new HttpParams({ fromObject: {action: 'list'} });
    return this.http.get<Hero[]>(this.heroesUrl, { params });
  }

  /** GET hero by id. Will be 404 if id not found */
```

```
  getHero(id: number): Observable<Hero> {
    const params = new HttpParams({ fromObject: {action: 'select', id: id} });
    return this.http.get<Hero>(this.heroesUrl, { params });
  }

  /* GET heroes whose name contains search term */
  searchHeroes(term: string): Observable<Hero[]> {
    if (!term.trim()) {
      // if no search term, return empty hero array.
      return of([]);
    }
    const params = new HttpParams({fromObject: {action: 'search', name: term}});
    return this.http.get<Hero[]>(this.heroesUrl, { params });
  }

  //////// Save methods //////////
  /** POST: add a new hero to the server */
  addHero(hero: Hero): Observable<Hero> {
    const params = new HttpParams({
      fromObject: {action: 'insert', data: JSON.stringify(hero)}
    });
    //return this.http.post<Hero>(this.heroesUrl,  params );
    return this.http.get<Hero>(this.heroesUrl, { params });
  }

  /** DELETE: delete the hero from the server */
  deleteHero(id: number): Observable<Hero> {
    const params = new HttpParams({ fromObject: {action: 'delete', id: id} });
    return this.http.delete<any>(this.heroesUrl, { params });
  }

  /** PUT: update the hero on the server */
  updateHero(hero: Hero): Observable<any> {
    const params = new HttpParams({ fromObject: {
        action: 'update',
        data: JSON.stringify(hero),
    }});

    //return this.http.put(this.heroesUrl,  params );
    return this.http.get(this.heroesUrl, { params });
  }
}
```

Die Methoden zum Einfügen und zum Ändern sollten eigentlich `post` und `put` heißen. Da wir hier aber auf einen fremden Server zugreifen, wird damit der Zugriff verhindert. Die Anweisungen sind daher auskommentiert. Mit `get` werden die Funktionen eigentlich nicht korrekt implementiert, aber es funktioniert hier.

5. Einrichten der Komponenten und des Routings. Aus dem Navigationsfluss (Abbildung 18.11) ergeben sich drei Komponenten: `Dashboard` zur Anzeige einzelner Einträge und einer

Suchmaske, `Heroes` zur Anzeige aller Einträge mit einem Button [x] zum Löschen des jeweiligen Eintrags sowie einer Einfügemaske für einen neuen Eintrag, und schließlich `Hero Details` zur Editierung der Daten für einen ausgewählten Helden. Die Navigationsleiste mit den Links zu `Dashboard` und `Heroes` soll dabei in allen drei Views fix sein. Erzeugen wir sie zunächst mit den Konsolenanweisungen:

```
ng generate component dashboard
ng generate component heroes
ng generate component hero-details
```

Keine eigene View, aber eine weitere Komponente ist die Suchleiste. Wir legen auch sie an:

```
ng generate component hero-search
```

Damit erhalten wir das Grundgerüst der Dateien unserer Angular App wie in Abbildung 18.12. Ferner ergeben sich daraus die notwendigen Routendefinitionen, die wir in die Datei `app-rou-`

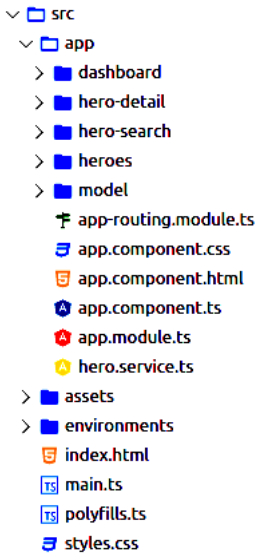


Abbildung 18.12: Das Grundgerüst der Dateien der Angular App `Heroes`.

`ting.module.ts` nach dem Import der Komponenten eintragen:

```
import { DashboardComponent } from './dashboard/dashboard.component';
import { HeroesComponent } from './heroes/heroes.component';
import { HeroDetailComponent } from './hero-detail/hero-detail.component';

const routes: Routes = [
  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },
  { path: 'dashboard', component: DashboardComponent },
  { path: 'detail/:id', component: HeroDetailComponent },
  { path: 'heroes', component: HeroesComponent }
];
```

Mit der ersten Definition wird die Dashboard View zur Startview der Anwendung. Die Route zur Detailkomponente erwartet die Übergabe der ID des betreffenden Heroes. Da die Suchkomponente Teil des Dashboards sein soll, gibt es keine eigene Route dafür.

Die Komponente Dashboard nun setzt sich aus der folgenden Klasse und dem entsprechenden Template im Unterverzeichnis `dashboard` zusammen:

dashboard.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { Hero } from '../model/hero';
import { HeroService } from '../hero.service';

@Component({
  selector: 'app-dashboard',
  templateUrl: './dashboard.component.html',
  styleUrls: [ './dashboard.component.css' ]
})
export class DashboardComponent implements OnInit {
  heroes: Hero[] = [];

  constructor(private heroService: HeroService) { }

  ngOnInit(): void {
    this.getHeroes();
  }

  getHeroes(): void {
    this.heroService.getHeroes()
      .subscribe(heroes => this.heroes = heroes.slice
          (0, 5));
  }
}
```

dashboard.component.html:

```
<h2>Top Heroes</h2>
<div class="heroes-menu">
  <a *ngFor="let hero of heroes"
      routerLink="/detail/{{hero.id}}">
      {{hero.name}},
  </a>
</div>
<br />
<app-hero-search></app-hero-search>
```

In der View werden mit einer foreach-Schleife die Einträge des Arrays `heroes` angezeigt, dass aber in der Methode `getHeroes` mit dem Array-Befehl `slice(0,5)` nur die Einträge 0 bis 4 aus der Datenbank speichert. Die 5 Einträge werden mit der Komponente `HeroDetail` verlinkt. Beachten wir hier, dass dieser interne Link nicht mit `href` erstellt ist, sondern mit `routerLink`! Der Link ruft die Komponente HeroDetail mit der ID des jeweiligen Hero-Objekts auf.

Die Klasse und das Template der Komponente `HeroDetail` in dem Unterverzeichnis lauten entsprechend:

hero-detail.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute } from '@angular/router';
import { Location } from '@angular/common';

import { Hero } from '../model/hero';
import { HeroService } from '../hero.service';

@Component({
  selector: 'app-hero-detail',
  templateUrl: './hero-detail.component.html',
  styleUrls: [ './hero-detail.component.css' ]
})
export class HeroDetailComponent implements OnInit {
  hero: Hero | undefined;

  constructor(
    private route: ActivatedRoute,
    private heroService: HeroService,
    private location: Location
  ) {}

  ngOnInit(): void {
    this.getHero();
  }

  getHero(): void {
    const id = Number(this.route.snapshot.paramMap.get(
        'id'));
    this.heroService.getHero(id)
      .subscribe(dokument => this.hero = hero);
  }

  goBack(hero? : Hero): void {
    if (hero != undefined) this.dukument = dokument;
    this.location.back();
  }

  save(): void {
    if (this.hero) {
      this.heroService.updateHero(this.hero)
        .subscribe(hero => this.goBack(hero));
    }
  }
}
```

hero-detail.component.html:

```
<div *ngIf="hero">
 <h2>Details of {{hero.name}}</h2>

 <table>
  <tr><th>ID</th><th>Name</th><th>Location<
      /th><th>Feat</th></tr>
  <tr>
   <td>{{hero.id}}</td>
   <td>
     <input id="hero-name" [(ngModel)]="
         hero.name" />
   </td>
   <td>
    <input id="hero-location" [(ngModel)]="
        hero.location" />
   </td>
   <td>
    <textarea id="hero-feat" [(ngModel)]="
        hero.feat" rows="1" cols="50"></
        textarea>
    </td>
  </tr>
 </table>
 <br/>
 <div>
  <button type="button" (click)="goBack()">
      Back</button>
  
  <button type="button" (click)="save()">
      Save</button>
 </div>
</div>
```

Für den Back-Button ist die Methode `back` des Services `Location` notwendig, der importiert werden muss.

Die Klasse und das Template der Komponente Heroes ist durch die folgenden Dateien im Unterverzeichnis `heroes` gegeben:

heroes.component.ts:

```
import { Component, OnInit } from '@angular/core';

import { Hero } from '../model/hero';
import { HeroService } from '../hero.service';

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
  heroes: Hero[] = [];

  constructor(private heroService: HeroService) { }

  ngOnInit(): void {
    this.getHeroes();
  }

  getHeroes(): void {
    this.heroService.getHeroes()
    .subscribe(heroes => this.heroes = heroes);
  }

  add(name: string): void {
    name = name.trim();
    if (!name) { return; }
    this.heroService.addHero({ name } as Hero)
      .subscribe(hero => this.heroes.push(hero));
  }

  delete(hero: Hero): void {
    this.heroes = this.heroes.filter(h => h !== hero);
    this.heroService.deleteHero(hero.id).subscribe();
  }
}
```

heroes.component.html:

```
<h2>The Heroes</h2>

<ul class="heroes">
  <li *ngFor="let hero of heroes">
    <a routerLink="/detail/{{hero.id}}">
      {{hero.name}}
    </a>
     
    <button type="button" title="delete
        hero"
      (click)="delete(hero)">x</button>
  </li>
</ul>

<h2>Add New Hero</h2>
<div>
<label for="new-hero">Hero name: </label>
<input id="new-hero" #heroName />

<!-- (click) passes input value to add()
     and then clears the input -->
<button type="button" (click)="add(
    heroName.value); heroName.value=''">
  Add new hero
</button>
</div>
```

Die Methode `getHeroes()` abonniert die gleichnamige Funktion des Hero-Services und speichert die empfangene Liste in seiner Property `heroes`, einem Array. Die Methode `add` lässt den übergebenen Datensatz in die Datenbank speichern und fügt ihn dann mit der Array-Funktion `push` zur Property `heroes` hinzu. In TypeScript wird hierbei mit **`as`** der String `name` zu einem Objekt der Klasse `Hero` gecastet (d.h., umgeformt). (Hier klappt es, da die erste String-Property der Klasse Hero sein Name ist.)

18.4.3 Die Suchkomponente und asynchrone Aufrufe aus Input-Tags

Wir haben oben bereits die Komponente HeroSearch angelegt. Sie zeigt aber noch nichts an. Sie soll ein Textfeld als Input-Tag zeigen und in schneller Abfolge (z.B. 300 ms, das ist die übliche menschliche Reaktionszeit[10]). Da die Implementierung etwas komplexer, aber dafür nicht weniger interessant ist, behandeln wir die Komponente in einem eigenen Abschnitt. Für Details sei auf `https://angular.io/tutorial/toh-pt6#create-herosearchcomponent`. verwiesen.

Die Klasse und das Template der Komponente Heroes ist durch die folgenden Dateien im Unterverzeichnis `heroes` gegeben:

[10] vgl. z.B. Hardcastle (1995):S. 182; Nørretranders (1997):S. 328.

hero-search.component.ts:

```
import { Component, OnInit } from '@angular/core';

import { Observable, Subject } from 'rxjs';

import {
   debounceTime, distinctUntilChanged, switchMap
 } from 'rxjs/operators';

import { Hero } from '../model/hero';
import { HeroService } from '../hero.service';

@Component({
  selector: 'app-hero-search',
  templateUrl: './hero-search.component.html',
  styleUrls: [ './hero-search.component.css' ]
})
export class HeroSearchComponent implements OnInit {
  heroes$!: Observable<Hero[]>;
  private searchTerms = new Subject<string>();

  constructor(private heroService: HeroService) {}

  // Push a search term into the observable stream.
  search(term: string): void {
    this.searchTerms.next(term);
  }

  ngOnInit(): void {
    this.heroes$ = this.searchTerms.pipe(
      // wait 300ms after each keystroke
      debounceTime(300),
      // ignore new term if same as previous term
      distinctUntilChanged(),
      // switch to new observable when term changes
      switchMap((term: string) => this.heroService.
          searchHeroes(term)),
    );
  }
}
```

hero-search.component.html:

```
<div id="search-component">
<label for="search-box">Search Hero: </label>
<input #searchBox id="search-box" (input)="
    search(searchBox.value)" />

<ul class="search-result">
  <li *ngFor="let hero of heroes$ | async">
    <a routerLink="/detail/{{hero.id}}">
      {{hero.name}}
    </a>
  </li>
</ul>
</div>
```

Betrachten wir zunächst das Template. In dem Input-Tag wird ein Event-Handler für registriert, der die `search`-Methode der Komponentenklasse nach einer Eingabe aufruft. Darunter werden die zu dem Suchbegriff gefundenen Einträge aufgelistet. Das $-Zeichen am Ende einer Variable ist in Angular eine Konvention für die Bezeichner einer Observablen. Eigentlich muss sie ja mit `subscribe` abonniert werden, damit sie ausgeführt wird. Hier aber wird sie in der *ngFor-Direktive mit dem Pipe-Zeichen `|` und `async` als *AsyncPipe* abonniert:

```
<li *ngFor="let hero of heroes$ | async">
```

Damit brauchen wir sie nicht mehr in der Komponentenklasse mit `subscribe` zu abonnieren. Die Property `searchTerm` ist ein RxJS Subject, das die Werte einer Observablen darstellen und selbst wieder eine Observable ist. Mit `next` wird ein neuer Wert in der Observablen gespeichert. So wird ein Datenstrom aus Suchbegriffen erzeugt. Gesteuert wird das Ganze durch eine Pipe, also eine Hintereinanderschaltung von drei Operatoren:

- `debounceTime(300)` wartet, bis der Fluss der Eingabe-Events für 300 Millisekunden pau-

siert, bevor zum letzten String gesprungen wird.

- `distinctUntilChanged()` sichert, dass ein Request nur gesendet wird, wenn der Suchtext verändert wurde.
- `switchMap()` ruft für jeden Suchtext, der es durch `debounce()` und `distinctUntilChanged()` geschafft hat, den Suchservice auf. Sie verwirft vorherige Suchobservablen und gibt nur die letzte zurück. Die wird in dem Template asynchron aufgerufen.

Die Wirkung dieser ganzen Konstruktion ist, dass bei jedem Tastendruck nach 300 ms der Suchtext in der Datenbank gesucht und ie Ergebnismenge als Liste in dem Template angezeigt wird.

18.4.4 Zusammenfassung: Erstellung einer SPA mit Angular

Hat man eine SPA vom Navigationsfluss der Views bzw. dem Workflow wie in Abbildung 18.11 entworfen und serverseitige Web API zur zentralen Datenbank erstellt, z.B. mit PHP, werden im wesentlichen die in Abschnitt 18.3.3 beschriebenen Schritte durchgeführt. Durch die Besonderheit des Ablaufs der Views einer SPA sind nun jedoch im ersten Schritt die Einrichtung des Routings notwendig sowie im letzten Implementierungsschritt die Festlegung der Routen.

1. Erzeugung eines Angular-Projekts mit `ng new ... --routing`
2. Spiegelung der Datenbanktabellen als Interfaces mit `ng generate interface ...`. Hier müssen die Properties der Interfaces den Spaltennamen der Tabellen exakt entsprechen und ihre Datentypen kompatibel sein.
3. Import der Module `FormsModule` und `HttpClientModule` in der Hauptdatei `app.module.ts`.
4. Erzeugung von Services mit `ng generate service ...`, die die wesentlichen Funktionen der Anwendung ermöglichen, insbesondere die Zugriffe auf die Serverprogramme. Dazu muss das Modul `HttpClientModule` im Hauptmodul `app.module.ts` und die Module `HttpClient` und `Observable` im Service importiert werden.
5. Erzeugen der Komponenten mit `ng generate component ...`, die die implementierten Funktionen einbinden und die Ergebnisse im Browser darstellen. Mit diesem Konsolenbefehl werden nicht nur die notwendigen Dateien erzeugt, sondern die Komponente auch im Hauptmodul `app-module.ts` registriert. Zusätzlich sind in der Datei `app-routing.module.ts` die Komponenten zu importieren und die Routen festzulegen.
6. Ist das Projekt fertig entwickelt und lauffähig, so kann es mit dem Befehl

   ```
   ng build --base-href ./
   ```

   als App in JavaScript erstellt werden. Die Dateien dazu werden dann in dem Unterverzeichnis `dist` erstellt und können auf einen beliebigen Webserver deployed werden.

19

WebSockets

WebSockets sind eine Webtechnik, die auf Basis der TCP/IP-Protokolle einen bidirektionalen Kommunikationskanal zwischen einem Webclient und einem Webserver aufzubauen, also Nachrichten während einer stehenden Verbindung in beide Richtungen geschickt werden können. Das grundsätzliche Problem dabei ist, dass HTTP (bzw. HTTPS) als Protokoll zwischen Browser und Server zustandslos ist und damit auf Protokollebene gar keine Verbindung ermöglicht, erst recht keine bidirektionale. Zwar kann eine Verbindung in der auf HTTP aufsetzenden Anwendungsschicht aufgebaut werden, beispielsweise durch Sessions, allerdings kann auch hier nach dem starren Request-Response-Muster von HTTP der Server immer nur auf Anfragen des Clients reagieren und nicht selbst spontan Nachrichten an den Client senden.

Zwar kann man mit dem Objekt `XMLHttpRequest`, das als Standardobjekt in JavaScript zur Verfügung steht, mit Hilfe des Pollings eine Verbindung simulieren, allerdings sehr aufwendig. Da die Clients vor jeder Servernachricht immer erst eine Anfrage an ihn gestellt haben müssen, kann die Kommunikation bei vielen Servernachrichten und vielen Clients unverhältnismäßig hoch werden und zu stockenden Verbindungen führen. In diesem Kapitel werden wir ein noch recht junges Protokoll kennenlernen, mit dem eine echte bidirektionale Datenübertragungen zwischen Browser und Webserver eingerichtet werden können.

Kapitelübersicht

19.1 Das WebSocket Protokoll

Wie kann man aber überhaupt eine bidirektionale Kommunikation zwischen Browser und Webserver ermöglichen? Da HTTP ein zustandsloses Protokoll ist, kann es über HTTP ganz prinzipiell nicht funktionieren. Das darunterliegende TCP dagegen ist verbindungsorientiert und könnte dafür verwendet werden. Die Grundidee ist daher, dass WebSockets über HTTP, also über eine

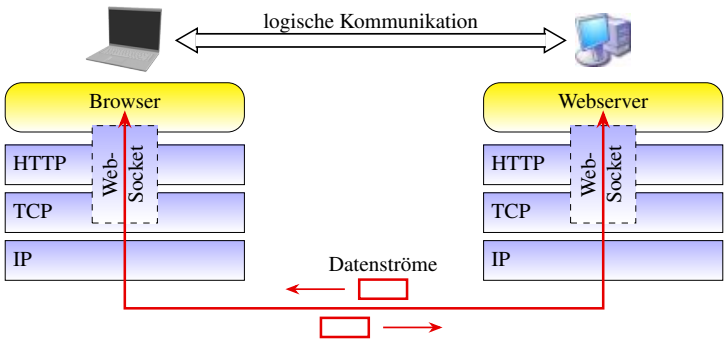


Abbildung 19.1: Prinzip der WebSocket-Verbindung über TCP als „Tunnelverbindung“ durch HTTP und TCP.

Anfrage eines Webclients, den Aufbau einer bidirektionalen Verbindung über TCP ermöglichen. Wie in Abbildung 19.1 illustriert, tunnelt eine WebSocket-Verbindung gewissermaßen durch die Protokolle TCP und HTTP. Technisch wird dies dadurch bewirkt, dass der Browser eine HTTP-Anfrage zur Änderung des Protokolls stellt, worauf der Server mit dem Status-Code 101 antwortet, wenn er diesen Wechsel ausführen kann.

Ein WebSocket tunnelt durch HTTP, indem der Server die Anfrage nicht schließt, sondern mit dem Code 101 das Protokoll wechselt.

WebSockets wurden als Protokoll mit RFC 6455 (`http://tools.ietf.org/html/rfc6455`) im Dezember 2011 eingeführt, die WebSocket API zur Implementierung von WebSocket-Clients in Browsern im September 2012 (`http://w3.org/TR/websockets`). Das WebSocket-Protokoll basiert direkt auf TCP, baut jedoch die WebSocket-Verbindung über eine HTTP-Anfrage auf und am Ende wieder ab. (RFC 6455, §1.5). Das WebSocket-Protokoll ist in drei Teile gegliedert, das Opening Handshake, die bidirektionale Kommunikation mit Hilfe von *Data Frames* (also Datenpaketen variabler Länge), und das Closing Handshake. Der schematische Ablauf von Auf- und Abbau einer WebSocket-Verbindung nach Herstellung einer TCP-Verbindung ist in Abbildung 19.2 skizziert. Der URL wird für eine unverschlüsselte Verbindung mit dem Protokollschema `ws://` versehen. Analog wird mit `wss://` eine über TLS verschlüsselte Verbindung aufgebaut, die sich prinzipiell wie in Abbildung 19.2 darstellen lässt, wobei nach dem TCP-Handshake eine TLS-Verbindung mit einem TLS-Hanshake aufgebaut wird und die WebSocket-Verbindung nun innerhalb des TLS-Kanals abläuft.

Während eine stehenden WebSocket-Verbindung können beliebig lange *Data Frames* übertragen werden, die als Datentyp entweder Unicode-Text oder Binärdaten enthalten (RFC 6455, §5.6). Es gibt drei spezielle Frames, die *Control Frames* `Close`, `Ping` und Pong. Ein `Close`-Frame beendet eine WebSocket-Verbindung, während der Empfang eines `Ping`-Frames unverzüglich mit einem `Pong`-Frame zu beantworten ist. `Ping`- und `Pong`-Frames sind dazu vorgesehen, um den Status einer WebSocket-Verbindung zu prüfen, Latenzen (Signallaufzeiten) zu messen oder die Verbindung auch bei längeren Kommunikationspausen zu erhalten (was oft bei Verbindungen über Proxies wichtig sein kann).

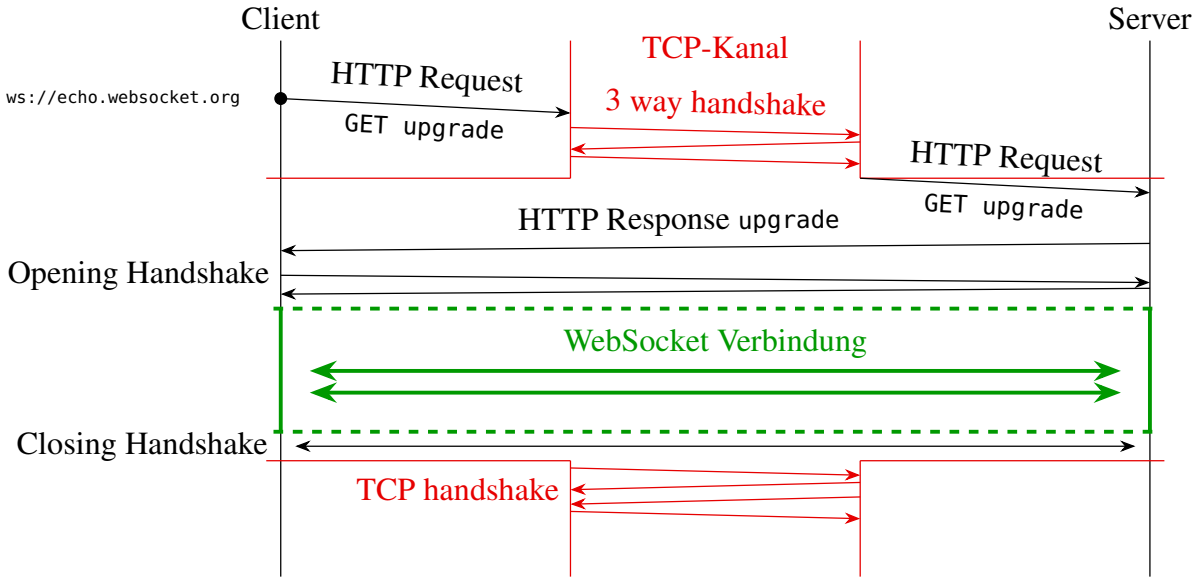


Abbildung 19.2: Auf- und Abbau einer WebSocket-Verbindung über TCP.

19.2 Die WebSocket API

Zur Implentierung eines WebSockets im Browser hat das W3C auf Basis des RFC 6455 eine Schnittstelle für WebSocket API's definiert (`http://w3.org/TR/websockets`). Hier werden die möglichen Zustände eines WebSockets sowie die möglichen Zustandsübergänge festgelegt, wie sie in Abbildung 19.3 dargestellt sind.

Zustand	Beschreibung
`CONNECTING` (readyState 0)	Der WebSocket wurde instanziiert und baut gerade eine Verbindung auf
`OPEN` (readyState 1)	Die Verbindung ist erfolgreich aufgebaut und ein bidirektionaler Kommunikationskanal steht zur Verfügung
`CLOSING` (readyState 2)	Die Verbindung durchläuft einen Closing Handschake und wird beendet
`CLOSED` (readyState 3)	Die Verbindung ist geschlossen oder konnte gar nicht erst aufgebaut werden

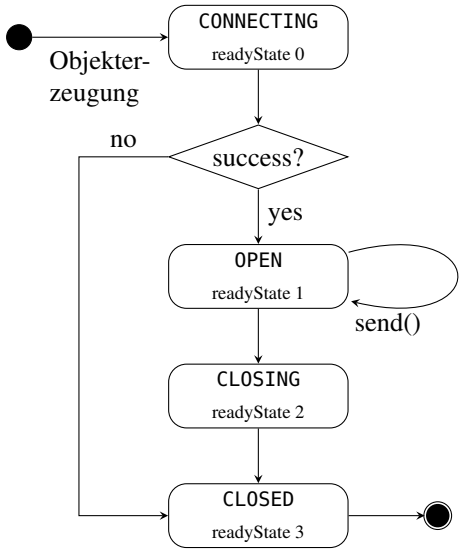


Abbildung 19.3: Zustände und Zustandsdiagramm eines WebSockets. Nach[1]

Ferner muss ein WebSocket-Objekt die in Tabelle 19.1 aufgeführten Event-Handler implementieren, die auf die jeweiligen Ereignisse reagieren. In JavaScript werden die Event-Handler als Funktionen implementiert; oft werden dabei insbesondere `onopen` und `onclose` ohne Parameter überladen, also als `onopen()` oder als `onclose()`.

Für weitere Informationen und Demos siehe `http://www.websocket.org/`. Dort findet man auch einen einfachen WebSocket Echodienst, mit dem man seinen WebSocket-Client ausprobieren und testen kann.

Event Handler	Bemerkung
`onopen(open)`	Wird ein WebSocket in den Zustand `OPEN` versetzt, wird ein Ereignis `open` erzeugt, das dieser Event Handler verarbeiten kann.
`onmessage(message)`	Trifft eine Nachricht beim Client ein, wird das Ereignis `message` ausglöst und dem Event Handler `onmessage` übergeben. (Nachrichten umfassen nach RFC 6455, §1.2 insbesondere die Nutzdaten der WebSocket-Verbindung.)
`onerror(error)`	Tritt ein Fehler beim Verbindungsaufbau, während der Datenübertragung oder beim Verbindungsabbau auf, wird der Event Handler `onerror` aufgerufen und mit dem Ereignis `error` die Gründe dafür übertragen.
`onclose(close)`	Gelangt ein WebSocket in den Zustand `CLOSED`, wird der Event Handler `onclose` ausgelöst und das Ereignis `close` übergeben, das weitere Informationen enthält.

Tabelle 19.1: Die Event Handler eines WebSockets, Gorski et al. (2015:S. 83)

19.3 Programmierung eines WebSocket-Clients

Als Programmierbeispiel eines WebSocket-Systems betrachten wir den einfachsten Fall, einen „Echo-Service“, also ein Kommunikationssystem, bei dem ein zentraler Server einfach die Eingaben eines Clients zurücksendet. Für jeden einzelnen Client, der den Server aufruft, erscheint also ein Terminalfenster im Browser, in dem die eigenen Eingaben und die Antworten des Servers erscheinen. Ein solches Echosystem ist für jedes Kommunikationssystem insofern ein gutes Einführungsbeispiel, da weder Client noch Server eine eigene Logik benötigen und sich die Programmierung auf die für die reine Kommunikation notwendigen Anweisungen beschränkt.

Betrachten wir zunächst den Echo-Client. Da er vom Browser aus gestartet wird, ist er als JavaScript in einem HTML-Dokument integriert:

Listing 19.1: WebSocket-Client für einen Echo-Server

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>WebSockets - A Simple Echo Server</title>
  <script>
var onload = function() {
   var terminal = document.getElementById("terminal"); // reference for outputs

   // open WebSocket connection
   var connection = new WebSocket('ws://194.94.2.20:18087/echo');

   // The event handlers:
   connection.onopen = function() {
      document.getElementById("status").innerHTML = connection.readyState;
   };

   connection.onclose = function() {
      document.getElementById("status").innerHTML = connection.readyState;
   };

   connection.onmessage = function(message) {
      document.getElementById("status").innerHTML = connection.readyState;
      if (message.data) { // only print of there are data sent
```

```
            terminal.innerHTML += "SERVER: " + message.data + "\n";
            if (message.data === "bye") {
               connection.close();
            }
         }
         terminal.scrollTop = terminal.scrollHeight; // focus to the last line
         document.getElementById("status").innerHTML = connection.readyState;
      };

      /* Event handler for input field: Send message when user presses Enter key. */
      document.getElementById("input").onkeydown = function(event) { // event
          handler
         if (event.keyCode === 13) { // <= Enter"key
            var msg = document.getElementById("input").value;
            if (msg) { // only act if message is not empty ...
               connection.send(msg);
               document.getElementById("input").value = "";
               terminal.innerHTML += "YOU: " + msg + "\n";
               terminal.scrollTop = terminal.scrollHeight; // focus to the last line
            }
         }
         document.getElementById("status").innerHTML = connection.readyState;
      };
   }
     </script>
   </head>
   <body>
     <textarea id="terminal" rows="10" cols="80" readonly></textarea>
     <p>
       <input type="text" id="input" size="81" placeholder="Send a message, quit
           with 'bye'"/>
     </p>
     <div>(Connection readyState: <span id="status"></span>)</div>
   </body>
   </html>
```

Der HTML-Teil am Ende des Listings ist auf das Wesentliche beschränkt, es ist lediglich ein Textbereich (`<textarea>`) als „Terminalfenster“ und ein Textfeld zur Eingabe des Anwenders. Der Textbereich ist übrigens als mit dem Attribut `readonly` versehen, so dass darin keine Eingaben gemacht werden können.

Der gesamte Quelltext in JavaScript ist im `<head>`-Element des Dokuments konzentriert. Das ist möglich, da das Programm komplett innerhalb des Ereignisbehandlers `onload` in Zeile 7 eingepackt ist, der nach Beendigung des Seitenaufrufs verarbeitet wird und daher auf den gesamten DOM-Baum zugreifen kann. Zum Ablauf des Programms: Nachdem als erstes in Zeile 8 eine Referenzvariable `terminal` zum Zugriff auf das „Terminal“ im HTML-Dokument definiert wird, wird in Zeile 11 ein WebSocket-Objekt `connection` mit dem URI `ws:// ...` des WebSocket-Servers erzeugt. In den darauf folgenden Zeilen werden dann die für ein WebSocket notwendigen Ereignisbehandler implementiert. In unserem Beispiel wird in jedem Ereignisbehandler der Statuscode `readyState` der Verbindung angezeigt.

Wo wird jedoch eine Nachricht über den Socketkanal gesendet? Ab Zeile 14 wird dazu der

Ereignisbehandler `onkeydown` für das Eingabefeld implementiert, der bei Drücken der Entertaste (`keyCode` 13) den Eingabetext `msg` in Zeile 38 abschickt.

19.3.1 Programmierung eines WebSocket-Servers

Ein WebSocket-Server muss, wie allgemein jeder Server, als eigenständiger Prozess laufen, der in einer Endlosschleife permanent auf Serviceanfragen horcht. Wir können ihn daher nicht mit PHP als Unterprogramm des Webservers oder mit JavaScript als Unterprogramm des Browsers programmieren, sondern benötigen eine Programmiersprache, die eigenständig ablaufende Programme ermöglicht. Eine populäre Möglichkeit ist ein Server mit `node.js` mit JavaScript. Auch für Java gibt es recht komfortable Lösungen, eine davon is das auf dem Netty-Server basierende Projekt *Webbit* `http://webbitserver.org/` von Joe Walnes. Verwendet man die dort verfügbaren jar-Bibliotheken, so genügt zur Implementierung eines WebSocket-Servers ein kurzes Java-Programm wie in Listing 19.2.

Listing 19.2: Einfacher Echo WebSocketServer in Java

```
import org.webbitserver.handler.StaticFileHandler;
import org.webbitserver.BaseWebSocketHandler;
import org.webbitserver.WebServer;
import org.webbitserver.WebServers;
import org.webbitserver.WebSocketConnection;

/**
 * This class enables to let start a simple echo WebSocket server and
 * offers event handlers to communicate with a WebSocket client.
 * It bases on the webbit package
 * <a href="http://webbitserver.org/">http://webbitserver.org/</a> by Joe Walnes.
 * @author Andreas de Vries
 */
public class EchoWebSocketServer extends BaseWebSocketHandler {
   /** Path and port on this web server to connect to the WebSocket.
    *  The WebSocket client then has to invoke the URI "ws://hostname:port/path".
    */
   static String path = "/echo";
   static int port = 18087;

   /** Stores the current numbers of sockets on this WebSocket server. */
   private int connections = 0;

   /**
    * Event handler reacting on the opening of a WebSocket connection.
    * @param connection WebSocket connection
    */
   @Override
   public void onOpen(WebSocketConnection connection) {
      this.connections++;
      connection.send("Welcome!");
   }

   /**
```

```
     * Event handler reacting on the closing of a WebSocket connection.
     * @param connection WebSocket connection
     */
    @Override
    public void onClose(WebSocketConnection connection) {
       this.connections--;
    }

    /**
     * Event handler reacting on the opening of a WebSocket connection.
     * @param connection WebSocket connection
     */
    public void onError(WebSocketConnection connection) {
       System.err.println("An error occurred: " + connection);
    }

    /**
     * Event handler reacting to the reception of a message via a WebSocket
         connection.
     * @param c WebSocket connection
     * @param message the message received via the WebSocket connection
     */
    @Override
    public void onMessage(WebSocketConnection c, String message) {
       if (message.equalsIgnoreCase("Bye")) {
          c.send("Bye!");
          c.close();
       } else {
          c.send("Aha, you said: \"" + message + "\"");
       }
    }

    /**
     * Starts an echo WebSocket server.
     * @param args command line parameters (are ignored in this method)
     */
    public static void main(String[] args) {
       WebServer webServer = WebServers.createWebServer(port);
       webServer.add(new StaticFileHandler("/static-files"));
       webServer.add(path, new EchoWebSocketServer());
       webServer.start();
       System.out.println("WebSocket Server is running on port "+port+" ...");
    }
}
```

Wie der WebSocket-Client implementiert der Server die notwendigen Ereignisbehandler. Zusätzlich wird der Server in der main-Methode ab Zeile 70 gestartet, der Port und der Pfad sind als statische Attribute zu Beginn der Klasse festgelegt. Desweiteren hat der Server ein Attribut `connections`, in dem die aktuelle Anzahl bestehender Verbindungen gespeichert wird.

Der eigentliche Kern des Servers ist der Ereignisbehandler `onMessage` ab Zeile 57, der bei

Empfang der Nachricht `"bye"` die Verbindung schließt, alle anderen Nachrichten einfach nur wiederholt und abschickt (Zeile 62).

19.4 Anwendungsfälle

Da man mit WebSockets prinzipiell jedes beliebige Kommunikationsprotokoll einbinden kann, sind die Anwendungsfälle für WebSockets schier unbegrenzt. Einige Beispielanwendungen sind auf der Website `http://www.websocket.org` aufgeführt, so etwa Chatserver, Steuerung von entfernten Systemen wie Bildschirme, Sensoren und Aktoren (WebCams oder Smart Homes) über einen Browser, oder Computerspiele. Entsprechend entstanden in der letzten Zeit einige auf WebSockets basierende Echtzeitprotokolle, so zum Beispiel WAMP (`http://wamp-proto.org/`) der Tavendo GmbH (`http://tavendo.com/`) mit der darauf aubbauenden quelloffenen JavaScript-Bibliothek `http://autobahn.ws` und dem WAMP-Router `http://crossbar.io`.

Grundsätzlich ist mit WebSockets zwar nichts realisierbar, was nicht auch mindestens genauso effizient, manchmal sogar effizienter, über ein Netzwerk mit einem für die erwünschte Kommunikation spezifiertem Netzwerkprotokoll direkt realisiserbar wäre. Die großen Vorteile einer WebSocket-Verbindung sind jedoch zweierlei. Einerseits kann mit dem Web die Infrastruktur und die Verbindungen zu weltweit Knoten verwendet werden und muss nicht erst teuer und langwierig aufgebaut werden. Andererseits kann mit WebSockets jeder Browser als Client fungieren, ganz egal auf welchem Betriebssystem er läuft, es muss die Clientsoftware nicht erst für jede einzelne Plattform und jedes Endgerät erstellt und verteilt werden. Damit hat eine WebSocket-Anwendung praktisch instantan eine weltweite Reichweite.

Für anspruchsvolle Kommunikationsprotokolle bieten WebSockets also einige Möglichkeiten. Aus diesem Grund werden meines Erachtens WebSockets eine Schlüsseltechnologie künftiger Kommunikationssysteme bilden, insbesondere für das Internet der Dinge (*Internet of Things* IoT). Einige Ansätze der vielfältigen Möglichkeiten werden wir in diesem Abschnitt betrachten.

19.4.1 Chatsysteme

Eine naheliegende Anwendung von WebSockets ist ein Chatsystem, also ein elektronisches Kommunikationssystem, bei dem sich die Teilnehmenden in Echtzeit öffentlich vor allen anderen Teilnehmenden äußern oder privat, separiert in eingerichteten Chatrooms. In einem Chatsystem muss die Kommunikation notwendig bidirektional sein, denn jeder kann jederzeit jedem anderen Teilnehmer eine Nachricht mitteilen. Der Chat-Server fungiert als zentrale Schaltzelle, in dem alle Kanäle zusamentreffen und die Teilnehmende mit ihren etwaigen Berechtigungen verwaltet.

Ein Beispiel für ein Chatsystem ist das auf der PHP-Bibliothek Ratchet basierende Chatsystem `http://socketo.me/demo`, bei dem man sich mit einem Usernamen anmeldet und spontan separierte Chatrooms bilden kann.

19.4.2 XMPP

Das *Extensible Messaging and Presence Protocol* (*XMPP*) ist ein von der IETF mit den RFC's 6120, 6121, 6122, 3922 und 3923 definierter Internetstandard für XML-Routing. XMPP folgt dem XML-Standard und wird primär für Instant Messaging eingesetzt. XMPP und seine Erweiterungen unterstützen unter anderem Funktionen zur Nachrichtenübermittlung, Konferenzen mit mehreren Benutzern, Anzeigen des Online-Status, Dateiübertragungen und Versendung von Zertifikaten.

Für den Betrieb eines XMPP-Netzwerkes wird mindestens ein XMPP-Server benötigt. Jeder an das Internet angebundene XMPP-Server kann Nachrichten mit anderen Servern austauschen. So sind Verbindungen über Anbieter-Grenzen hinweg möglich. Nachrichten werden vom Sender über eigenen Server und den Server des Empfängers zum Empfänger weitergeleitet. Die Adressierung geschieht über das Format `user@example.com` wie bei SMTP (also E-Mails), wobei `user` die Kennung des Benutzers und `example.com` der URI des Servers ist.

Verbreitete Kommunikationsdienste, die XMPP einsetzen, sind WhatsApp und Jitsi. Der Dienst WhatsApp ist ein auf einer angepassten Version von XMPP basierendes Netzwerk und stellt die Kommunikation zwischen Sendern und Empfängern über einen einzigen Server `s.whatsapp.net` her. Die Benutzerkennung ist hier dessen Telefonnummer, also z.B.

`01234@s.whatsapp.net.`

Im Februar 2010 öffnete Facebook eine Chat-Anbindung für fremde Anwendungen via XMPP,[2] allerdings läuft intern kein XMPP-Server.[3] Am 19. Februar 2014 übernahm Facebook WhatsApp für etwa 19 Millionen Dollar.

19.4.3 WhatsApp Web

Seit Januar 2015 existiert *WhatsApp Web* ist eine Anwendung, mit der man per Browser direkt auf seinen WhatsApp-Account zugreifen und Daten mit dem Smartphone austauschen kann.[4] Es erstellt eine verschlüsselte WebSocket-Verbindung vom Browser über den WhatsApp-Server zum Smartphone. Das Smartphone muss daher für die gesamte Verbindung angeschaltet und mit dem Internet verbunden sein und ist die zentrale Steuereinheit der Verbindung. Das Protokoll der WebSocket-Verbindung ist ein modifiziertes XMPP. Der Ablauf des Verbindungsaufbaus ist dabei wie folgt:[5]

1. Der Browser (Web-Client) ruft WhatsApp Web unter `http://web.whatsapp.com` auf.
2. Im Browser erscheint ein QR-Code mit Anmeldedaten für einen XMPP-Kanal über WebSocket, den nur ein bei WhatsApp angemeldetes Smartphone freischalten kann.
3. Scannt das Smartphone des Web-Accounts den QR-Code ein, so wird eine bidirektionale verschlüsselte XMPP-Verbindung zwischen Browser (Web-Client) und Smartphone (mobiler Client) aufgebaut, eingepackt in der WebSocket-Verbindung.

Zu bemerken ist dabei, dass die Verbindung zwischen Browser und Smartphone zwar verschlüsselt ist, aber zu jedem Zeitpunkt über den WhatsApp-Server läuft. Damit ist die aus kryptologischer Sicht klassische Situation eines *Man-in-the-Middle-Angriffs* gegeben[6]. Da WhatsApp die Quelltexte seiner Server-Software nicht öffentlich macht, kann eine Entschlüsselung und Speicherung der übermittelten Daten daher nicht ausgeschlossen werden.

19.4.4 Computerspiele

WebSockets können (und werden so meine Prognose) auf vielfältige Weise für vernetzte Computerspiele eingesetzt werden. Einerseits ermöglichen sie es, Einzelspielerspiele mit mehreren

[2] `http://blog.facebook.com/blog.php?post=297991732130`

[3] Facebook Chat API: `http://developers.facebook.com/docs/chat/`

[4] `https://blog.whatsapp.com/614/WhatsApp-Web`

[5] `http://censore.blogspot.de/2015/01/breaking-open-httpswebwhatsappcom.html` [2016-02-24]

[6] de Vries (2012a).

Bildschirmen zu spielen, beispielsweise Tetris auf einem großen Bildschirm, aber gesteuert über ein mobiles Endgerät, das den eingeblendeten QR-Code einscannt (`http://www.websocket.org/demos/tetris/`).

19.5 Webkonferenzen mit WebRTC

In den letzten Jahren etablierten sich im unternehmerischen Umfeld und vor allem im Bildungsbereich zunehmend der Einsatz von Webkonferenzen. *Webkonferenzen*, oft auch *Online-Meetings* genannt, sind eine Form der internatbasierten bidirektionalen Echtzeitkommunikation *(real time communication)*. Es handelt sich um ein virtuelles Treffen mehrerer Teilnehmenden, die mit ihren Browsern einen vereinbarten Hyperlink eines geeignet eingerichteten zentralen Webservers aufrufen und damit eine gemeinsame Sitzung bilden. Der Server dient also als

Webkonferenz

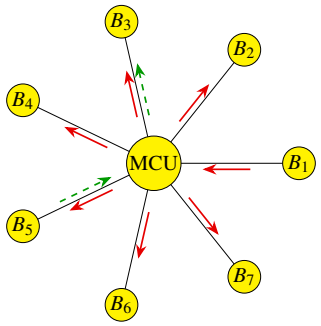


Abbildung 19.4: Prinzip einer Webkonferenz mit einem zentralen Server als Vermittler (MCU) und den Browsern B_1, B_2, ... als Teilnehmende. Es können mehrere Datenströme simultan übertragen werden, hier Ton (→) und ein privater Chat (- →)

technischer Vermittler der Echtzeitkommunikation, wie in Abbildung 19.4 skizziert. Da er ein Webserver ist, wird zur Teilnahme an einer Webkonferenz bis auf einen Browser grundsätzlich keine weitere Software benötigt. Er ist also ein sogenannter MCU *(Multipoint Control Unit)* und verteilt die Datenströme zwischen allen Teilnehmenden. Grundsätzlich können Datenströme zwischen allen oder nur einzelnen Teilnehmenden simultan übertragen werden, beispielsweise der Ton nur vom Sprechenden zu allen anderen (rote Pfeile in Abbildung 19.4), oder ein privater Chat von einem zum anderen (gestrichelte grüne Pfeile in Abbildung 19.4).

Bei vielen Anbietern von Webkonferenzen muss eine Webkonferenz von einem Moderator oder Organisator eingerichtet werden, der als Nutzer registriert sein muss. Beispiele für solche Systeme sind Zoom, BigBlueButton, Google Meet, Skype oder Wonder.me. Grundsätzlich ist es technisch aber durchaus möglich, einen Webserver so einzurichten, dass er für anonyme[7] Teilnehmende Webkonferenzen zulässt. Eine populäre Software für solche anonymen Konferenzen ist das quelloffene Projekt Jitsi Meet. Ebenso ermöglicht Senfcall[8] solche datenschutzkonformen und anonymen Sitzungen, ein von Studierenden der Universität Darmstadt betriebenes Projekt. Es basiert auf dem quelloffenen Konferenzsystem BigBlueButton.[9]

Die meisten Systeme für Webkonferenzen basieren auf Websockets. Einige, wie offenbar Google Meet (vormals Hangouts) oder GoToMeeting von LogMeIn, scheinen dagegen eine Sitzung mit XHtmlRequests zu implementieren. In jedem Fall muss also ein Browser zur Teilnahme an einer Konferenz das notwendige Sitzungsprotokoll als JavaScript-Programm ausführen. Dieses Sitzungsprotokoll kann eine für das jeweilige System spezifische Implementierung sein, seit

WebRTC

[7] genauer gesagt natürlich: pseudoanonym, d.h. bis auf die IP-Adresse anonym

[8] `https://senfcall.de/`

[9] `https://github.com/bigbluebutton/bigbluebutton`

Januar 2021 jedoch empfiehlt das W3C die JavaScript-API WebRTC als Standard. Es entstand 2011 als quelloffenes Projekt von Google.[10] Grundlage für WebRTC sind die Protokolle SIP, RTP und RTCP. SIP (RFC 3261) dient dazu, eine Streaming-Sitzung einzurichten. Mit SIP kann ein logisches Peer-to-Peer-Netzwerk auf Basis von UDP eingerichtet werden, in das sich Teilnehmende *(user agents)* über ihre URI registrieren oder eingeladen werden können. Ein

SIP

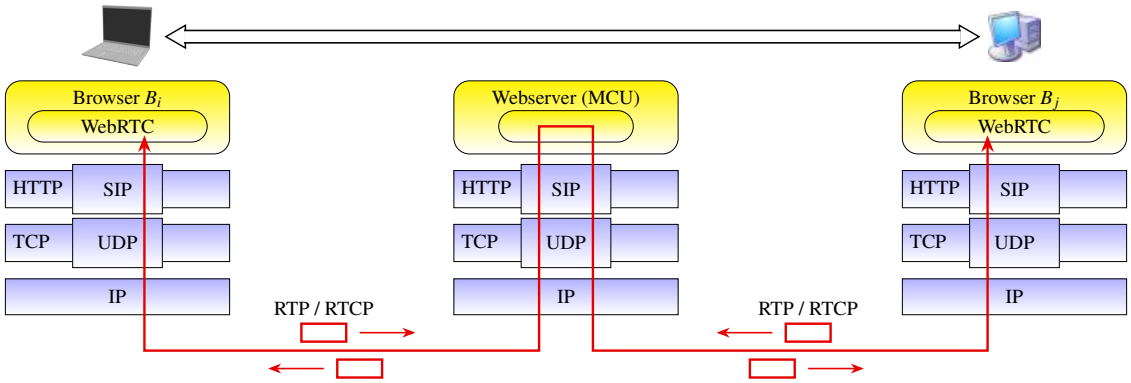


Abbildung 19.5: Protokolle und Datenwege (↔) einer Webkonferenz; die logische Kommunikation (⇔) findet nur zwischen den Teilnehmenden statt und bildet ein SIP-Netzwerk, die Streaming-Daten werden in RTP- und RTCP-Paketen übertragen

SIP-Netzwerk kann im Rahmen einer Websocket-Verbindung aufgebaut werden (RFC 7118). Meist wird SIP auf Basis von UDP implementiert, kann aber grundsätzlich auch über TCP kommunizieren.

RTP und RTCP

Die Streaming-Daten während einer SIP-Sitzung werden mit den Protokollen RTP und RTCP (beide RFC 3550) übertragen. Hierbei werden die Datenströme selbst über das Protokoll RTP in Form von Datenpaketen übertragen. RTP arbeitet mit RTCP zusammen, das die Dienstqualität *(Quality of Service)*) optimiert und die Sitzungsteilnehmenden identifiziert. RTCP stellt zum Beispiel Verzögerungszeiten oder Ausfälle von Datenpaketen fest und passt gegebenfalls Übertragungsraten an; auch können mit RTCP getrennte Medienströme zwischen verschiedenen Teilnehmenden gesteuert werden.

Die verschlüsselten Protokollvarianten von RTP und RTCP sind SRTP und SRTCP, sie sind bei der IETF seit 2004 unter RFC 3711 als Standard vorgeschlagen und werden inzwischen von den meisten Browsern unterstützt. Die an einer Webkonferenz beteiligten Protokolle, Daten- und Kommunikationswege sind in Abbildung 19.5 schematisch dargestellt.

Verschiedene gängige Systeme für Webkonferenzen sind unter `https://de.wikipedia.org/wiki/Liste_von_Webkonferenz-Software` aufgelistet.

19.5.1 Jitsi Meet

Eine populäre Software für Webkonferenzen ist Jitsi Meet. Es ist Teil des Projekt Jitsi, das 2003 Jahren an der Universität Straßburg durch Emil Ivov entstand und als quelloffene Version auf GitHub gehostet ist.[11] Als Client kann ein Browser (HTML5) genutzt werden, aber es existieren auch native Apps für Android (mit Java) und iOS (in Objective-C). Als Konferenzserver dient der XMPP-Server Prosody, der lokal installiert werden kann. Es existiert aber auch ein freier Dienst `https://meet.jit.si`, der die adhoc Einrichtung einer Webkonferenz mit einem beliebigen Raumnamen ermöglicht und als MCU einen Prosody-Server hat. Die Teilnehmerzahl

[10] `http://lists.w3.org/Archives/Public/public-webrtc/2011May/0022.html` [2021-02-29]
[11] `https://github.com/jitsi`

bei `meet.jit.si` ist begrenzt auf 75 Personen.[12] Eine grundsätzliche Begrenzung der Teilnehmerzahl bei einem eigenen selbstgehosteten Server gibt es nicht, sie hängt natürlich stark von Einrichtung und Ausstattung des Servers und der Netzauslastung ab.

Es existieren zwei JavaScript-APIs zur Implementierung eines Jitsi-Meet-Clients, die `IFRame API`[13] zur einfachen Einbettung einer Konferenz in eine eigene Webseite, und die `lib-jitsi-meet API (low level)`[14] zur flexibleren, aber auch komlexeren Implementierung von Konferenzen, z.B. mit eigener GUI. Ein einfaches Hallo-Welt-Programm zur Erstellung einer Webkonferenz mit dem Jitsi-Server ist in Listing 19.3 dargestellt.

JavaScript APIs

Listing 19.3: „Hallo-Welt“ einer Webkonferenz mit Jitsi Meet

```
<!DOCTYPE HTML>
<html lang="de">
  <head>
    <title>WebTech Konferenzraum</title>
    <meta charset="utf-8">
  </head>
  <body>
    <script src="https://meet.jit.si/external_api.js"></script>
    <script>
      var domain = "meet.jit.si";
      var options = {
        roomName: "webtech-konferenz",
        width: 840,
        height: 540,
        parentNode: undefined,
        configOverwrite: {},
        interfaceConfigOverwrite: {}
      }
      var api = new JitsiMeetExternalAPI(domain, options);
    </script>
  </body>
</html>
<!--// Source https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-iframe
    -->
```

Hier wird in der Domain `meet.jit.si` ein Konferenzraum mit dem Namen `webtech-konferenz` in einem Fenster der Größe 840×540 eingerichtet. Für einen eigenen Jitsi-MCU muss entsprechend dessen Domain angegeben werden.

Zentral ist also, die externe JavaScript-Bibliothek `external_api.js` von `meet.jit.si` einzubinden, sowie die obligatorische Variable `domain` mit der URL des MCU-Servers und das Objekt `options` mit den gewünschten Werten zu belegen.

[12] `https://community.jitsi.org/t/22273/98`. In demselben Thread werden 2 Postings weiter Erfahrungswerte von bis zu 230 Teilnehmenden mit abgeschalteten Mikros und Kameras genannt.

[13] `https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-iframe`

[14] `https://jitsi.github.io/handbook/docs/dev-guide/dev-guide-ljm-api`

20
Webservices versus REST

Kapitelübersicht

20.1 SOA: automatisierte Dienste

20.1.1 Service-Orientierung als neues Paradigma

Die Abkürzung SOA steht für *service-oriented architecture*, d.h. der zentrale Begriff von SOA ist der Dienst oder die Dienstleistung.

Definition 20.1. [1] Ein Dienst *(service)* ist ein Programm oder eine Software-Komponente, die lokal auf einem Rechner oder allgemein über ein Netzwerk oder das Internet genutzt werden kann. Ein Dienst muss eine Schnittstellenbeschreibung in maschinenlesbarer Form besitzen, eine *Service Description*. Zugriff auf den Dienst sind nur über diese Schnittstelle möglich, genaue Details über die Implementierung können oder müssen nach außen verborgen bleiben. □

Der Begriff des Dienstes ist in der Informatik an sich nicht neu, er liegt schon dem klassischen Client-Server-Prinzip zugrunde. Ein Beispiel für einen Dienst ist das HTTP-Protokoll, bei dem ein Web-Server Webdokumente öffentlich bereitstellt und diese durch Browser als Clients abgerufen werden können. Implizit ist über das zugrunde liegende Protokoll HTTP auch eine Schnittstellenbeschreibung gegeben. Wir werden allerdings sehen, dass es im Sinne der SOA strenggenommen kein Dienst ist, da er nicht von anderen Diensten aufgerufen werden kann.

In der historischen Entwicklung des Software-Engineering und der Programmierung bildet SOA damit eine weitere Stufe, um der zunehmenden Komplexität von Software zu begegnen[2].

[1] Melzer (2008):§2.3.
[2] Melzer (2008):§2.6.

Paradigma	Kapselung	Beispiele
Prozedurale Programmierung	... der Funktionen	statische Methode
↓		
Objektorientierung (OOP)	... der Daten	Klassen
↓		
Komponentenentwicklung	... der Komponenten	JavaBeans, DCOM, Plug-ins
↓		
Service-Orientierung (SOA)	... der Dienste	Web Service

Das erste Programmierkonzept gegen den zunehmend unwartbar werdenden Spaghetti-Code in den Anfängen der Software-Entwicklung war die *prozedurale Programmierung*. Deren Ziel war es im Grunde, einzelne Funktionalitäten der Software zu kapseln, also zu isolieren, um sie als nur ein einzelnes Quelltextfragment an beliebig vielen Stellen abrufbar zu machen. Zentraler Begriff der prozeduralen Programmierung ist die mathematische Funktion, in Java implementierbar als statische Methoden. So wurde eine Strukturierung der Programme und die Erstellung von Modulen und ganzen Software-Bibliotheken möglich.

Mit zunehmender Komplexität der Software reichte jedoch dieses Konzept nicht mehr aus, es entstand die *Objektorientierung*, durch die nun auch die Daten gekapselt werden konnten. Aufgrund der Vielfalt an Systemen und Programmiersprachen in der Folgezeit entstand der Begriff der *Software-Komponente*, die sich durch eine von einer Programmiersprache unabhängig verwendbare Schnittstellen- und Verhaltensbeschreibung auszeichnet, also ein „programmiersprachen- oder plattformunabhängiges Objekt“. Ein Beispiel für Software-Komponenten sind Plug-ins.

Der Ansatz von SOA schließlich ist die Kapselung ganzer Dienste. In einer SOA muss es entsprechend drei verschiedene Rollen geben, den *Anbieter (service provider)*, den *Nutzer (service requester)* und den *Vermittlungsdienst (service registry)* (Abb. 20.1). Auf diese Weise

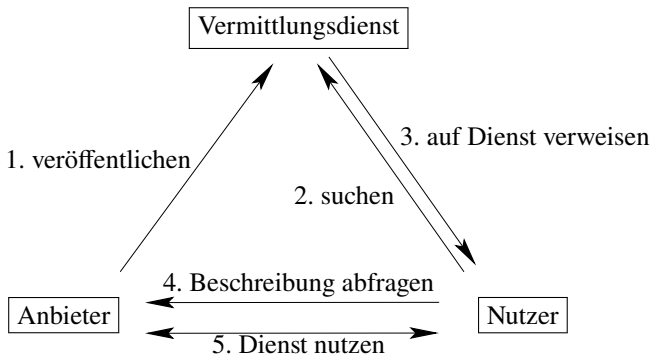


Abbildung 20.1: Das magische Dreieck des Prozessablaufs einer SOA

ist SOA also eine Verallgemeinerung sowohl des Begriffs der Software-Komponente als auch der Client-Server-Architektur.

Definition 20.2. [3] Eine *SOA* ist eine Systemarchitektur, die vielfältige und verschiedene Methoden oder Applikationen als wiederverwendbare und über ein Netzwerk plattform- und sprachu-

[3] Melzer (2008):§2.3.

nabhägig öffentlich zugreifbare Dienste darstellt. Jeder Dienst wird durch einen Vermittlungsdienst zwischen dem Nutzer und den Anbietern vermittelt. □

Eines der Hauptziele von SOA war es von Anfang an, *Geschäftsprozesse* durch Rechner durchführbar zu machen. SOA nutzt dabei die Sichtweise aus, dass die einzelnen Schritte eines Geschäftsprozesses *(business process)* sich als fachliche Dienste *(business services)* modellieren lassen[4]. SOA hat den Anspruch, die in den vorherrschenden Systemen „vertikal“ implementierten Dienste „horizontal“ zu verbinden: meist verwendet ein vollständiger Geschäftsprozess eben mehrere, in verschieden Systemen implementierte Dienste.

Praktische Probleme

Das Konzept der SOA ist schlüssig und auf den ersten Blick sehr einleuchtend. Dennoch gab und gibt es erhebliche Probleme, eine SOA praktisch zu implementieren. Sie ergeben sich aus dem Anspruch auf Allgemeingültigkeit von SOA, komplette Geschäftsprozesse über verschiedene Plattformen zu ermöglichen, und betreffen vor allem zwei Ebenen:

- *Standardisierung:* Auf der technischen Ebene ergibt sich durch die Forderung nach Plattformunabhängigkeit sofort das Problem, Standards über verschiedene Branchen und konkurrierende Unternehmen festzulegen, denn zur Informationsübermittlung und zum Dienstaufruf sind wohldefinierte Protokolle und Schnittstellen notwendig.
- *Unternehmenstrukturen.* Es müssen vollständige Geschäftsprozesse unternehmensintern, aber oft auch unternehmensübergreifend definiert, festgelegt und dokumentiert sein. Dieses Problem liegt nicht auf Ebene der IT, sondern zunächst auf betriebswirtschaftlicher Ebene. In einem ersten Schritt müssen also die abzubildenden Geschäftsprozesse überhaupt modelliert werden. Viele Unternehmen sind jedoch gar nicht prozessorientiert („horizontal“) strukturiert, sondern hierarchie- oder aufbauorientiert (“vertikal“).

20.1.2 Webservices

Webservices, oder auch *Webdienste*, sowie eine Vielzahl damit in Zusammenhang stehenden Spezifikationen sind eine mögliche Implementierungstechnik einer SOA. Das W3C definiert einen Webservice als eine Technik zur Maschine-Maschine-Kommunikation, bei der ein Mensch zwar Initiator sein kann, aber den Webservice nur mittelbar nutzt. Insbesondere benennt das

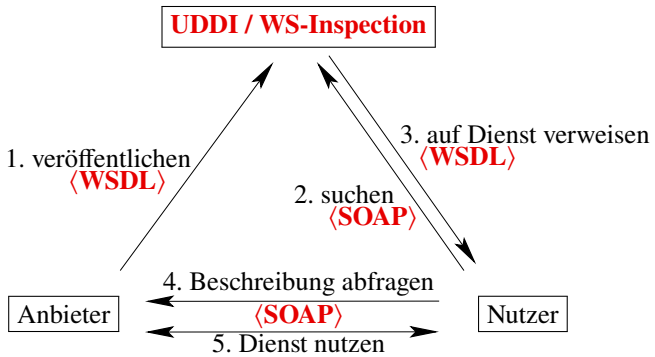


Abbildung 20.2: Das magische Dreieck des Prozessablaufs eines Webservices

W3C konkrete technische Spezifikationen als Bestandteil eines Webservices, nämlich WSDL und SOAP:

[4]Melzer (2008):§3.3.

Definition 20.3. A *Web Service* is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the web service in a manner prescribed by its description using SOAP messages. [W3C[5]] □

Die Basiskomponenten eines Webservices sind in Abbildung 20.2 illustriert und werden durch die folgenden Spezifikationen beschrieben[6]:

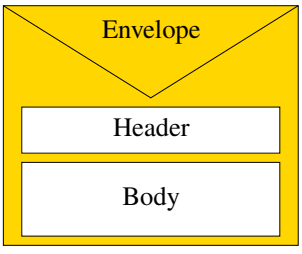
- *SOAP* – XML-basiertes Nachrichtenformat und Transportprotokoll. Es legt fest, wie eine Nachricht aufgebaut sein muss, um als SOAP-Nachricht zu gelten. SOAP ist nicht an ein Betriebssystem oder eine Programmiersprache gebunden.
- *WSDL* – XML-basierte Beschreibungssprache für Webservices.[7] Ein WSDL-Dokument beschreibt einen Webservice, der mittels SOAP erreichbar ist.
- *UDDI (Universal Description, Discovery, and Integration protocol)* – Verzeichnisdienst für Webservices. Bereits Ende 2005 kündigten IBM, Microsoft und SAP die Unterstützung für UDDI auf, in den SAP-Docs finden sich aber immer noch Hinweise darauf.[8]
- *WS-Inspection* – dezentralisierter Verzeichnisdienst.

SOAP

SOAP[9] ist ein Netzwerkprotokoll, um im Rahmen einer SOA Daten zwischen verschiedenen Systemen auszutauschen und Webdienste aufzurufen. Die detaillierten SOAP-Spezifikationen sind auf den W3C-Seiten erhältlich,

`http://www.w3.org/TR/soap12-part0/`

Die Spezifikation für den Aufbau einer SOAP-Nachricht ist die „SOAP Version 1.2 Part 1: Message Framework“. Eine SOAP-Nachricht ist demnach prinzipiell wie in Abbildung 20.3 aufgebaut, also ganz ähnlich wie ein HTML-Dokument. Allerdings besteht ein wichtiger Un-




```
<?xml version="1.0"?>
<env:Envelope
 xmlns:env="http://www.w3.org/2003/05/soap-envelope">
   <env:Header>
      ... Sicherheitsinformationen (z.B.) ...
   </env:Header>
   <env:Body>
      ... XML-Dokument als Nachricht ...
   </env:Body>
</env:Envelope>
```

Abbildung 20.3: Aufbau einer SOAP-Nachricht, nach Melzer (2008:§5.4)

terschied darin, dass `Header` und `Body` groß geschrieben werden (für XML ja relevant!). Mit der Wahl des Namensraums `xmlns` wird die Version der SOAP-Spezifikation festgelegt, auf die sich das an `xmlns` gebundene Präfix `env` im Folgenden bezieht. In diesem Fall ist der Namensraum ein URL, der zur Schemadefinition von SOAP 1.2 führt.

[5] `https://www.w3.org/TR/ws-gloss/#webservice`

[6] Melzer (2008):§4.2.

[7] `https://www.w3.org/TR/wsdl20-primer/`

[8] `https://help.sap.com/doc/saphelp_nw70/7.0.31/de-DE/e2/36a53dc1204c64e10000000a114084/content.htm`

[9] Ursprünglich stand SOAP für *Simple Object Access Protocol*, allerdings war es nie wirklich geeignet, auf *Objekte* zuzugreifen (und schon gar nicht einfach); wegen der frühen Verbreitung des Begriffs einigte man sich beim W3C darauf, dass SOAP *kein* Akronym ist (Melzer (2008):§5.1).

SOAP Header

Der erste Teil einer SOAP-Nachricht, der SOAP-Header, ist optional. Er kann maximal einmal in einer SOAP-Nachricht vorkommen, und zwar ausschließlich als erstes Kindelement des SOAP-Envelopes. Der Inhalt des SOAP-Headers ist in der SOAP-Spezifikation nicht definiert, er enthält Steuerungsinformationen, beispielsweise zur Verarbeitung der eigentlichen Nachricht. Die Kindelemente des `<env:Header>`-Elements heißen *Header-Blöcke*. In dem folgenden Quelltext gibt es beispielsweise einen Header-Block, *authentication*.

```
<env:Header>
  <m:authentication
   xmlns:m=\az http://beispiel.xy/Annahme">
   env:role="http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver"
   env:mustUnderstand=\az true\az>
     <m:passwort>
       sjkjagfg,175zgdh73jbd
     </m:passwort>
  </m:authentication>
</env:Header>
```

Alle Header-Blöcke müssen einen Namensraum spezifizieren. Ein Header-Block muss das Attribut *role* und kann die optionalen Boole'schen Attribute *mustUnderstand* und *relay* enthalten:

- Das Attribut *role* spezifiziert den Empfänger bzw. die Zwischenstation („Knoten“) der SOAP-Nachricht, der den Header-Block bearbeiten darf oder muss. Es gibt drei Standardwerte für *role*:
  - `http://www.w3.org/2003/05/soap-envelope/role/next` identifiziert eine Zwischenstation *(intermediary)*. Nach der Verarbeitung wird der Header-Block entfernt, so dass dieselbe Nachricht nicht mehrfach von einem Knoten verarbeitet wird.
  - `http://www.w3.org/2003/05/soap-envelope/role/none` bedeutet, dass dieser Knoten diese SOAP-Nachricht *nicht* verarbeiten darf
  - `http://www.w3.org/2003/05/soap-envelope/role/ultimateReceiver` identifiziert den Empfänger der SOAP-Nachricht.
- Das optionale Attribut *mustUnderstand* kann die Werte *true* oder *false* annehmen, je nachdem ob der adressierte Knoten (Empfänger bzw. Zwischenstation) den Header-Block auswerten muss oder nicht. Bei *true* wird die weitere Bearbeitung der SOAP-Nachricht sofort unterbrochen, wenn der Knoten den Header-Block nicht verarbeiten kann, und eine Fehlermeldung zurück zum Absender geschickt.
- Das optionale Attribut *relay* bestimmt, ob der Header-Block von einer Zwischenstation weitergeleitet wird oder nicht, falls der Header-Block *nicht* verarbeitet wird.

SOAP-Body

Der SOAP-Body ist ein zwingendes Element einer SOAP-Nachricht. Der Inhalt des Bodys ist die eigentlich zu übertragende Information und muss selber ein gültiges XML-Dokument sein, bis auf den Prolog, der hier nicht mehr erscheinen darf. Mit SOAP lassen sich also alle Informationen verschicken, die sich als XML-Dokument darstellen lassen. Beispiele sind HTML-Seiten, PDF-Dokumente, Verträge oder Bestellformulare.

```
<?xml version="1.0"?> \\
<env:Envelope
 xmlns:env=\az http://www.w3.org/2003/05/soap-envelope">
  <env:Body>
    <html>
      <body>
        Willkommen auf dem a href=\az haegar.fh-swf.de/\az>Hägar-Server</a>!
      </body>
    </html>
  </env:Body>
</env:Envelope>
```

SOAP-Fehler *(SOAP Fault)*

Bei einer Kommunikation können an beliebiger Stelle in der Kommunikationskette Fehler auftreten. Diesem Thema ist in Teil 1 der SOAP-Spezifikation ein eigenes Kapitel gewidmet.[10] Im Falle eines Fehlers enthält der SOAP-Body als einziges Element einen SOAP-Fehlerblock *(SOAP fault block)*, in dem sich wiederum die Elemente gemäß Tabelle 20.1 befinden. Für jedes

Element	verpflichtend	Beschreibung
`Code`	ja	Ein von der SOAP-Spezifikation festgelegter Code der Fehlerquelle, vgl. Tab. 20.2
`Reason`	ja	Textuelle Beschreibung des aufgetretenen Fehlers
`Node`	nein	Beschreibt, an welcher Stelle der SOAP-Kommunikationskette (Knoten = *node*) der Fehler aufgetreten ist
`Role`	nein	Beschreibt die Rolle des Knotens, bei dem der Fehler aufgetreten ist
`Detail`	nein	Enthält weitere Informationene zum aufgtertetenen Fehler; der Inhalt des Detailelements kann von der Anwendung frei festgelegt werden

Tabelle 20.1: Elemente des SOAP-Fehlerblocks Melzer (2008:§5.4.3)

Element existieren Festlegungen, wie es zu strukturieren und wie der Wert der Information zu übertragen ist. Das Element `Code` enthält die beiden folgenden Elemente:

- `Value` enthält den Fehlercode der Fehlermeldung und ist verpflichtend. Die möglichen Fehlercodes sind durch die SOAP-Spezifikation vorgegeben und in Tabelle 20.2 aufgelistet.
- `Subcode` ist ein optionales Element von `Code` und ermöglicht die genauere Spezifikation des Fehlercodes im `Value`-Element.

Wichtig ist, dass das Element `Subcode` rekursiv aufgebaut ist, es enthält verpflichtend wieder ein Element `Value` und optional ein Element `Subcode`. Der einzige Unterschied zum `Code`-Element ist, dass die `Value`-Elemete von `Subcode` keine Werte wie in Tabelle 20.2 vordefiniert sind.

Als zweites verpflichtendes Element eines Fehlerblocks ist das Element `Reason` vorgeschrieben. Es enthält ein oder mehrere Elemente `Text`, in denen sich jeweils eine für Menschen lesbare Beschreibung des aufgetretenen Fehlers befindet. Sie sind also nicht für eine automatische Auswertung durch Maschinen gedacht.

Je `Text`-Element sollte dieselbe Fehlerbeschreibung in einer anderen Sprache enthalten sein. Deshalb muss jedes `Text`-Element ein eindeutiges Attribut `xml:lang` mit einem wohldefinierten Wert (`"de"`, `"en"`, ...) enthalten. In Abbildung 20.4 ist ein vollständiges Beispiel einer SOAP-Fehlermeldung wiedergegegeben.

[10] `http://www.w3.org/TR/soap12-part1/#soapfault`

SOAP-Fehlercode	Beschreibung
`VersionMismatch`	Der Knoten der SOAP-Kommunikationskette erwartet eine andere SOAP-Version.
`MustUnderstand`	Ein Knoten kann ein Pflichtelement eines SOAP-Headerblocks nicht auswerten
`DataEncodingUnknown`	Es sind Datentypen aufgetreten, die nicht in eine SOAP-Nachricht übersetzt werden können.
`Sender`	Die SOAP-Nachricht konnte vom Sender nicht verarbeitet werden.
`Receiver`	Die SOAP-Nachricht konnte vom Empfänger nicht verarbeitet werden.

Tabelle 20.2: SOAP-Fehlercodes Melzer (2008:§5.4.3)

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>rpc:BadArguments</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en-US">Processing error</env:Text>
        <env:Text xml:lang="cs">Chyba zpracování</env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body>
</env:Envelope>
```

Abbildung 20.4: Beispiel für eine SOAP-Fehlermeldung

Remote Procedure Call (RPC) mit SOAP

Die allgemeine Form einer SOAP-Nachricht, also der Austausch eines XML-Dokuments zwischen zwei oder mehr Anwendungen, ermöglicht auch den Ansatz des Remote Procedure Calls (RPC). Hierbei erwartet der ursprüngliche Sender der Nachricht eine Rückantwort. Speziell für diesen RPC-Mechanismus wird in der SOAP-Spezifikation *Version 1.2 Part 2 (Adjuncts)*,

`http://www.w3.org/TR/soap12-part2/`

eine eigene Syntax festgelegt. Der RPC-Mechanismus in SOAP sieht drei Arten von SOAP-Nachrichten vor:

1. Anfrage *(Request)*: Der Anfrager ruft eine Methode des Dienstanbieters auf und übergibt dazu die geforderten Eingabeparameter in der richtigen Reihenfolge.

2. Antwort *(Response)*: Die Anfrage des Anfragers konnte vom Anbieter fehlerfrei bearbeitet werden und das Ergebnis wird von ihm an den Anfrager zurück geschickt.

3. Fehler *(Fault)*: An irgendeiner Stelle des RPC-Aufrufs ist ein Fehler aufgetreten, dessen SOAP-Fehlermeldung statt einer Rückantwort an den Anfrager geschickt wird.

Als Beispiel betrachten wir eine in Java verfasste Methode, die ein Dienstanbieter als Webservice verfügbar machen will. Abbildung 20.5 zeigt eine Java-Klasse, mit der ein Anbieter den Dienst

```
public class Auftragsannahme {
   /** Führt die Bestellung durch und gibt das Lieferdatum zurück.*/
   public String bestellen(String artikelnummer) {
      ...
      return lieferdatum;
   }
}
```

Abbildung 20.5: Beispiel für eine Service Methode des Anbieters in Java

`Auftragsannahme` bereitstellt, der wiederum eine Methode `bestellen` zur Verfügung stellt, mit der ein Dienstnutzer eine Bestellung anhand der Artikelnummer durchführen kann. Die Methode führt alle Schritte zur Bestellabwicklung durch und gibt das Lieferdatum als String zurück. Mit der SOAP-Anfrage in Abbildung 20.6 wird die Methode aufgerufen. In dem SOAP-Header wird angezeigt, dass die Bestellung im Rahmen einer größeren Transaktion, der Auftragsabwicklung, abläuft. Der SOAP-Body enthält den eigentlichen RPC-Request. Gemäß der Spezifikation muss das erste Element des SOAP-Bodys den Namen der Methode übernehmen, die aufgerufen werden soll (`bestellen`), zugleich muss ein Namensraum definiert werden. In diesem Element befinden sich dann in der richtigen Reihenfolge die Elemente der Eingabeparameternamen und deren Werte, im Beispiel `<m:artikelnummer>123456789</m:artikelnummer>`.

Woher weiß die Laufzeitumgebung, welche Klasse gemeint ist, deren Methode `bestellen` aufgerufen werden soll? Die Klasse ist der *Service-Endpunkt (service endpoint)*. Nach der SOAP-Spezifikation ist die Identifikation des Service-Endpunktes eine Aufgabe des Transportprotokolls, und nicht der SOAP-Nachricht selbst. In diesem Fall wird der Service-Endpunkt im Namensraum spezifiziert. Entsprechend muss die Laufzeitumgebung des Dienstanbieters selbst dafür sorgen, dass der angegebene Service-Endpunkt auch erreicht wird, also hier beispielsweise die Klasse `Auftragsannahme` über ein Servlet.

Die Antwort, die *SOAP-Response*, ist ganz ähnlich aufgebaut wie die zugehörige Anfrage, vgl. Abbildung 20.7. Doch wird an den Namen der Methode nun das Wort `Response` angehängt, so

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
  <env:Header>
    <t:transaction
     xmlns:t="http://thirdparty.example.org/Auftragsabwicklung"
     env:encodingStyle="http://example.com/encoding"
     env:mustUnderstand="true">
  </env:Header>
  <env:Body>
    <m:bestellen
     xmlns:m="http://haegar.fh-swf.org/Auftragsannahme"
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <m:artikelnummer>123456789</m:artikelnummer>
    </m:bestellen>
  </env:Body>
</env:Envelope>
```

Abbildung 20.6: Beispiel für ein SOAP-Request der Methode *bestellen*

```
<?xml version="1.0"?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
    xmlns:rpc="http://www.w3.org/2003/05/soap-rpc">
  <env:Header>
    <t:transaction
     xmlns:t="http://thirdparty.example.org/Auftragsabwicklung"
     env:encodingStyle="http://example.com/encoding"
     env:mustUnderstand="true">
  </env:Header>
  <env:Body>
    <m:bestellenResponse
     xmlns:m="http://haegar.fh-swf.org/Auftragsannahme"
     env:encodingStyle="http://www.w3.org/2003/05/soap-encoding">
      <rpc:result>m:lieferdatum</rpc:result>
      <m:lieferdatum>2009-04-11</m:lieferdatum>
    </m:bestellenResponse>
  </env:Body>
</env:Envelope>
```

Abbildung 20.7: Beispiel für ein SOAP-Response der Methode *bestellen*

dass klargestellt ist, dass es sich um die Rückantwort der Methode `bestellen` handelt. Es können im Prinzip mehrere Daten zurückgeliefert werden, jedoch muss die Rückgabe der Methode als Element `<rpc:result>` gekennzeichnet werden. Die Aufrufparameter sind in der Antwort natürlich nicht mehr enthalten.

Im Falle eines Fehlers im Ablauf der Anfrage wird statt der SOAP-Antwort eine SOAP-Fehlermeldung zurückgeliefert.

20.2 REST

20.2.1 REST als Programmierprinzip

REST als Abkürzung für *representational state transfer* ist ein Programmierprinzip, gemäß dem alle Informationen in einem Netzwerk verknüpft sind und daher als Ressourcen mit einem URI zu versehen und ausschließlich über die HTTP-Methoden GET, POST, PUT und DELETE manipulierbar sind. REST stellt damit eine schlanke Alternative zu SOAP über WSDL oder zu RPC dar.

Der Begriff REST wurde von Roy Fielding in seiner Dissertation[11] im Jahr 2000 als Architekturstil für das Web geprägt. *„The name «Representational State Transfer» is intended to evoke an image of how a well-designed Web application behaves: a network of web pages (a virtual state-machine), where the user progresses through the application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.“* [12] [13] Wie in Abbildung 20.8 skizziert,

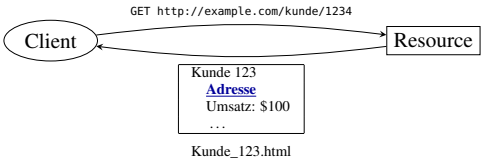


Abbildung 20.8: Representational State Transfer, oder „Transfer von Zustandsrepräsentationen“

ruft ein Client eine Resource im Web mit einem URI auf und erhält eine *Repräsentation* der Resource, beispielsweise ein HTML-Dokument.[14] Diese Repräsentation wiederum verändert den *Zustand* der Clientanwendung und ermöglicht durch Hyperlinks den Aufruf weiterer Ressourcen (in Abbildung 20.8: „Adresse“). Somit transferiert die Clientanwendung den Zustand mit dessen Resourcenrepräsentation, daher *representational state*. Fielding sah REST als allgemeines Ar-

[11] Fielding (2000):§5.

[12] Übersetzt etwa: „Der Name «Transfer von Zustandsrepräsentationen» soll ein Bild hervorrufen, wie eine gut gestaltete Webapplikation sich verhalten sollte: ein Netzwerk von Webseiten (ein virtueller Zustandsautomat), in dem der Anwender sich durch Auswahl von Links (Zustandsübergänge) durch die Applikation bewegt, so dass ihm die nächste Seite (die Repräsentation des nächsten Zustands der Anwendung) zur weiteren Nutzung übertragen wird.“

[13] Fielding (2000):§6.1.

[14] „REST components perform actions on a resource by using a representation to capture the current or intended state of that resource and transferring that representation between components. A representation is a sequence of bytes, plus representation metadata to describe those bytes. [. . .] A representation consists of data, metadata describing the data, and, on occasion, metadata to describe the metadata (usually for the purpose of verifying message integrity). Metadata is in the form of name-value pairs, where the name corresponds to a standard that defines the value's structure and semantics. Response messages may include both representation metadata and resource metadata: information about the resource that is not specific to the supplied representation. [. . .] The data format of a representation is known as a media type.“ (Fielding (2000):§5.2.1.2)

chitekturprinzip für das Web und war an dessen Realisierung Mitte und Ende der 1990er Jahre beteiligt, insbesondere als Architekt des Protokolls HTTP/1.1.

Ein Grundsatz von REST ist, dass URI's sowohl die einfachsten als auch die wichtigsten Elemente des Web sind. Ein URI stellt hierbei nicht nur ein Dokument dar, sondern ein *Konzept*[15], er muss also nicht unbedingt eine *physikalische* Ressource adressieren, sondern kann auch eine *logische* sein, beispielsweise das Ergebnis einer Datenbankabfrage. URI's sollten sich entsprechend so selten wie möglich ändern, auch wenn die Zustände der referenzierten Ressourcen dahinter sich ändern. Der URI sollte außerdem möglichst die *Semantik*, also die Bedeutung der Ressource ausdrücken, so wie in Abbildung 20.8 der URI

```
http://example.com/kunden/123
```

die Ressource der Informationen zu Kunde 123 im Unternehmen example.com ist. Mit anderen Worten wird also der Zustand der Ressource dem Client dargestellt und verändert so *dessen* Zustand, wie in Abbildung 20.9 skizziert. In jeder Ressourcenrepräsentation können sich weitere

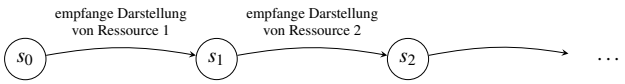


Abbildung 20.9: Änderungen der Zustände s_i eines Clients durch Transfers von Zustandsrepräsentationen (*representational states*) von Ressourcen.

URI's in Form von Hyperlinks befinden, die ein *logisches* Netzwerk bilden, das nicht nur die *physischen* Ressourcen des Webs enthält.

Die Operationen einer REST-Architektur zur Zustandsänderung bestehen nur aus den vier HTTP-Methoden POST, GET, PUT, und DELETE[16] Hierbei gelten die Konventionen:

- PUT erzeugt eine Ressource; falls sie bereits exisitiert, wird sie neu angelegt, also logisch aktualisiert.
- GET liefert die angebene Ressource in einer bestimmten Repräsentation, in der Regel als HTML-Dokument.
- POST übermittelt Daten an die angegebene Ressource zur weiteren Verarbeitung.
- DELETE löscht die angegebene Ressource.

Die vier HTTP-Methoden repräsentieren damit die vier grundlegenden Operationen „CRUD" für Datenstrukturen, also *Create*, *Read/Retrieve*, *Update* und *Delete*. Oft wird als weitere Update-Methode neben PUT auch PATCH verwendet:

- PATCH aktualisiert die angegebene Ressource teilweise.

20.2.2 Missachtungen von REST im Web: Caches und Cookies

„Communication must be stateless in nature [. . .], such that each request from client to server must contain all of the information necessary to understand the request, and cannot take advantage of any stored context on the server. Session state is therefore kept entirely on the client. This constraint induces the properties of visibility, reliability, and scalability. Visibility is improved

[15] Fielding (2000):§6.2.1.
[16] de Vries (2022):§1.6, Tabelle 1.2.

URI	PUT	GET	POST	DELETE
Datenstruktur, z.B. `http://example.com/kunden/`	Fügt ein neues Element in die Datenstruktur ein	Listet die URIs der Elemente der Datenstruktur auf	Aktualisiert die Datenstruktur	Löscht die gesamte Datenstruktur
Element, z.B. `http://example.com/kunden/123`	Sieht die adressierte Ressource als eigene Datenstruktur und erzeugt ein neues Element	Zeigt eine Repräsentation des Elements	Aktualisiert das Element oder fügt es neu ein	Löscht das Element

Tabelle 20.3: Verwendung der HTTP-Methoden für Datenstrukturen (Listen, Datenbanken, . . .) und ihre Elemente gemäß REST.

because a monitoring system does not have to look beyond a single request datum in order to determine the full nature of the request. Reliability is improved because it eases the task of recovering from partial failures [. . .]. Scalability is improved because not having to store state between requests allows the server component to quickly free resources, and further simplifies implementation because the server doesn't have to manage resource usage across requests. Like most architectural choices, the stateless constraint reflects a design trade-off. The disadvantage is that it may decrease network performance by increasing the repetitive data (per-interaction overhead) sent in a series of requests, since that data cannot be left on the server in a shared context. In addition, placing the application state on the client-side reduces the server's control over consistent application behavior, since the application becomes dependent on the correct implementation of semantics across multiple client versions."[17]

Dem Prinzip der zustandslosen Kommunikation widersprechen sowohl Caches, die am Client oder im Web Repräsentationen speichern, als auch Cookies[18]. Ein Problem ist, dass ein Cookie einen Zustand speichert, den der Anwender beispielsweise durch die Zurücktaste des Browsers zwar ändert, der im Cookie nicht synchronisiert wird und bei einem späteren Aufruf der betreffenden Seite ignoriert wird. Cookies erlauben zudem die Nachverfolgung des Aufrufverhaltens des Anwenders und stellen damit ein Risiko für den Datenschutz dar. Nach REST dürfte der Zustand nur auf Clientseite gespeichert und auch nur für ihn abrufbar sein. Entprechend sollte nach REST ein virtueller Einkaufswagen nicht auf Cookies basieren, um ihn serverseitig zu identifizieren, sondern der Anwender sollte die URIs der selektierten Artikel clientseitig in einer Liste speichern und diese beim Abschluss des Einkaufs mit einem URI an den Server senden.[19]

20.2.3 Konventionen für REST-konforme Programmierung

Die folgende Auflistung ist ein Auszug der Best Practices von Roger Costello.[20]

1. Gib jeder Ressource, also jeder Information, die sichtbar ist oder werden kann, einen URI.

2. Bevorzuge stets logische gegenüber physikalischen URI's, also lieber

```
http://example.com/kunden/123
```

[17] Fielding (2000):§5.1.3.
[18] Fielding (2000):§6.3.4.
[19] Fielding (2000):§6.3.4.2.
[20] `http://www.xfront.com/sld059.htm`

als

```
http://example.com/kunden/123.html
```

Ein logischer URI ermöglicht die Änderung der Implementierung einer Ressource, ohne auch die Clientanwendung ändern zu müssen.

3. Verwende Nomen in einem URI, nicht Verben. Denn Ressourcen sind „Dinge“, nicht „Aktionen“.

4. Verwende die HTTP-Methoden nur gemäß REST wie auf Seite 66 beschrieben. Programmiere insbesondere jedes GET nebeneffektfrei (*side-effect free*), so dass jeder GET-Aufruf sicher im Sinne der ursprünglichen Definition (siehe §1.8.4) ist. Andererseits sollte ein POST keinen Lesezugriff implementieren.

5. Verwende möglichst Hyperlinks in einer HTTP-Response. So werden Abfrageergebnisse mit anderen Ressourcen verknüpft und lassen dem Anwender Freiheiten zum weiteren Vorgehen. Durch Links entsteht eine schrittweise Entfaltung von Information für den Anwender. Eine Antwort ohne Links ist endgültig, sie stellt in dem logischen Netzwerk von Informationen (Ressourcen) einen Endknoten dar.

6. Verwende einen Schrägstrich „/“, um Eltern-Kind-Knoten oder Ganzes-Teile-Beziehungen darzustellen.

20.2.4 RESTful Webservices

Da Webserver üblicherweise nur die HTTP-Methoden GET und POST ermöglichen, müssen PUT und DELETE durch serverseitige Software und über POST ermöglicht werden. APIs, die das liefern, heißen RESTful. Beispiele:

- Node.js `https://nodejs.org` mit dem Paket `express` `http://expressjs.com/`
- JAX-RS (Java API for RESTful Web Service) `https://jax-rs-spec.java.net/`
- Jersey `https://jersey.java.net/`: Auf JAX-RS aufsetzende API, die eine REST-konforme Programmierung vereinfachen will.
- Slim `http://www.slimframework.com/`: Eine RESTful API für PHP
- Restler `https://www.luracast.com/products/restler`: Eine RESTful API für PHP
- Symfony `http://symfony.com/`: allgemeine API für PHP, das auch REST-konforme Programme ermöglicht; eine schlankere Variante ist Silex `http://silex.sensiolabs.org/`.

Weitere RESTful PHP-APIs sind unter `http://davss.com/tech/php-rest-api-frameworks/` aufgelistet. Bei dem Java-API JAX-RS wird über die Annotationen @POST, @GET, @PUT und @DELETE die Verbindung eines HTTP-Requests zu der entsprechenden Webservice-Methode auf der JVM des Servers bestimmt.

21

NoSQL: Big Data und verteilte Datenbanken

Kapitelübersicht

In den letzten Jahren ist „Big Data“ zu einem Megabegriff geworden, der viele Betrachtungen über digitale Ökonomie und über Informatik dominiert. Allerdings ist nicht klar definiert, was mit Big Data eigentlich genau gemeint ist. Übereinstimmung besteht jedoch darin, dass es sich um extrem große Datenmengen handelt, die an die Grenzen der technisch möglichen Kapazitäten von Speichergröße und Verarbeitbarkeit stoßen[1]. Was aber ist der ökonomische Wert von großen Datenmengen? Im Grundsatz ist es die Annahme, mit geeigneten Methoden aus Daten wertvolle Information zu gewinnen, wobei eine größere Datenmenge ein größeres Wertschöpfungspotenzial impliziert. Wir werden hier diese Grundannahme nicht in Frage stellen, sie hat auf den ersten Blick ihre Berechtigung, obschon ihre ethischen und datenschutzrechtlichen Auswirkungen sicher diskussionswürdig sind.

Beispiel 21.1. *(Vestas)*[2] Das dänische Unternehmen *Vestas Wind Systems* ist einer der weltweit führenden Hersteller von Windkraftanlagen. Das Unternehmen verwendet Big Data zur Standardoptimierung neuer Windräder und bietet seinen Kunden damit eine wertvolle Dienstleistung an: Auf Basis von 160 Faktoren u.a. über Temperatur, Feuchtigkeit, Niederschläge,

[1] Wrobel et al. (2015).
[2] Dillerup und Stoi (2013):S. 776.

Windrichtungen, Gezeiten oder Satellitendaten sowie der Leistungsdaten und Laufzeiten seiner installierten Windräder kann *Vestas* für jeden beliebigen Standort der Erde abschätzen, wieviel Wind dort in den nächsten Jahrzehnten geerntet werden kann. Für *Vestas* stellt diese datenbasierte Unterstützung zur Standortplanung seiner Kunden einen wichtigen Wettbewerbsvorteil dar. □

Relationale Datenbanken eignen sich aufgrund ihrer zeilenorientierten Tabellenschemata nicht ideal für eine verteilte Datenspeicherung. Ein relationales Datenbankschema mit den Attributen ID, Name und Alter wird beispielsweise in der Form

```
1, Peter, 42, 2, Paul, 18, 3, Mary, 36
```

gespeichert, also in Datensätzen. Mit *vertikaler Skalierung (scale-up)*, also einer Speichererweiterung oder CPU-Steigerung des Datenbankservers, ist ein relationales Datenbanksystem für höhere Last effizient aufrüstbar. Allerdings stößt dieses Konzept an seine Grenzen, einerseits wenn die Datenmenge zu groß wird und schnelle Datenzugriffe nicht mehr möglich werden, andererseits wenn zu viele Nutzer auf die Datenbank zugreifen und sich gegenseitig blockieren. Eine *horizontale Skalierung (scale-out)*, also eine Systemerweiterung durch zusätzliche Rechner wie bei einer Cloud-Umgebung ist dagegen mit klassischen relationalen Datenbanken nicht effizient zu erreichen[3]. Mit dem Konzept der horizontalen Skalierung, also einer verteilten

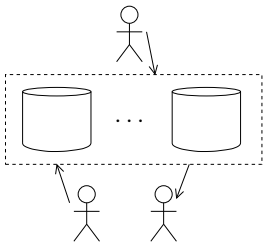


Abbildung 21.1: Verteilte Datenbank (horizontale Skalierung) mit mehreren Nutzern.

Datenbank, ergeben sich jedoch durch parallelen Zugriff auf gemeinsam genutzte Daten einige logische und praktische Probleme, mit deren Erkennen und Lösungen wir uns in den folgenden Absätzen beschäftigen wollen.

In den letzten Jahren konnten sich in diesem Zuge eine ganze Reihe von andersartigen Datenspeicherungskonzepten etablieren, insbesondere *schemalose Datenbanken*. Sie werden üblicherweise unter dem Begriff NoSQL-Datenbank zusammengefasst. Wegweisend für die Durchsetzung dieser Technologien waren speziell Google mit BigTable und MapReduce, Facebook mit Cassandra und Amazon mit SimpleDB und DynamoDB. Insgesamt stellt NoSQL keine einheitliche oder genau definierte Technik dar, sondern ergänzt relationale Datenbankkonzepte um für gewisse Anwendungsfälle besser geeignete Modelle[4].

Entsprechend vielfältig und heterogen ist auch das Angebot an NoSQL-Datenbanken. Um Bewertungskriterien für den Einsatz einer geeigneten SQL- oder NoSQL-Datenbank in einem gegebenen Anwendungsfall zu erlangen, ist zunächst das Verständnis des CAP-Theorems notwendig. Mit ihm können Vor- und Nachteile der Datenbanken strukturiert und somit deren geeigneten Einsatzgebiete umrissen werden.

[3] Edlich et al. (2010):§8.4.

[4] Edlich (2013); Spichale und Wolff (2013).

21.1 Das CAP-Theorem

Warum eignen sich relationale Datenbanken nicht für eine horizontale Skalierung, also für eine verteilte Datenbankspeicherung? Die Ursache liegt in der zentralen Rolle, die die *Konsistenz* der gespeicherten Daten bei parallelen Zugriffen auf die Datenbank spielt. Das folgende Beispiel zeigt die fatalen Folgen, die Dateninkonsistenzen haben können.

Beispiel 21.2. *(Dateninkonsistenz)* Gegeben sei Konto mit Kontostand 0 € und zwei Transaktionen, die gleichzeitig 1000 € bzw. 50 € überweisen möchten. Betrachte dann folgendes Anweisungsszenario:

Zeitpunkt	Aktion	Kontostand
1.	Überweisung von Transaktion 1 beginnt	0 €
2.	Überweisung von Transaktion 2 beginnt	0 €
3.	Berechnung von Transaktion 1 beendet	1000 €
4.	Berechnung von Transaktion 2 beendet	50 €

Jede Transaktion beendet für sich betrachtet korrekt, aber am Ende fehlen 1000 € auf dem Konto! □

Für geschäftskritische Daten oder Finanzdienstleistungen ist die Konsistenz der Daten wesentlich. In vielen Fällen, gerade im Zusammenhang mit sozialen Netzen, ist sie allerdings weniger relevant als die Verfügbarkeit und die Zugriffszeit bei hoher Belastung durch eine große Anwenderzahl. Ein mathematisches Resultat über die den Zielkonflikt zwischen Datenkonsistenz, Verfügbarkeit und Partitionstoleranz (also Ausfallsicherheit einzelner Partitionen) liefert das CAP-Theorem. Eine pragmatische Abschwächung des Konsistenzmodells, die den Zielkonflikt des CAP-Theorems umgeht, ist BASE. Beides betrachten wir in diesem Abschnitt.

21.1.1 CAP

Die für das CAP-Theorem zentralen Begriffe sind Konsistenz, Verfügbarkeit und Partitionstoleranz eines verteilten Datenbanksystems. Sie stellen Eigenschaften dar, die jede Datenbank erfüllen sollte, egal ob klassisch relational oder nicht. Allerdings wird sich herausstellen, dass nicht alle drei gleichwertig erfüllt werden können, und somit unterscheiden sich die verschiedenen Datenbanktypen in der Priorisierung der Eigenschaften.

Consistency C: *Konsistenz* bedeutet, dass eine Datenbank unter allen Umständen und jederzeit einen konsistenten Zustand der Daten bzw. eine perfekte Datenintegrität garantiert, also einen Zustand wie in Beispiel 21.2 stets vermeidet. Datenkonsistenz impliziert also eine starke Restriktion für *Transaktionen*: Entweder wird eine Transaktion *komplett* ausgeführt oder gar nicht[5]. Insbesondere ist also ein verteiltes Datenbanksystem mit mehreren replizierenden Knoten nur dann konsistent, wenn nach einer Transaktion nachfolgende Lesezugriffe stets den aktualisierten Wert zurückgeben. Insbesondere bedeutet dies, dass der geänderte Wert erst dann gelesen werden kann, wenn *alle* replizierenden Knoten aktualisiert sind. Bei Störungen der Verbindungen der Knoten untereinander wird also eine einmal begonnene Transaktion nicht ausführt.

Availability A: *Verfügbarkeit* einer Datenbank bezeichnet eine stets schnelle und für die gegebene Anwendung akzeptable Reaktionszeit auf eine Datenbankanfrage. In einem verteilten Datenbanksystem kann die Verfügbarkeit durch Redundanz erhöht werden, so dass bei Ausfall einzelner Knoten andere Knoten auf eine Datenbankabfrage antworten können.

[5] Kurowski (2012):S. 13.

Partition tolerance P: *Partitionstoleranz* eines verteilten Datenbanksystems bedeutet, dass der Ausfall eines Knotens oder einer Kommunikationsverbindung zwischen Knoten einer verteilten Datenbank nicht zum Ausfall des gesamten Systems führt. Wenn beispielsweise in einem System Knoten A und B die gleichen Dienste anbieten und beide nach außen sichtbar sind, aber es durch eine Verbindungsstörung sie nicht miteinander kommunizieren können und Nachrichten verloren gehen, muss ein partitionstolerantes System dennoch weiterhin einwandfrei funktionieren[6].

21.1.2 Das Theorem

Satz 21.3 (**CAP-Theorem**). *In einem verteilten Datenbanksystem können nur zwei der drei Eigenschaften C, A, P erfüllt sein, nicht aber alle drei.*

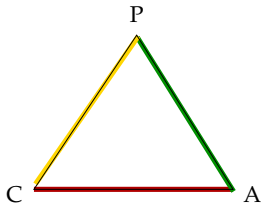


Abbildung 21.2: Das CAP-Theorem: Es kann nur die Paare C–A, C–P oder A–P geben.

Beweis. Betrachten wir eine aus zwei Knoten N1 und N2 bestehende verteilte Datenbank. Die Knoten stellen hierbei Replikationen der gleichen Datenbank im Zustand D0 dar. Es sei zunächst die Ausfallsicherheit unverzichtbar. Sei Knoten N1 dann (ohne Beschränkung der Allgemeinheit) zuständig für Schreiboperationen auf D0, und Knoten N2 für das Lesen der Daten, wie in Abb. 21.3 illustriert. Wird nun die Datenbank D0 durch eine Schreiboperation

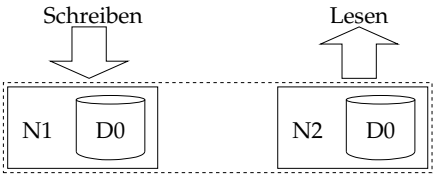


Abbildung 21.3: Schema der Schreib- und Leseprozesse einer verteilten Datenbank.

in den Zustand D1 geändert, so wird im Normalfall eine Synchronisation durch die Nachricht M von N1 zum Knoten N2 in Gang gesetzt, die dessen Datenbankreplikation ebenfalls in den Zustand D1 versetzt, wie in Abb. 21.4 dargestellt. Fällt jedoch die Kommunikation zwischen

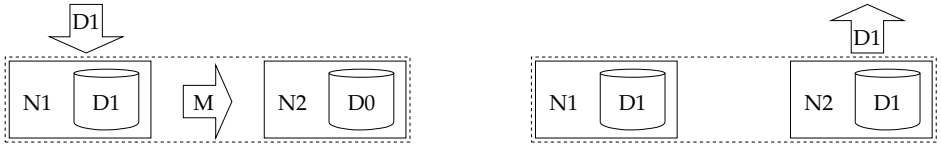


Abbildung 21.4: Synchronisation modifizierter Daten.

den Knoten N1 und N2 durch eine Störung aus, wird die Datenbank im Knoten N2 nicht synchronisiert (Abb. 21.5). Hat die Datenbank als Priorität die Konsistenz der Daten, so muss in diesem Fall auf die Verfügbarkeit verzichtet werden (C–P). Spielt dagegen die Datenkonsistenz

[6] Kurowski (2012):S. 13.

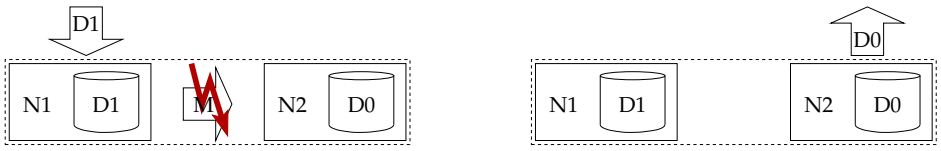


Abbildung 21.5: Dateninkonsistenz bei gestörter Verbindung zwischen Knoten.

eine geringere Rolle als die Verfügbarkeit und die Ausfallsicherheit (A–P), so kann der Zustand in Abb. 21.5 akzeptiert werden. Verlangen wir schließlich Konsistenz und Verfügbarkeit (C–A), so kann es sich nicht um ein verteiltes System handeln, sondern entgegen Abb. 21.3 muss Knoten N2 entfallen und N1 Lesezugriff ermöglichen; damit ist jedoch die Ausfallsicherheit nicht mehr gegeben. □

Die nach dem CAP-Theorem möglichen Kombinationen bedeuten also für ein gegebenes verteiltes Datenbanksystem konkret:

- CA – Die Daten sind zu jedem Zeitpunkt konsistent auf allen Knoten, die online sind, und man kann stets den konsistenten Zustand lesen und schreiben. Allerdings muss bei einer Verbindungsstörung zwischen zwei Knoten das gesamte System blockieren, die Knoten können nicht mehr synchronisiert werden, d.h. das System ist nicht partitionstolerant.
- CP – Die Daten sind jederzeit konsistent auf allen Knoten, und von Verbindungsstörungen innerhalb des Systems betroffene Knoten werden vom Netz genommen und sind damit nicht verfügbar.
- AP – Die Knoten sind stets verfügbar, also auch bei Verbindungsstörungen. Allerdings kann keine Garantie für die jederzeitige Datenkonsistenz auf allen verfügbaren Knoten gegeben werden.

Das CAP-Theorem wurde als Vermutung von Eric Brewer auf einem Symposium im Jahr 2000 vorgetragen und von Seth Gilbert und Nancy Lynch 2002 bewiesen.[7] Domain Name Server (DNS) oder Cloud-Computing-Systeme sind typische Beispiele für verteilte A–P-Datenbanken,

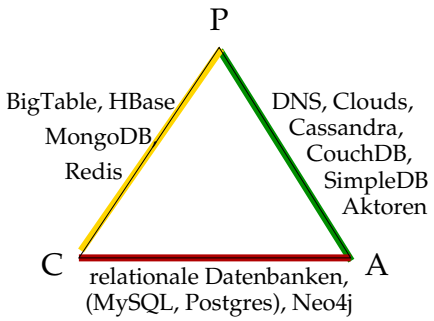


Abbildung 21.6: Typische Anwendungssysteme gemäß dem CAP-Theorem. Nach `https://blog.nahurst.com/visual-guide-to-nosql-systems` [2021-09-21]

relationale Datenbanken sind vom Typ C–A, und Online-Bankanwendungen sind vom Typ C–P, siehe Abbildung 21.6.[8]

[7] `http://blog.codecentric.de/2011/08/grundlagen-cloud-computing-cap-theorem/` [2012-07-08]

[8] Die Rolle von Neo4j ist etwas umstritten; ursprünglich wurde es als CA-System entwickelt und konnte gar nicht verteilt gespeichert werden, die Version Neo4j HA dagegen verwendet Cache Sharding und scheint auf die jederzeitige Konsistenz zu verzichten und ist damit ein AP-System. (`jim.webber.name/2011/02/scaling-neo4j-with-cache-sharding-and-neo4j-ha/` [2013-09-11])

21.2 Typen von NoSQL-Datenbanken

Bei den meisten Webanwendungen sind Verfügbarkeit und Ausfallsicherheit für den erfolgreichen Betrieb wichtiger als strenge Datenkonsistenz. Solche Anwendungen benötigen daher horizontal skalierbare, also verteilte Datenbanken vom Typ A–P. Natürlich würden diese Systeme nicht funktionieren, wenn sie auf Konsistenz der Daten vollständig verzichten würden. Tatsächlich wird das strikte Konsistenzmodell ACID *(atomicity, consistency, integrity, durability)* der relationalen Datenbanken bei verteilten Datenbanken üblicherweise durch das schwächere Konsistenzmodell BASE *(basically available, soft state, eventually consistent)* verwendet[9]. Daraus ergibt sich die folgende Definition für NoSQL-Datenbanken.

Definition 21.4. Eine *NoSQL-Datenbank* ist ein Datenbanksystem, das schemafrei ist oder nur schwache Schemarestriktionen hat. Insbesondere ist sie nicht relational. Daneben hat eine NoSQL-Datenbank oft auch die folgenden Eigenschaften[10][11]:

- Die Datenbank ist horizontal skalierbar, also verteilt. (Die horizontale Partitionierung einer Datenbank kann durch *Sharding*[12] auf die einzelnen als *Shards*, also „Scherben“, „Splitter“, bezeichneten Serverknoten geschehen; oder nach Amazons *Dynamo-Modell* über eine Aufteilung der Datenreferenzen gemäß dem konsistenten Hashing[13], siehe §21.4 auf S. 78.)
- Das Konsistenzmodell der Datenbank ist BASE, nicht ACID.
- Die Datenbank ist quelloffen.

Nach dem CAP-Theorem ist eine NoSQL-Datenbank mit den letzten drei Eigenschaften also entweder vom Typ A–P, d.h. ausfallsicher (P) und hoch verfügbar (A), oder vom Typ C–P, also ausfallsicher (P) und konsistent (C). □

Abhängig von ihrer Datenlogik können NoSQL-Datenbanken in die Typen der dokumentbasierten Datenbanken, der Schlüssel-Wert-Speicher *(Key-Value Stores)*, der spaltenorientierten und der Graphdatenbanken eingeteilt werden.

21.2.1 Spaltenorientierte Datenbanken

Eine der möglichen Alternativen zu relationalen Datenbanksystemen sind *spaltenorientierte Datenbanken*, also Datenbanken, die zwar auch ein Tabellenschema haben, deren Daten jedoch nach ihren Spalten sortiert gespeichert sind:

```
1, 2, 3, Peter, Paul, Mary, 42, 18, 36
```

Dieses Speicherkonzept ist auch unter dem Begriff *Wide Column Stores* bekannt und hat Effizienzvorteile bei der Datenanalyse, der Datenaggregation und der Datenkompression. Das Einfügen, Suchen oder Löschen von Datensätzen dagegen ist aufwendiger als bei einer reihenorientierten Speicherung. Joins sind damit zwar prinzipiell möglich, erfordern bei großen Datenmengen jedoch hohe Laufzeiten[14]. Wichtige Datenbanken, die partiell spaltenorientierte

[9] Edlich et al. (2010):§2.2.3.
[10] `http://nosql-database.org` [2012-10-17]
[11] Edlich et al. (2010):§1.2.
[12] `http://www.codefutures.com/database-sharding/`, `http://www.scalebase.com/products/database-sharding/` [2012-10-17]
[13] Wolff (2013).
[14] Kurowski (2012):S. 23.

Speicherkonzepte verwenden, sind BigTable von Google, bzw. die quelloffene Variante HBase, Cassandra von Facebook und SimpleDB von Amazon[15]. SAP stellt mit HANA eine In-Memory-Datenbank zur Echtzeit-Reporting seiner ERP-Systeme bereit[16].

21.2.2 Dokumentbasierte Datenbanken

Die Daten einer *dokumentenbasierten Datenbank* werden in Dokumenten gespeichert. Ein *Dokument* ist hierbei eine Ansammlung von verschiedenen Objekten, also dynamisch strukturierten Werten. Typischerweise werden die Dokumente in semistrukturierten Formaten wie JSON oder BSON (binärem JSON) gespeichert[17], aber auch in XML. Die bekanntesten Vertreter dieses Datenbanktyps sind CouchDB und MongoDB[18],[19].

21.2.3 Schlüssel-Wert-Speicher *(Key-Value Stores)*

Schlüssel-Wert-Speicher (Key-Value Stores) sind im wesentlichen assoziative Arrays. Im Gegensatz zu dokumentbasierten Datenbanken gibt es für Schlüssel-Wert-Speicher keine Möglichkeit, Daten über den Schlüssel zu suchen.[20] Bekannte Systeme dieses Typs sind Riak, Redis und MemCacheDB.

21.2.4 Graphdatenbanken

In einer *Graphdatenbank*, oder *graphenorientierten Datenbank*, werden effizient Knoten und deren Beziehungen untereinander gespeichert. Typische Anwendungsfälle solcher Datenbanken sind das Speichern von Netzwerken, seien es soziale Netzwerke, Straßennetze oder Busfahrpläne. Bekannte Graphdatenbanken sind Neo4j, FlockDB von Twitter oder Pregel von Google[21].

Um bei sehr großer Anzahl zu speichernder Knoten und Kanten ($\gtrsim 10^{10}$) die Daten horizontal auf mehrere Rechner zu verteilen, stößt man auf die Schwierigkeit, einen Graphen effizient zu partitionieren. Es gibt in der Mathematik jedoch kein exaktes Verfahren, das einen allgemeinen zusammenhängenden Graphen in zwei oder mehrere gleichgroße Teilgraphen zerlegt[22]. Üblicherweise ist eine Graphdatenbank daher nicht oder nur rudimentär horizontal skalierbar, solange sie die tiefe Traversierung von Graphen ermöglicht.[23]

21.3 MapReduce

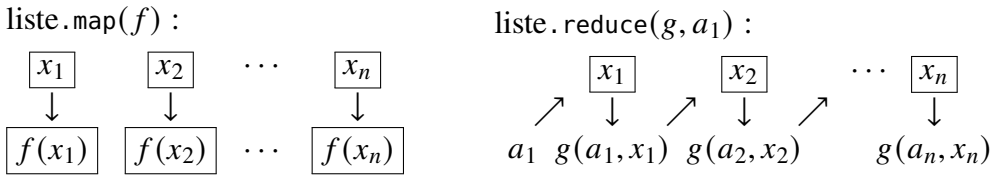
In einer nichtrelationalen Datenbank ist SQL als Abfragesprache nicht mehr verwendbar. Insbesondere steht die `SELECT`-Anweisung zur bedingten Abfrage von Dateninhalten nicht mehr zur Verfügung. Da einerseits für Datenmengen der Größenordnung mehrerer Terabytes („Big Data“)

[15] Edlich et al. (2010):§3.
[16] Kessler et al. (2014):§1.3.3.
[17] Gürsoy (2012).
[18] Edlich et al. (2010):§4.
[19] Kurowski (2012):§1.2.2.
[20] Kurowski (2012):§1.2.2.
[21] Edlich et al. (2010):§6.
[22] Edlich et al. (2010):§6.1.5.
[23] So scheint Neo4j echte Partitionierung in viele Shards nicht zu ermöglichen, während FlockDB dies mit Traversierung nur der direkten Nachbarschaft eines Knoten erlaubt, vgl. Ayende Rahien (2010): `http://ayende.com/blog/4490/that-no-sql-thing-scaling-graph-databases` [2012-10-23]. Neo4j ist demnach eine Datenbank vom Typ C–A, FlockDB vom Typ A–P.

horizontal skalierbare Datenbanken notwendig und daher klassische relationale Datenbanken nicht verwendbar sind, entwickelte Google 2004 den Algorithmus MapReduce.[24]

21.3.1 Map und Reduce in der funktionalen Programmierung

MapReduce setzt sich zusammen aus zwei Routinen, die ihren Ursprung in der funktionalen Programmierung haben, `map` und `reduce` oder `fold`. In der funktionalen Programmierung wenden beide Routinen eine übergebene Funktion `f` auf die Elemente einer Liste an und aggregiert deren Auswertungen zu einem Endergebnis. Während `map` die Funktion auf jedes Element anwendet und somit eine Liste mit derselben Größe zurückgibt, ergibt `reduce` einen einzelnen Rückgabewert, in der Regel eine Zahl. Die Funktionsweise von `map` und `reduce` bei ihrer Anwendung auf die Liste `liste` $= [x_1, x_2, \ldots, x_n]$ und die Funktion $f(x)$ bzw. die Funktion $g(a, x)$ und den Startwert a_1 ist durch folgendes Schema illustriert:



Hierbei erwartet `reduce` eine zweiparametrige Funktion $g(a, x)$, deren erster Parameter a den bis jetzt errechneten Wert darstellt und x den aktuell einzusetzenden Listenwert. In der Illustration ist der Startwert a_1, und für die folgenden Werte gilt

$$a_{n+1} = g(a_n, x_n) \tag{21.1}$$

Ein Beispiel in JavaScript zeigt `map`, mit der Funktion $f(x) = x^2$ angewendet auf eine Liste (eigentlich: ein Array) von Zahlen:

```
var reihe = [1, 2, 3, 4, 5];
var f = function(x) {return x*x;}
var quadrate = reihe.map(f);
document.write(quadrate);        // 1,4,9,16,25
```

Ein typisches Beispiel für `reduce` ist die Berechnung der Summe aller Listenelemente:

```
var g = function(a, x) { return 2*x - 1 + a; }  // a: bisher berechneter Wert
var summe = [1, 2, 3, 4, 5].reduce(g, 0);
document.write("<br>" + summe);  // 25
```

Die Berechnung einer Summe wie $\sum_{i=1}^{3} i^3$ lässt sich mit `map` und `reduce` wie folgt programmieren:

```
var f = function(i) {return i*i*i;}
var g = function(a, x) { return x + a; }  // a: bisher berechneter Wert
var liste = [1, 2, 3];

var aggregat = liste.map(f).reduce(g, 0);

document.write("<br>" + aggregat);  // 36
```

[24] `http://research.google.com/archive/mapreduce.html` [2013-09-11]

21.3.2 Arbeitsweise von MapReduce

Das von Google entwickelte MapReduce verarbeitet verteilte Daten hauptsächlich in zwei Arbeitsphasen, der Map-Phase und der Reduce-Phase. Der Anwender kann hierbei seine Funktionen auf verteilten Daten in einem Rechnercluster o.ä. anwenden. Um die komplizierten Details der Daten- und CPU-Verteilung braucht sich das Anwenderprogramm dabei nicht zu kümmern, dies wird von der eingebundenen MapReduce Bibliothek durchgeführt. Der Anwender muss nur die beiden Funktionen `map` und `reduce` implementieren[25], wobei `map` zwei String-Parameter `key` und `value` erwartet und eine Liste von Schlüssel-Wert-Paaren erstellt,

```
map(key,value) ⟶ list<key2,value2>
```

und `reduce` einen String `key` und eine Liste als Parameter benötigt und eine Liste von Werten erstellt:[26]

```
reduce(key2, list<value2>) ⟶ list<key3,value3>
```

Der Datenfluss bei einer Anwendung von MapReduce ist in Abb. 21.7[27] skizziert. In einem

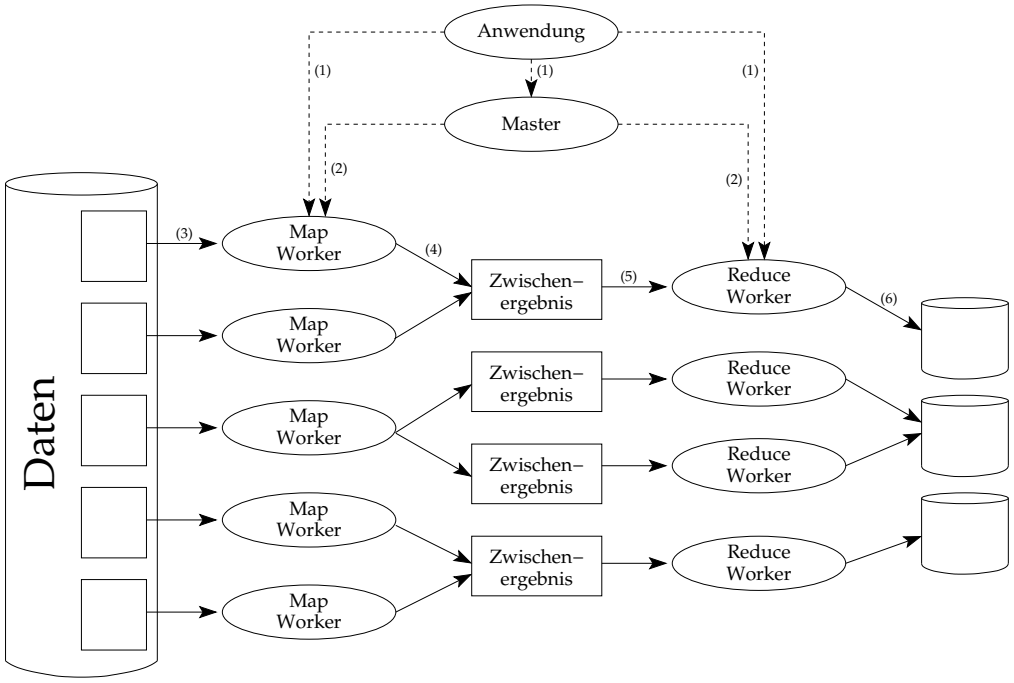


Abbildung 21.7: Arbeitsphasen und Datenfluss von MapReduce. Modifiziert nach[28]

ersten Schritt (1) teilt MapReduce die Eingabedateien in Teile der Größe 16–64 MB auf und Kopien des Anwenderprogramms werden auf mehreren Rechnern innerhalb des Rechnerclusters gestartet. Im zweiten Schritt (2) übernimmt eine Kopie des Programm, der *Master*, spezielle Steuerungsaufgaben, während die anderen Kopien des Programms, die *Worker*, die von ihm zugewiesenen map- und reduce-Aufgaben durchführen. Die Map Worker lesen im nächsten Schritt (3) die ihnen jeweils zugewiesenen Teile der Eingabedatei, verarbeiten die `(key, value)`-Paare gemäß der vom Anwender implementierten Funktion `map` und speichern danach (4) ihre Resultate in Tupeln `(key, list)` als Zwischenergebnisse, wobei deren Speicheradressen dem Master

[25] Edlich et al. (2010):§2.1.2.

[26] Java API Doc für Release 1.0.3: `http://hadoop.apache.org/common/docs/r1.0.3/api/` [2012-06-30], Klassen `Mapper` und `Reducer` im Paket `org.apache.hadoop.mapreduce`

[27] `http://code.google.com/intl/en/edu/parallel/mapreduce-tutorial.html` [2012-06-29]

mitgeteilt werden. In Schritt (5) werden den Reduce Workern diese Adressdaten übermittelt, so dass sie auf die Zwischenergebnisse zugreifen, sie anhand des Schlüssels gruppieren und die akkumulierten Ausgabewerte in eine Ausgabedatei speichert (6). Nachdem alle Worker ihre Aufgaben beendet haben, wird die Steuerung vom Master wieder an das Anwendungsprogramm übergeben.

21.4 Konsistentes Hashing

Eine Hashfunktion oder Streuwertfunktion ist eine Funktion $h\colon V \to H$ von einer Menge V von Wörtern möglichst gleichverteilt auf eine endliche Menge $H \subset \mathbb{N}_0$ von Hashwerten abbildet. Die Berechnung eines Wertes $y = h(w)$ für ein Wort $w \in V$ ist dabei effizient, meist höchstens von der Zeitkomplexität $O(|w|)$, wobei $|w|$ die Länge des Wortes bezeichnet. Hashfunktionen werden bei der Übertragung von Daten zur Prüfsummenberechnung und in der Kryptographie zur Integrität einer Nachricht eingesetzt.

Ein weiteres Einsatzgebiet für Hashfunktionen ist die Datenspeicherung. Die Idee ist hierbei, einem Objekt w abhängig von dessen Wert eine Speicherort mit der Adresse $h(w)$ zuzuweisen. In verteilten Systemen ist der Hashwert $h(w)$ in der Regel ein fester Speicherort, Slot oder Server aus einer Menge von n Speicherorten $\{S_1, \ldots, S_n\}$. In verteilten Systemen ist jedoch üblicherweise die Anzahl der Speicherorte nicht konstant, sondern ändert sich dynamisch, beispielsweise durch Systemausfälle oder Netzwerkfehler, aber auch durch hinzukommende Speicherknoten. Generell erzwingt bei dem klassischen Hashverfahren eine Änderung der möglichen Speicherorte eine Ersetzung der Hashfunktion, da sich ja nun der Wertebereich der Funktion geändert hat. Ein weiteres praktisches Problem der Datenspeicherung in verteilten Systemen ist die ungleiche Speicherkapaizität der Speicherslots. Eine Hashfunktion würde idealerweise die statistische

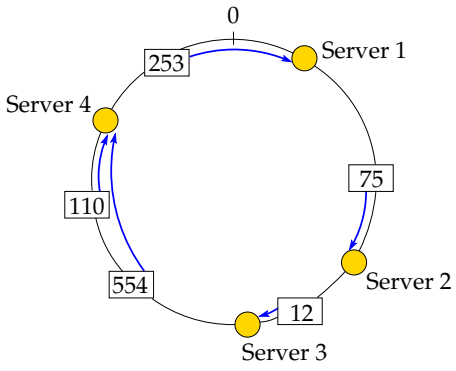


Abbildung 21.8: Abbildung von Hashwerten (Rechtecke) auf Server (Kreise). Nach[29]

Verteilung der Hashwerte an die Verteilung der Speicherkapaizitäten anpassen, also kleine Slots seltener belegen als größere.

Das *konsistente Hashing* löst beide Probleme, also die dynamische Veränderung der Speicherorte und die ungleiche Verteilung der Speicherkapazitäten, auf elegante Weise. Kernidee ist hierbei, dass ein Speicherort (Server) für einen Bereich von Hashwerten zuständig ist, nicht für einen einzelnen Hashwert. Die Zuständigkeit für die Bereiche ist dabei zyklisch organisiert, siehe Abbildung 21.8. Wird nun ein Server hinzugefügt oder entfernt (Abbildung 21.9), so hat dies nur Auswirkungen auf die Zuordnung der Hashwerte in der unmittelbaren Umgebung des dem Server zugeordneten Bereichs. Ein neuer Server übernimmt dabei einfach Hashwerte und die Speicherung der entsprechenden Objekte seines Nachfolgers. Verlässt dagegen ein Server den Ring des konsistenten Hashings, so werden alle ihm zugeteilten Objekte auf seinen Nachfolger kopiert. Die Anpassung an die Speicherkapazitäten der einzelnen Server wird üblicherweise

durch ein Einfügen von virtuellen Servern erreicht, wobei ein Server mit hoher Speicherkapazität entsprechend viele virtuelle Server bereitstellt. Auf diese Weise kann die durch die Hashfunk-

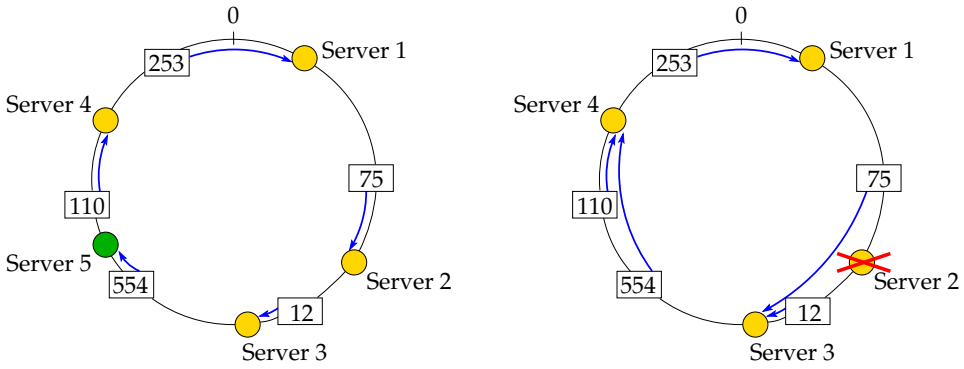


Abbildung 21.9: Konsistentes Hashing bei Hinzufügen und Entfernen eines Servers. Nach[30]

tion gegebene statistische Verteilung der zu speichernden Objekte an die Speicherkapazitäten angepasst werden. Es gibt mehrere Implementierungen des konsistenten Hashings im Umfeld der NoSQL-Datenbanken.[31]

21.5 Vektoruhren

In verteilten Systemen werden Daten von parallelen Prozessen geschrieben und verarbeitet. Dabei entsteht häufig das Problem, dass Daten nachträglich zu synchronisieren sind. Eine Synchronisation anhand der realen physikalischen Zeit ist bei nebenläufigen Prozessen nicht immer ausreichend, denn die rein zeitliche Abfolge sagt nicht notwendig etwas über den ursächlichen Zusammenhang von Ereignissen aus. In Abbildung 21.10 ist beispielsweise ein System mit drei

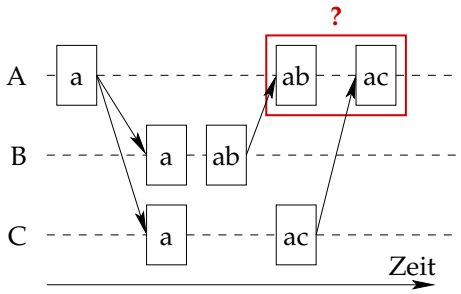


Abbildung 21.10: Nachrichten in einem System mit drei Prozessen (Aktoren, s.u.).

nebenläufigen Prozessen dargestellt, in dem ein Prozess jedem anderen spontan eine Nachricht schicken kann. Die von A empfangene Nachricht `ac` hat dabei keinen kausalen Zusammenhang mit der zuvor versendeten Nachricht `ab`, obwohl diese zeitlich davor versendet wurde.

21.5.1 Kausalität

Kausalität bezeichnet eine Beziehung zwischen zwei Ereignissen. Die Kausalität setzt notwendig eine zeitliche Richtung voraus. Das zeitlich frühere Ereignis heißt *Ursache*, das spätere *Wirkung*. Kausalität gibt damit eine Verknüpfung von Vergangenheit, Gegenwart und Zukunft wieder.[32] In der klassischen Physik beispielsweise werden die Ursachen von Bewegungen Kräfte

[31] Edlich et al. (2010):§2.3.
[32] von Weizsäcker (1985):S. 301.

genannt.[33] Gemäß der Relativitätstheorie ist ein *Ereignis* ein Punkt in der vierdimensionalen Raumzeit, also ein Geschehen, das zu einer bestimmten Zeit an einem bestimmten Ort im Raum stattfindet. Der Begriff der Kausalität wird in der Relativitätstheorie zwar nicht definiert, jedoch

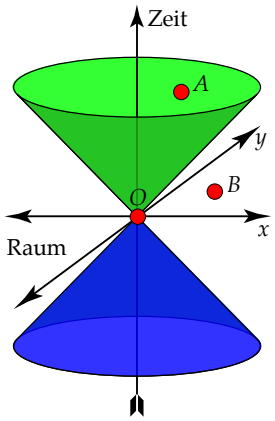


Abbildung 21.11: Lichtkegel eines Ereignisses O gemäß der relativistischen Physik. Der dreidimensionale Raum (x, y, z) ist in dem Diagramm der Anschaulichkeit halber auf zwei Dimensionen (x, y) reduziert. Das Ereignis O kann hier Ursache des Ereignisses A sein, nicht aber von B.

ist dort die präzise Bedingung bestimmt, wann zwei Ereignisse kausal abhängig sein *können*. Wie in Abbildung 21.11 dargestellt, sind alle Ereignisse, für die ein gegebenes Ereignis O Ursache sein kann, innerhalb seines Zukunftslichtkegels oder auf seiner Oberfläche, während alle Ereignisse, die Ursache von O sein können, sich im Innern oder auf der Oberfläche des Vergangenheitslichtkegels befinden. Den Rand des Lichtkegels, also dessen Oberfläche, bilden dabei diejenigen Photonen, die in die verschiedenen Raumrichtungen fortschreiten, im Vergangenheitslichtkegel auf O zu, im Zukunftslichtkegel von O fort. Die Bedingung, dass sich kausal zusammenhängende Ereignisse innerhalb des Lichtkegels oder auf seiner Oberfläche befinden müssen, drückt also geometrisch die Tatsache aus, dass sich keine Wirkung schneller als Licht ausbreiten kann.[34]

Hume charakterisiert Kausalität als eine aus der Erfahrung abgeleitete Beziehung, die im Verstand entsteht, nicht aber den Ereignissen selber angehört.[35] Nach Kant gehört Kausalität zu einer der vier Kategorien des reinen Verstandes.[36] In der Physik werden allgemein die Naturgesetze als Differenzialgleichungen nach der Zeit dargestellt und drücken somit einen kausalen Gesetzesbegriff aus: Die zu einer Zeit vorliegenden Kräfte bestimmen die *Änderungen* des Zustands eines Systems. Das Gesetz bestimmt alle *möglichen* Bewegungen, die *Anfangsbedingungen* legen fest, welche Bewegung wirklich stattfindet. Nach Wigner war es Newtons größte Leistung, den Unterschied von Gesetz und Anfangsbedingungen entdeckt zu haben[37].

Kritik. Über die richtige Definition der Kausalität besteht in der Physik allerdings Unsicherheit[38]. Die klassische Physik charakterisiert Kausalität in der Form: „Ist der Zustand eines

[33] von Weizsäcker (1992):S. 816.

[34] Für weitere Details siehe Landau und Lifschitz (1997):S. 8ff; Penrose (1995):S. 275ff; de Vries (1994):Def. 1.19 & Satz 1.20; für Aspekte der Kosmologie siehe Börner (2003):§1.2.

[35] "*Cause* and *effect* are relations, of which we receive information from experience, and not from any abstract reasoning or reflexion." (Hume (1896):S. I, III, I) "We define a cause to be, *An object precedent and contiguous to another, and so united with it in the imagination, that the idea of the one determines the mind to form the idea of the other, and the impression of the one to form a more lively idea of the other;*" (Hume (1896):S. I, III, XIV)

[36] `https://korpora.zim.uni-duisburg-essen.de/Kant/aa03/134.html` [2021-09-21]

[37] von Weizsäcker (1985):S. 243.

[38] von Weizsäcker (1992):S. 831ff.

abgeschlossenen Systems in einem Zustand gegeben, so ist der Zustand in jedem früheren oder späteren Zeitpunkt eindeutig bestimmt“, also in einem eindeutigen funktionalen und deterministischen Zusammenhang von Ereignissen. Unser Bewusstsein macht jedoch einen Unterschied zwischen Ursache und Wirkung, also zwischen Vergangenheit und Zukunft.

Ferner sind Naturvorgänge soweit und nur soweit kausal, als man sie als von der Beobachtung unabhängige Abläufe beschreiben kann. In der Quantenmechanik gibt es jedoch wegen des Einflusses der Messung eines physikalischen Systems durch den Beobachter die Freiheit des Zufalls[39]. Die Gültigkeit der Kausalität wird durch das EPR-Paradox sogar ernsthaft in Frage gestellt[40]. Nach Bohr sind Raum-Zeit-Beschreibung und Kausalität nur in der klassischen, d.h. nichtquantenmechanischen, Physik vereinbar.[41]

Das Kausalgesetz. Das *Kausalprinzip* ist ein philosophisches Prinzip, wonach jedes Ereignis eine Ursache hat.[42] Aus dem Kausalprinzip lässt sich das Kausalgesetz ableiten:

Satz 21.5 (**Kausalgesetz**). *Jedes Ereignis hat eine Ursache und ist selbst wiederum Ursache für andere Ereignisse. Ferner haben gleiche Ursachen stets die gleiche Wirkung.*

Beweis. Kant (1787, `https://korpora.zim.uni-duisburg-essen.de/Kant/aa03/166.html` ff) □

Nach Kant gehört das Kausalgesetz zu den Prinzipien des reinen Verstandes.[43][44]

21.5.2 Kausalität und Nebenläufigkeit

Da ein Ereignis in der Regel nicht durch ein einziges Ereignis als hinreichende Bedingung verursacht wird („Monokausalität“), sondern ein Ereignis im Gegenteil oft mehrere Ursachen hat, führte der australische Philosoph John Mackie 1974 die INUS-Bedingung ein[45]. INUS steht für „insufficient, but necessary part of an unnecessary but sufficient condition“. Eine Ursache für ein Ereignis ist nach diesem Konzept genau ein Ereignis, welches nicht hinreichender, aber notwendiger Teil einer Bedingung ist, die wiederum für die Wirkung nicht notwendig, aber hinreichend ist. In Abbildung 21.12 beispielsweise sind zwei Ereignisse P und Q skizziert, die beide allein nicht hinreichend für Ereignis O sind, während die Bedingung $P \wedge Q$ hinreichend für O sind,

$$P \wedge Q \Rightarrow O, \qquad \text{aber} \qquad P \not\Rightarrow O, \quad Q \not\Rightarrow O.$$

Nach Definition des logischen UNDs sind sowohl P als auch Q notwendig für die Bedingung $P \wedge Q$.

---

[39] Penrose (1995):S. 271.

[40] de Vries (2012c):§2.3.

[41] von Weizsäcker (1985):S. 522.

[42] von Weizsäcker (1992):S. 816, siehe auch `https://spektrum.de/lexikon/philosophie/kausalgesetz/1052`.

[43] „Das Schema der Ursache und der Kausalität eines Dinges überhaupt ist das Reale, worauf, wenn es nach Belieben gesetzt wird, jederzeit etwas anderes folgt. [...] Das Schema der Gemeinschaft (Wechselwirkung), oder der wechselseitigen Kausalität der Substanzen in Ansehung ihrer Akzidenzen, ist das Zugleichsein der Bestimmungen der Einen, mit denen der Anderen, nach einer allgemeinen Regel. Das Schema der Möglichkeit ist die Zusammenstimmung der Synthesis verschiedener Vorstellungen mit den Bedingungen der Zeit überhaupt (z. B. da das Entgegengesetzte in einem Dinge nicht zugleich, sondern nur nacheinander sein kann,) also die Bestimmung der Vorstellung eines Dinges zu irgendeiner Zeit. Das Schema der Wirklichkeit ist das Dasein in einer bestimmten Zeit. Das Schema der Notwendigkeit ist das Dasein eines Gegenstandes zu aller Zeit. [...] Die Schemate sind daher nichts als Zeitbestimmungen a priori nach Regeln, und diese gehen nach der Ordnung der Kategorien, auf die Zeitreihe, den Zeitinhalt, die Zeitordnung, endlich den Zeitinbegriff in Ansehung aller möglichen Gegenstände.“ `https://korpora.zim.uni-duisburg-essen.de/Kant/aa03/138.html` [2021-09-21]

[44] von Weizsäcker (1985):S. 510.

[45] Mackie (1974):§3.

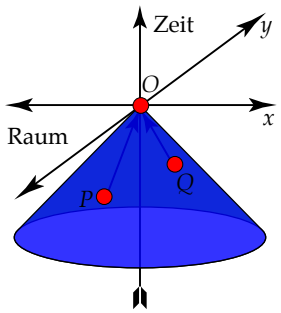


Abbildung 21.12: Multikausalität: Das Ereignis O hat zwei Ursachen P und Q.

Definition 21.6. Zwei Ereignisse, die einander weder Ursache noch Wirkung sind, heißen *nebenläufig (concurrent)* oder *kausal unabhängig*. □

In der Relativitätstheorie ist ein Ereignis innerhalb des Lichtkegels zu einem Ereignis außerhalb des Lichtkegels stets nebenläufig. Zwei Ereignisse innerhalb des Lichtkegels können dagegen kausal voneinander abhängen.

Ein in der Informatik bei Nebenläufigkeit gebräuchliches Konzept ist das *Aktorenmodell*. Ein *Aktor* ist ein eigenständiger Prozess, der an andere Aktoren Nachrichten versenden und von ihnen Nachrichten empfangen kann. Ein Aktor speichert dabei empfangene Nachrichten in seiner Mailbox, einer Warteschlange, und arbeitet sie nach seinen Kriterien ab. Eine typische Anwendung nach dem Aktorenmodell ist ein E-Mail-System. Da beim Aktorenmodell jeder Aktor seinen eigenen Speicher verwaltet, braucht es – im Gegensatz zu einem System mit einem zentralen Speicher *(shared memory)* – keine Synchronisationen zur Herstellung konsistenter Daten zu geben. Aktoren kommunizieren also *asynchron*.

Um bei der Kommunikation zwischen Aktoren wie in Abbildung 21.10 kausale Zusammenhänge überhaupt erkennen zu können, müssen die Nachrichten nach einer Idee von Lamport[46] einen Zeitstempel beinhalten, der den Zeitpunkt ihrer Entstehung dokumentiert. Solch ein Zeitstempel $t\colon \mathscr{A} \to \mathbb{R}$ auf der Menge $\mathscr{A}$ der Ereignisse oder Nachrichten heißt *Lamport-Uhr*, wenn er im Zeitverlauf streng monoton steigend ist, also die *schwache Uhrenkonsistenzbedingung* erfüllt ist[47]: Ist Ereignis A Ursache von Ereignis B, so ist der Zeitstempel von A kleiner als der von B,

$$A \to B \quad \Rightarrow \quad t(A) < t(B). \tag{21.2}$$

Eine Lamport-Uhr kann also die physikalische Zeit angeben, z.B. die UNIX-Systemzeit, oder einfach ein Zähler sein. Eine Lamport-Uhr erfüllt die *starke Uhrenkonsistenzbedingung*, wenn auch umgekehrt

$$t(A) < t(B) \quad \Rightarrow \quad A \to B. \tag{21.3}$$

gilt, wenn also aus der Tatsache, dass der Zeitstempel von A kleiner als der von B ist, auch stets folgt, dass A die Ursache von B ist. Die starke Konsistenzbedingung kann in nebenläufigen Systemen mit einer global geltenden Uhr wie einer Lamport-Uhr nicht erfüllt sein, wie das in Abbildung 21.10 skizzierte Gegenbeispiel zeigt. In Systemen mit nur einem einzelnen Prozess dagegen ist die strenge Konsistenzbedingung mit Lamport-Uhren stets erfüllt.

Vektoruhren hängen jeder Nachricht die ID ihres Absenders und dessen individuelle Lamport-Uhr an, in der Regel ein inkrementeller Zähler. Die Sender-ID kann dabei eine Prozess-ID, eine MAC-Adresse oder eine IP-Nummer sein. Vektoruhren lösen damit das Problem der

[46] Leslie Lamport ist ein Informatiker, der neben seinen Arbeiten über Uhren in nebenläufigen Systemen das TEX-System von Donald Knuth zu dem LATEX erweitert hat, mit dem auch der vorliegende Text erstellt wurde.

[47] Edlich et al. (2010):§2.5.

strengen Konsistenz, indem man das System auf die von einem einzelnen individuellen Prozess empfangenen und versendeten Nachrichten beschränkt, den Begriff der Konsistenz am Ende also nur auf logisch einprozessige Systeme anwendet. In Abbildung 21.13 ist ein Szenario skizziert,

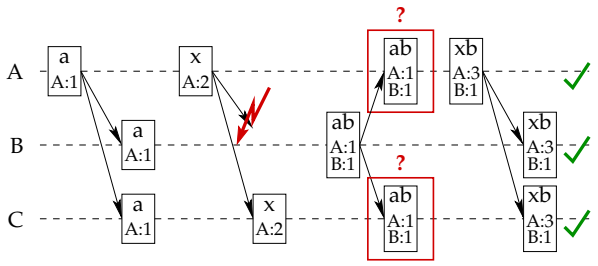


Abbildung 21.13: Nachrichtenaustausch zwischen drei Aktoren mit Vektoruhren. Im zweiten Schritt ist die Kommunikation zwischen A und B gestört, dennoch kann die Konsistenz wiederhergestellt werden.

in dem ein Aktor A erst eine Nachricht `a` an die anderen Aktoren des Systems sendet und sich im zweiten Schritt mit `x` korrigiert. Leider kann Aktor B die korrigierte Nachricht nicht empfangen und schickt daher an alle eine Nachricht mit dem veralteten Datenbestand. Allerdings können anhand der Vektoruhr sowohl A als auch C den aktuellen Stand rekonstruieren und das System wieder in einen streng konsistenten Zustand bringen.

Teil V

Digitale Ökonomie

Today, after more than a century of electric technology, we have extended our central nervous system in a global embrace

Marshall McLuhan 1964[a]

[a] McLuhan (1964 (reissued 2001)):Introduction.

22

Die digitale Revolution

Kapitelübersicht

Zum Ende der 1990er Jahre kam der Begriff der „New Economy“ auf, in der durch die neuen Kommunikationsmedien die bisherigen Grundlagen der Wirtschaft sich ändern und die Erzeugung, Verarbeitung und Verbreitung von Information und Content, also digitalen Gütern, in den Vordergrund treten. Mit der „New Economy“ verband man damals die Erwartungen in eine rasante Aufwärtsentwicklung von Internetunternehmen, die allerdings mit dem Absturz des „Dotcom“-Booms im Jahr 2000 endete, als die Aktienkurse weltweit einbrachen. Die „New Economy“ ist seitdem als Begriff eher negativ besetzt.

Dennoch: Das Internet hat unsere alltäglichen Handlungen bereits heute umgewälzt: Wir erledigen Geldgeschäfte mittlerweile zumeist mit Online-Banking, holen uns vor einem Autokauf Information aus dem Web. Für den Handel wird das Internet als weiterer Vertriebskanal zunehmend wichtiger, Handelsplattformen wie Amazon und eBay haben neue Märkte erschlossen, indem sie als digitale Kaufhäuser für Bücher oder Gebrauchtwaren auftreten. Noch dramatischer sind die Veränderungen für Märkte mit immateriellen Gütern, beispielsweise für den Musikmarkt, wo Dank der vom Fraunhofer Institut patentierten MP3-Technik die klassische Produktion auf materiellen Tonträgern wie CDs stark zurückgegangen ist. So war *Crazy* von der Gruppe Gnarls Barkley die historisch erste Single, die am 2. April 2006 allein aufgrund von Downloads Platz 1 der britischen Single Charts erlangte.[1]

[1] `http://news.bbc.co.uk/1/hi/entertainment/4870150.stm` [2011-11-18]

22.1 Drei Entwicklungen

Zur Zeit treffen drei historische Entwicklungen aufeinander, die die Lebens- und Arbeitsbedingungen unserer Gesellschaft tiefgreifend verändern werden:

- *Technische Entwicklung: Web 2.0, Big Data und vernetzte Mobilität.* Seit den 2000er Jahren werden interaktive Web-Techniken („Web 2.0“) eingesetzt, die die Erstellung und Modifizierung von im Netz gespeicherten und allgemein einsehbaren Dokumenten ermöglichen. Parallel verbreiteten sich die mobilen Technologien auf Basis immer komplexerer Endgeräte wie Smartphones. Beide Entwicklungen ihrerseits ermöglichen in Kombination, dass detaillierte individuelle Daten („Big Data“) aus immer mehr Lebensbereichen gesammelt und gespeichert werden können.

- *Demografische Entwicklung: Generation Y.* Es wächst eine Generation heran, die seit ihrer Jugend mit dem Internet und mobilen Kommunikationstechniken vertraut ist und sich zunehmend weltweit vernetzt.

- *Ökonomische Entwicklung: Globalisierung.* Bedingt durch eine zunehmende Arbeitsteilung auch über Grenzen einzelner Staaten hinweg, verteilen und entwickeln sich Lieferketten und Warenströme weltweit. Die theoretische Grundlage für die Arbeitsteilung bildet hierbei das Außenhandelsmodell von Ricardo[2], und die Auswirkungen auf die Arbeitsmärkte erklärt das Modell von Heckscher und Ohlin, demgemäß es egal ist, ob Arbeitskräfte über Grenzen wandern oder Produkte, in denen die Arbeitskraft von Menschen steckt[3],[4].

Durch diese Entwicklungen auf verschiedenen gesellschaftlichen Bereichen hat eine Digitalisierung der Gesellschaft begonnen, die zu einer neuen technischen Revolution führen dürfte, der Automatisierung des Denkens[5]: Mit „intelligenten“ Algorithmen können mit Big Data Prognosen über individuelles Verhalten ebenso wie über gesellschaftliche Trends und darauf basierend optimierte Handlungsentscheidungen berechnet werden.

In den folgenden Abschnitten werden diese Entwicklungen näher untersucht und anhand von Beispielen gezeigt, wie sie eine zunehmende Digitalisierung des Wirtschaftslebens bewirken. Hauptthese dieses Kapitels ist es, dass wir uns am Beginn einer wirtschaftlichen Umwälzung befinden, der Entwicklung einer *Digitalen Ökonomie*, deren Auswirkungen auf Mensch und Gesellschaft vergleichbar sein werden mit denjenigen der großen Wirtschaftsrevolutionen der Vergangenheit, nämlich der Handelsrevolution und der Industriellen Revolution.

Auch wenn die zukünftigen Auswirkungen und Ausprägungen der Digitalen Ökonomie heute noch nicht absehbar sind, so soll doch eine Sensibilisierung und Intuition vermittelt werden, mögliche Indizien, Chancen und Risiken dafür zu identifizieren. In diesem Sinne ist ein Vergleich mit den vergangenen Umwälzungen hilfreich zur Beantwortung einiger Kernfragen. Gibt es gemeinsame Prinzipien, die diesen historischen Entwicklungen zugrunde liegen, folgten sie bestimmten Gesetzen, kurzum: gibt es eine „ökonomische Mechanik“? Eines der wichtigen „Hebelgesetze“, die die Entwicklungen zumindest teilweise zu erklären vermag, ist das Coase'sche Gesetz, das den Transaktionskosten bei der Produktion von Waren oder Dienstleistungen eine Schlüsselrolle zuweist.

[2] Bofinger (2007):§3.
[3] Bofinger (2007):§25.
[4] P. R. Krugman und Obstfeld (2009):§4.
[5] Kurz und Rieger (2013):§14.

Welche sozialen Folgen der Digitalen Ökonomie kann man bereits identifizieren, und kann man sie extrapolieren? Soziale Netze beeinflussen zunehmend die Kommunikation und das Zusammenleben, das Internet wird als Vertriebskanal und Marketingmedium genutzt, Online-Spiele verbreiten sich mit hohen Profitraten, nicht zuletzt durch das Angebot virtueller Güter („virtuelle Ökonomie“).

Inwieweit oder ob überhaupt sich diese Tendenzen zu zukünftigen Schlüsselindustrien entwickeln werden, ist heute nicht vorhersagbar. Aber dass sie neuartig sind und ein gewaltiges Innovations- und Entwicklungspotenzial besitzen, bleibt ein faszinierendes Faktum und lässt erahnen, dass wir uns in einer Zeit des Aufbruchs befinden könnten.

22.2 Kurze Wirtschaftsgeschichte

22.2.1 Der Niedergang Europas in der Spätantike

Das Römische Reich stieg bis zu seiner Blütezeit im ersten und zweiten Jahrhundert n. Chr. zu einer Europa und den Mittelmeerraum beherrschenden Großmacht auf. Im Jahre 164 hatte es etwa 58 Millionen Einwohner (32 Millionen in Europa, 14 Millionen in Vorderasien, und 12 Millionen in Nordafrika)[6]. Seine Wirtschaft basierte einerseits auf Handel sowohl innerhalb der Grenzen als auch mit den Nachbarn in Nordeuropa und Westasien, andererseits aber auch auf Sklaverei,[7] Plünderungen und militärischer Kontrolle. Insgesamt war die Gesellschaft stark militarisiert, jeder erwachsene Mann musste einen Wehrdienst von 16 Jahren in der Infanterie oder zehn Jahren in der Kavallerie ableisten[8].

Dieses komplexe und durch das Militär gestützte Staats- und Wirtschaftssystem zerfiel ab dem dritten Jahrhundert zunehmend: Im Jahr 285 teilte sich das Reich in Westrom und Ostrom, Westrom konnte dem zunehmenden Druck insbesondere der Völkerwanderungen aus dem Norden und Nordosten, bei wachsenden Finanzierungsproblemen für das Militär, nicht standhalten und brach im Jahre 476 endgültig zusammen[9]. Ostrom dagegen wiedererstarkte im sechsten Jahrhundert sogar, konnte aber letztendlich der islamischen Expansion der Araber nicht standhalten und degenerierte ab 800 zu einem Rumpfstaat, der sich endgültig 1453 mit der Einnahme von Konstantinopel durch die Türken auflöste.

Für Europa bedeutete diese Entwicklung zwischen dem fünften und elften Jahrhundert einen wirtschaftlichen und kulturellen Niedergang. Die Städte wurden zunehmend zu Festungen, der Handel mit Nordafrika und Asien kam nahezu zum Erliegen, feudale Landwirtschaft, Klerus, Rittertum und allgemein niedriges Bildungsniveau prägten die Gesellschaft. Um 800 gab es in Europa keine Münzprägung mehr, Zinsen waren verboten und der Warenimport aus dem Orient war versiegt[10]. Erst durch die umfassenden Staats- und Bildungsreformen Karls des Großen[11] und die Formung großräumiger Staaten an Stelle der dezentralen und oft durch Plünderungen wirtschaftenden Gauen und Stammesorganisationen in den Gebieten um Nord- und Ostsee trat ein allmählicher Wandel ein. Dennoch war um das Jahr 1000 im gesamten europäischen Gebiet des ehemaligen römischen Reichs die reale Wirtschaftsleistung pro Kopf geringer als um Christi Geburt, während sie in Westasien im selben Zeitraum gestiegen war[12].

---

[6] Maddison (2007):S. 37.

[7] Etwa 20% der Arbeitsleistung des Römischen Reichs um 14 n. Chr. wurde von Sklaven verrichtet, auf dem Gebiet des heutigen Italiens sogar etwa 55% (Maddison (2007):S. 50).

[8] Maddison (2007):S. 15.

[9] Maddison (2007):S. 31.

[10] Maddison (2007):S. 77.

[11] `http://de.wikipedia.org/wiki/Karl_der_Gro\T1\sse`

[12] Maddison (2007):S. 59, 192.

22.2.2 Die Handelsrevolution um das Jahr 1000

„Im Frühen Mittelalter, das heißt etwa vom 7. bis 10. Jahrhundert, solange in Europa der Grundbesitz dominierte, gab es weder Bank- noch Handelskompanien. Gesellschaft und Wirtschaft waren . . . primitiv: Handel betrieben nur *mercatores*, die einzeln oder in Kaufmannszügen von einer Messe und einer Burg zur nächsten zogen, um eine Vielzahl von Waren feilzubieten. [. . .] In einer Welt ohne Mobilität, in der die Mehrzahl der Menschen an die Erde gebunden und von einem Herrn abhängig war, blieb der Kaufmann eine Ausnahmeerscheinung, ein haus- und heimatloser Vagabund.

Im 11. Jahrhundert aber trat ein Wandel ein, dem man den Namen »Handelsrevolution« gegeben hat. [. . .] Immer häufiger traten an die Stelle [der fahrenden Händler] Kaufleute, wie wir sie kennen, d.h. deren Waren unterwegs waren, ohne daß sie sie selbst begleiten mußten. In den großen Städten Europas hatten sie Geschäftsführer und Partner. Sie konnten lesen und schreiben, hatten eine kaufmännische Buchführung entwickelt und gegen den Willen der Kirche Schulen gegründet [. . .]. Die sogenannte »Handelsrevolution« war im größten Teil Europas auch mit einem tiefgreifenden gesellschaftlichen Wandel verbunden. Neue Schichten entstanden, andere gingen unter. [. . .] Die Kaufleute, die in der agrarisch-feudalen Welt am untersten Ende der gesellschaftlichen Stufenleiter gestanden hatten, erklommen in einem Siegeszug ohnegleichen deren Spitze. Die neue Organisationsform, die in Italien für diese Form des Handels zu Lande geschaffen worden war, trug den Namen »Compagnia«.“[13]

Neben der Organisationsform der „Compagnia“ wurde zudem ein Finanzierungsmodell eingeführt, die *Commenda*. Sie entspricht unserer heutigen Kommanditgesellschaft. Die Grundidee bestand darin, die Gewinne gemäß der Einlagen und der Arbeitsleistung zu teilen, bei Verlusten jedoch den Kapitalgeber nur mit seinen Einlagen haften zu lassen, nicht aber mit seinem Privatvermögen[14]. „Die Commenda war im 12. Jahrhundert von entscheidender Bedeutung für den wirtschaftlichen Aufschwung der norditalienischen Städte Genua, Florenz, Pisa und Venedig. Die Risikostreuung zwischen den Partnern in Verein mit der Haftungsbeschränkung nach außen ermöglichte die lukrativen, aber gefährlichen Seereisen der italienischen Kaufleute nach Nordafrika und Vorderasien, die Venedig seinerzeit zur reichsten Stadt der Welt machten“[15]. Die Idee der Kapitalbeschaffung mit beschränkter Haftung wurde 1602 in den Niederlanden mit Gründung der ersten Aktiengesellschaft der Welt weiter getrieben, der *Vereenigde Oost-Indische Compagnie* VOC[16]. Durch die VOC stiegen die Niederlande für etwa zwei Jahrhunderte zu einer führenden Handels- und Seemacht auf.

Mit dem massiven Ausbau des Handels und der Entstehung stabiler Handelsstrukturen revolutionierten sich ebenso Kultur und Wissenschaft in Italien und später in ganz Europa durch Einführung wissenschaftlicher Institutionen, wie 1080 die Gründung der ersten europäischen Universität in Bologna[17]. Um 1500 gab es im westlichen Europa 70 Universitäten, und nach der Erfindung des Buchdrucks durch Gutenberg 1455 in Mainz verlagerte sich die Wissensvermittlung vom Mündlichen zum Gedruckten. Dies ermöglichte einerseits überhaupt erst die Entwicklung neuer Ideen und Theorien, aber auch deren Verbreitung in bis dahin ungekannter Geschwindigkeit. So konnten sich in der Renaissance die Ideen des Humanismus (Petrarca um 1350, Erasmus 1511) und der Reformation (Luther 1515) über ganz Europa und den gerade entdeckten Kontinent Amerika ausbreiten. Mit Kopernikus 1543 beginnend und in den folgenden anderthalb Jahrhunderten durch Kepler (1609), Galileo (1623) und Newton (1687) fortgeführt entstand der uns heute geläufige Begriff der Naturwissenschaft. Sie baut Theorien und Model-

[13] Cipolla (1995):S. 9–12.
[14] Sinn (2009):S. 82ff.
[15] Sinn (2009):S. 83.
[16] Sinn (2009):S. 83.
[17] Maddison (2007):S. 69.

le aus Hypothesen auf, die jeweils durch Experimente oder Beobachtungen bewiesen werden müssen oder falsifiziert werden können. Dieser neue Zugang führte zu wissenschaftlichen Erkenntnissen, die in der Folgezeit eine tiefgreifende Technisierung der Wirtschaft ermöglichte.

22.2.3 Die industrielle Revolution der 1770'er Jahre

Innerhalb eines einzigen Jahrzehnts entstand in Großbritannien die Basis für die industrielle Revolution, die die auf Feudalismus und Zunftstruktur fußende Wirtschaftsordnung Europas und Nordamerikas in den nachfolgenden 150 Jahren hinwegfegen sollte:

1771 Die erste Fabrik, in der die zentralen Arbeitsschritte nicht von Menschen sondern von Maschinen ausgeführt werden, ist die von dem englischen Perückenmacher Richard Arkwright (1732–1792) in Cromford, Derbyshire, errichtete und von Wasserkraft angetriebene maschinelle Baumwollspinnerei[18].

1776 Der schottische Philosoph Adam Smith (1723–1790) veröffentlicht sein Buch *„An Inquiry into the Nature and Causes of the Wealth of the Nations"*. Damit legt er die theoretischen Grundlagen der Marktwirtschaft und spricht sich insbesondere für freien Handel ohne Schutzzölle und Arbeitsteilung aus[19]. Er erkennt als zentrale Produktionsfaktoren Kapital, Arbeit und Boden; die Begriffe Energie und Information waren zur damaligen Zeit noch gar nicht bekannt[20].

1776 Der schottische Instrumentenmacher James Watt (1736–1819) installiert die Dampfmaschine in wirtschaftlichem Betrieb[21]. 1782 wird ihm das Patent für seinen „Universalmotor" erteilt, eine Dampfmaschine, die mit einem bis dahin unerreichten Wirkungsgrad eine Hubbewegung in eine Drehbewegung umwandelt[22].

Der erste Wirtschaftszweig, der von der Industrialisierung erfasst wird, ist die englische Textilbranche, später folgt die Stahlbranche aufgund effizienterer Herstellungsverfahren, noch später entsteht mit dem Eisenbahnwesen eine völlig neue Branche. Großbritannien wird fast ein Jahrhundert lang die wirtschaftliche Vormachtstellung weltweit behalten, bevor es vom Deutschen Reich und vor allem von den USA als stärkste Wirtschaftsmacht abgelöst wird, insbesondere durch die Entstehung der Automobil- und Chemieindustrie.

22.3 Ökonomische Mechanik: Das Coase'sche Gesetz

Die Gesamtkosten zur Erstellung eines Produkts, sei es eine Ware oder eine Dienstleistung, ergeben sich aus der Summe der Transaktionskosten und der im Unternehmen anfallenden Produktionskosten. Die *Transaktionskosten* umfassen hierbei Informationskosten für die Suche nach geeigneten Produzenten oder Zulieferern, Anbahnungs- und Vereinbarungskosten, sowie Abwicklungskosten, beispielsweise Maklercourtage oder Transportkosten.

Satz 22.1 (Coase'sches Gesetz 1937). *[23] Ein Unternehmen lässt eine Ware oder Dienstleistung genau dann horizontal, also auf dem freien Markt, produzieren, wenn die Transaktionskosten geringer sind als die internen Produktionskosten. Andernfalls produziert es vertikal, also unternehmensintern.*

[18] Gaede (2008):S.24–37.
[19] Gaede (2008):S.38–39.
[20] Kümmel (2011):S. 227.
[21] Kümmel (2011):S. 17, 49f, 227.
[22] Gaede (2008):S.40–52.
[23] Tapscott und A. D. Williams (2007):S. 55ff.

Der englische Wirtschaftswissenschaftler Ronald H. Coase fragte sich nach einem Besuch der amerikanischen Automobilfirmen Ford und General Motors Mitte der 1930er Jahre, was ein Großunternehmen von einem kommunistischen Staatssystem wie der Sowjetunion unterscheidet. Und grundsätzlicher: Warum handeln die Menschen nicht als individuelle Käufer und Verkäufer, sondern arbeiten in Unternehmen mit Tausenden anderen zusammen?[24] Coase's Antwort in seinem 1937 veröffentlichten Artikel war die Einführung und Berücksichtigung der Transaktionskosten, die in einer Marktwirtschaft als ökonomisches Regulativ für Ein- und Auslagerungen von Produktion wirkt.

Das Coase'sche Gesetz vermochte somit zu erklären, wieso sich aus der ursprünglich merkantilen, vorwiegend aus Kleinstunternehmern wie Handwerkern, Einzelhändlern und Bauern bestehenden Wirtschaft des Späten Mittelalters und der Renaissance die Industriegesellschaft des 19. und beginnenden 20. Jahrhunderts entwickeln konnte, die durch monolithische Großunternehmen der Metall-, Rohstoff-, Chemie- oder Energieindustrie geprägt war[25]. Zwar spielten für diese Entwicklung auch Skaleneffekte *(economies of scale)* eine wichtige Rolle, also am Ende die höhere Effizienz von Massenproduktion. Aber erst durch die Steigerung der Transaktionskosten, vor allem aufgrund Zeit und Ressourcen benötigender Suche nach geeigneten Zulieferern für die technisch immer anspruchsvolleren Produkte, wurde es ökonomisch rational, die Produktion ins Unternehmen zu verlagern. Einer der Vorreiter dieser Entwicklung war Henry Ford mit seiner berühmten Fabrik in River Rouge, die auf der einen Seite Rohgummi und Stahl verschluckte und fertige Automobile auf der anderen Seite ausspie.

Nach dem Zusammenbruch der Planwirtschaften des Warschauer Pakts Anfang der 1990er Jahre setzte jedoch eine umgekehrte Entwicklung ein, die *Globalisierung*. Ein großer Teil der Produktion wurde von den westeuropäischen und nordamerikanischen Wirtschaftsmächten in die Länder Osteuropas und Südostasiens ausgelagert. Auch diese Entwicklung zu globalisierten Lieferketten und Ausgründungen *(Outsourcing)* lässt sich mit dem Coase'schen Gesetz erklären. Denn in Osteuropa und Südostasien waren die Lohnkosten so niedrig, dass insgesamt die Transaktionskosten für die horizontale Produktion geringer waren als die Produktion im eigenen Unternehmen oder selbst im eigenen Land.

Doch die Lohnkosten allein können die Dynamik der Ausgründungen und der Globalisierung nicht erklären. Im letzten Jahrzehnt sind durch Einsatz von IT und Internet, also durch die Digitalisierung von Information und die zunehmende Vernetzung, vor allem die Informationskosten pro Transaktion gesunken[26], so durch die Vermittlung von Dienstleistungen (z.B. Elance.com) und nicht zuletzt durch Suchdienste wie Google. In letzter Zeit konnte so eine ganze Industrie von Kleinstunternehmen und Individuen für die Programmierung von Apps für mobile Endgeräte wie Smartphones entstehen.

22.4 Web 2.0, vernetzte Mobilität und Big Data

Web 2.0. Der Begriff Web 2.0 wurde 2004 bei der Planung einer Konferenz über den sich vollziehenden Wandel des Internets von Dale Dougherty und Craig Cline bekannt gemacht.[27] Eine einheitliche und allgemeingültige Definition des Begriffs gibt es nicht. Oftmals wird darunter auf technischer Ebene im Kern die Verwendung von XML (z.B. bei RSS) und clientseitigen Techni-

[24] Tapscott und A. D. Williams (2007):S. 55f.

[25] Gaede (2008).

[26] Kollmann (2009):S. 5.

[27] T. O'Reilly: "What is Web 2.0?", `http://wwww.oreilly.de/artikel/web20.html` [2010-07-05]. Nach der englischsprachigen Wikipedia-Seite scheint Darcy DiNucci 1999 als Erste den Begriff Web 2.0 erwähnt zu haben, spätestens 2003 taucht er in Blogs über Web Services auf, `http://en.wikipedia.org/wiki/Web_2.0#History` [2010-07-10].

ken wie Javascript (AJAX) und Flash Video verstanden. Deren Einsatz hat im Kern die logische Aufhebung der durch das HTTP-Protokoll bedingten hierarchischen Client-Server-Architektur des Internets. Beispiele für solche Software sind Google Docs, YouTube oder Flickr.

Verbreiteter ist die Definition aus *soziologischer* Sicht, gemäß der das Web 2.0 über neuartige Internetanwendungen die Interaktion und Zusammenarbeit von Menschen fördert, indem auf Webseiten Inhalte der Besucher veröffentlicht werden. Beispiele für solche Anwendungen und Plattformen sind Blogs, Wikis und soziale Netzwerke.

Anders, als der Name Web 2.0 suggeriert, handelt es sich dabei allerdings nicht um einen schlagartig durchgeführten Releasewechsel, sondern eher um eine mehrere Jahre andauernde Entwicklung. Unter den ersten kommerziellen Anbietern, die den Nutzern eine aktivere Rolle gewährten, waren Amazon, das schon früh die Möglichkeit zu Rezensionen über die dort angebotenen Bücher gab, und die 1995 gestartete Auktionsplattform eBay, bei der der Nutzer nicht nur Teile der Inhaltsgenerierung, sondern die gesamte Erstellung der Web-Inhalte wie Produktbeschreibungen oder Verkäuferbewertungen übernimmt.

Vernetzte Mobilität. Spätestens mit der Einführung des iPhones durch Apple im Jahre 2007[28] und etwas später dem ersten Android-Smartphone durch HTC und T-Mobile in 2008[29] ist vernetzte Mobilität zu einem Massenphänomen geworden. Ein *Smartphone* ist ein Mobiltelefon mit einem Betriebssystem mit offengelegter API, die die Installation von weiteren Programmen, (*Apps*) zulässt.

Technisch möglich wurde die vernetzte Mobilität durch die weitgehende Miniaturisierung der Computer, so dass ein Rechner in nahezu jeder Situation bedient werden kann, und durch höhere Übertragungsraten der Mobiltelefonnetze.

Big Data. In den 2000er Jahren explodierte die Speicherkapazität an Information[30]. Seit Mitte der 2000er Jahre wird weltweit mehr Information digital gespeichert als analog. Den größten

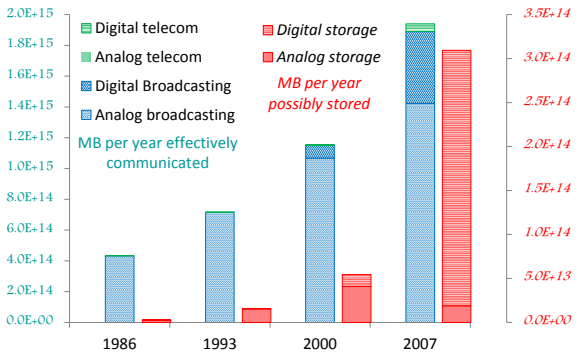


Abbildung 22.1: Gesamtspeicherkapazität und effektive Sendekapazität an Information, in optimal komprimierten MB pro Jahr für 1986, 1993, 2000 und 2007, logarithmisch skaliert. Das Wachstum ist überexponentiell. (Quelle: Hilbert und López (2012:Fig. 1))

Anteil an analoger Speicherkapazität hatten bis in die 2000er Jahre Videobänder, den größten Anteil an digitaler Speicherkapazität hatten bisher stets PC Festplatten[31].

Durch die Vernetzung dieser riesigen Datenmengen entsteht das Phänomen der „Big Data“. *Big Data* bezeichnet Datenmengen, die zu groß oder zu komplex sind, um sie mit klassischen Methoden der Datenverarbeitung auszuwerten.

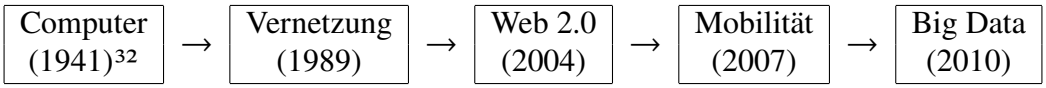
[28] `http://de.wikipedia.org/wiki/Iphone#Verkaufsstart` [2011-11-18]

[29] `http://en.wikipedia.org/wiki/List_of_Android_devices#HTC` [2011-11-18]

[30] Hilbert und López (2011).

[31] Hilbert und López (2011).

Insgesamt lässt sich die technische Entwicklung des letzten halben Jahrhunderts grob durch die folgenden Meilensteine illustrieren, an deren Ende die Mobiltelefonie mit den Internettechniken verschmolzen ist:



Durch die automasierte Sammlung der durch die mobilen Endgeräte verfügbaren Daten (Ort, Zeit, Tätigkeit, Kontakte, Interesse, Verhalten, . . .) können individualisierte (nicht notwendig *personalisierte*) Profile erstellt werden. Damit wiederum können Muster berechnet werden, welche Datenkombinationen welches weiteres Verhalten mit welcher Wahrscheinlichkeit zur Folge haben. Mit anderen Worten, das Verhalten eines Netzteilnehmers kann stochastisch anhand seiner Vergangenheitsdaten prognostiert werden. Die Algorithmen berechnen Interessen, Neigungen und Vorlieben, und können so individuelle und hilfreiche Hinweise geben, sei es das in einer fremden Stadt nächstgelegene passende Restaurant oder eine interessierende Neuerscheinung. Auf diese Weise werden die mobilen Endgeräte zunehmend zu „persönlichen intelligenten Lebensassistenten“[33], durch deren häufige Nutzung wiederum die auswertbare Datenbasis wächst. Eine aus des Netzteilnehmers nutzbringende Datenübermittlung führt so zu einem sich stetig verstärkenden Datenbestand, der den Nutzen vordergründig erhöht.

22.5 Die Generation Y

„Generation Y“, oft auch *Netzgeneration* genannt[34], ist die Bezeichnung für die Geburtenjahrgänge Mitte der 1970’er bis in die 2000’er Jahre. Sie sind damit die Nachfolgegeneration der Baby-Boomer nach dem Zweiten Weltkrieg (die Jahrgänge 1955–1965 in Deutschland, in den USA ein Jahrzehnt früher) und der in ersten ökonomischen Krisen- und Stagnationszeiten aufgewachsenen Generation X (die Jahrgänge 1965–1980). Die Generation Y ist die erste

Software-Pionier	**Geburtsjahr**	**Unternehmen, Gründungsjahr**
Baby-Boomer		
Bill Gates	1955	Microsoft, 1975
Steve Jobs	1955	Apple, 1976
Generation X		
Jeff Bezos	1964	Amazon.com, 1994
Pierre Omidyar	1967	eBay, 1995
Jimmy Wales und Larry Sanger	1966 & 1968	Wikipedia, 2001
Philip Rosedale	1968	Second Life 2003
Linus Torvalds	1969	Linux, 1991
Julian Assange	1971	WikiLeaks, 2006
Larry Page und Sergei Brin	1973	Google, 1998
Generation Y		
Mark Zuckerberg	1984	Facebook, 2004

Tabelle 22.1: IT-Pioniere und ihre Generation. (Quelle: Wikipedia `http://en.wikipedia.org`)

Jahrgangskohorte, deren Mitglieder zumindest seit ihrer Pubertät mit dem Internet und mobilen Technologien vertraut sind („Medialisierung der Jugend“). Ist für die älteren Generationen

[32] Der erste elektronische Digitalrechner war die Z3 von Konrad Zuse und Helmut Schreyer im Jahre 1941. Sie wurde am 21.12.1943 bei einem Bombenangriff auf Berlin zerstört, ein Nachbau existiert im Deutschen Museum in München; `http://de.wikipedia.org/wiki/Zuse_Z3` [2011-11-18]

[33] Kurz und Rieger (2013):S. 256f.

[34] Tapscott und A. D. Williams (2007):S. 37ff.

das Web eher ein Ort zur effizienten Informationsbeschaffung und Recherche, so bedeutet es für die Generation Y eine natürliche Lebensform, ein Medium auch für soziale Kontakte und Kommunikation. Das Internet ist für sie nicht allein die Vernetzung von Technik, sondern die Vernetzung von Menschen durch Technik[35].

Aufschlussreich ist Tabelle 22.1 der IT-Pioniere der verschiedenen Generationen und ihrer wesentlichen Firmen- oder Projektgründungen. Waren die Innovationen der Baby-Boomer Bill Gates und Steve Jobs einzelplatzbasierte Betriebssysteme, so entstehen durch die Generation X mit Amazon, eBay und Google erste webbasierte Geschäftsmodelle, und mit der Generation Y entwickeln sich soziale Netzwerke.

Sowohl als Konsumenten als auch als Produzenten wird die Generation Y die wirtschaftliche Entwicklung der nächsten Jahrzehnte bestimmen. Beispiele dafür sind die in den letzten Jahren expandierende Wirtschaftszweige der Sozialen Netzwerke und der Online-Spiele.

22.6 Begriffsdefinition *Digitale Ökonomie*

Was nun ist eigentlich „Digitale Ökonomie“? Die Definition dieses Begriffs ist nicht einheitlich und wird synonym oder abgewandelt auch unter anderen Bezeichnungen verwendet. In Anlehnung an Kollman[36] und die Seouler Erklärung der OECD[37] definieren wir:

Definition 22.2. Ein *Geschäftsprozess* beschreibt eine Folge von Einzeltätigkeiten („Aktivitäten“), die schrittweise sequenziell oder parallel ausgeführt werden, um ein geschäftliches Ziel zu erreichen. Der wirtschaftliche Bereich derjenigen Geschäftsprozesse, die wesentlich auf digitaler Information basieren, heißt *digitale Ökonomie* oder *Informationsökonomie*. Speziell die Geschäftsprozesse, die auf der Informationsübermittlung über das Internet basieren, also die elektronischen Geschäftsprozesse, bilden die *Netzökonomie (Net Economy)* oder *Internetökonomie*. □

Mit dieser Definition gilt also strenggenommen die Inklusionskette

$$\boxed{\text{Netzökonomie} \subset \text{digitale Ökonomie} = \text{Informationsökonomie}}$$

Beispielsweise gehörten die in den 2000’er Jahren auf Mobiltelefonen populären Handyspiele *(mobile games)* oder der Verkauf von Klingeltönen *(ring tones)* zur digitalen Ökonomie, ebenso wie Computer- und Videospiele. Da die Geschäftsaktivitäten dieser Bereiche aber nicht (nur) über das Internet erfolgten, sind sie nicht Teil der Internetökonomie. Das Internet erlangte jedoch zunehmende Bedeutung für Wirtschaft und Gesellschaft, insbesondere für die Mobiltelefonie und die Computerspiele, so dass man die Näherungsgleichung setzen kann:

$$\boxed{\text{Netzökonomie} \approx \text{digitale Ökonomie} = \text{Informationsökonomie}}$$

Ein wichtiger Bereich der digitalen Ökonomie ist das *E-Business*, die integrierte Ausführung der digitalisierbaren Geschäftsprozesse eines Unternehmens durch die elektronischen Informations- und Kommunikationstechnologie (IKT). Entsprechend den klassischen Geschäftsfeldern Verkauf, Einkauf und Handel umfasst das E-Business insbesondere die folgenden Geschäftsfelder[38]:

- Das *E-Procurement* ermöglicht den elektronischen Einkauf von Produkten oder Dienstleistungen über digitale Netzwerke.

[35] Tapscott und A. D. Williams (2007):S. 302.
[36] Kollmann (2009):§1.5.
[37] OECD (2008):S. 4.
[38] Kollmann (2009):§1.5.1.

- Der *E-Commerce* ermöglicht den elektronischen Verkauf von Produkten oder Dienstleistungen über digitale Netzwerke, typischerweise durch einen *E-Shop*.

- Eine *(elektronische) Handelsplattform*, oder ein *E-Marketplace*, ermöglicht den elektronischen Handel Produkten oder Dienstleistungen über digitale Netzwerke.

22.7 Geschäftsmodelle

Ein *Geschäftsmodell (business model)* ist eine modellhafte Beschreibung eines betriebswirtschaftlichen Geschäftes zur Erklärung oder Identifizierung der Schlüsselfaktoren des Unternehmenserfolges. Ein Geschäftsmodell besteht aus drei Hauptkomponenten:[39]

- Das *Nutzenversprechen* beschreibt den Nutzen, den Kunden und Partner des Unternehmens ziehen.

- Die *Wertschöpfungsarchitektur* beschreibt, wie der Nutzen für die Kunden und Partner generiert wird und welche Leistungen auf welchen Märkten angeboten werden.

- Das *Ertragsmodell* oder *Erlösmodell* beschreibt, welche Einnahmen das Unternehmen aus welchen Quellen erzeugt.

22.7.1 Ertragsmodelle

Die folgenden typischen Ertragsmodelle existieren in der digitalen Ökonomie[40]:

- *Margenmodell*: Bei diesem Ertragsmodell wird die Leistung des Unternehmens direkt an den Kunden verkauft. Der zu zahlende Preis für das Produkt ergibt sich aus der Summe der variablen und der fixen Kosten und einer Gewinnmarge. Ein typisches Beispiel für das Margenmodell ist ein E-Shop wie amazon.com.

- *Provisionsmodell*: Für die Vermittlung von Fremdleistungen erfolgt eine erfolgsabhängige Provisionszahlung. Typisches Beispiel für dieses Ertragsmodell sind Handelsplattformen wie e-bay oder Adwords von Google.

- *Grundgebührmodell*: Bei dieser Erlösform wird beim Angebot der eigenen Leistungen eine Gebühr erhoben, beispielsweise eine Zugangs-, Bereitstellungs- oder Aufnahmegebühr. Typisches Beispiel für dieses Ertragsmodell sind Handelsplattformen.

22.7.2 Geschäftsmodelle der digitalen Ökonomie

Im Wesentlichen haben sich in den letzten beiden Jahrzehnten zwei erfolgreiche Geschäftsmodelle der digitalen Ökonomie herausgebildet, einerseits der geschlossene Konzernkosmos mit der Option auf Auswertung von Big Data und andererseits Moral-Hazard-Systeme, die Risiken auf die Nutzer übertragen und so klassische Berufsfelder verdrängen.

[39] P. Stähler: `http://www.business-model-innovation.com/definitionen/geschaeftsmodell.htm`
[40] Kollmann (2009):§1.5.1.

Geschlossener Konzernkosmos

Charakteristisch für Geschäftsmodelle der digitalen Ökonomie ist, dass nicht nur eine einzelne Leistung angeboten wird, sondern neben einer Kernleistung eine oder mehrere Nebenleistungen. Ein Geschäftsmodell mit einer Kombination von Kern- und Nebenleistungen in der Form, dass die Nebenleistungen ohne die Kernleistung nicht möglich wären, funktioniert nach dem *Symbiose-Prinzip*[41]. Typische Beispiele dafür sind die Geschäftsmodelle der Sozialen Netze oder der Suchmaschinen. Im Exremfall wird die Kernleistung kostenlos angeboten und der Ertrag ausschließlich über die Nebenleistungen erzielt.

In der Regel wird dabei offensiv die Kundenbindung durch das Geschäftsmodell eines *geschlossenen Konzernkosmos* erhöht, also durch eine logische und informationstechnische Infrastruktur, die den Kunden zwingt, im System zu bleiben[42]. Beispiele dafür sind Apples iTunes, Googles Play oder Amazons Kindle, die den Zugriff auf die erworbenen Produkte vornehmlich über einen Login erlaubt. Speichert man die erworbenen Daten nicht lokal auf seinem Rechner, so führt eine Kündigung des Logins zum Verlust der gesamten Bibliothek oder der kompletten Musiksammlung.

Eine Folge eines geschlossenen Konzernkosmos ist ein Phänomen, das der Informatiker und Publizist Jaron Lanier eine *kommerzielle Asymmetrie* nennt[43]. Er verdeutlicht dies am Beispiel von E-Books. Der Käufer eines gedruckten Buches besitzt einen materiellen Wert, über den er nach Belieben verfügen kann. Er kann das Buch verleihen oder weiter verkaufen, er kann es signieren lassen und als Wertanlage betrachten. Bei einem E-Book ist man dagegen nicht mehr Käufer erster Klasse, da man bei einem Unternehmen über einen Vertrag nur eingeschränkte Rechte an einem Buch gekauft hat. Will man ein anderes Lesegerät verwenden oder in eine andere Cloud wechseln, verliert man je nach Grad der Geschlossenheit des Konzernkosmos vielleicht sogar den Zugang zu seinem Buch, obwohl man es doch „gekauft“ hat.

Ein weiterer wichtiger Bestandteil des geschlossenen Konzernkosmos ist die *Datenasymmetrie* zwischen Kunde und Unternehmen. In den Nutzungsvereinbarungen aller elektronischen sozialen Netzwerke gibt es keine Angaben über Art und Verwendung der Daten des Nutzers innerhalb des Unternehmens (es existieren in der Regel nur Zusicherungen zum Datenschutz bei der Datenweitergabe an Dritte), d.h. der Nutzer hat keinen Einfluss auf alle Informationen, die er dem Unternehmen durch seine Einstellungsangaben und sein Verhalten im Netzwerk gibt.[44] In einigen Netzwerken wird sogar offensiv versucht, die nationale Gerichtsbarkeit durch individuelle Schiedsverfahren außer Kraft zu setzen, so zum Beispiel in den Nutzungsbedingungen von Instagram.[45] Diese Datenasymmetrie ermöglicht es dem Netzwerkunternehmen, mit statistischen Auswertungsalgorithmen die riesigen individuellen und aggregierten Datenmengen (Big Data) zu nutzen. Beispielsweise registriert Amazon die Markierungen, die die Nutzer in einem E-Book vornehmen, und zeigt sie bei einer signifikanten Anzahl allen Nutzern als Zusatzinformation an. Wie Amazon diese Informationen weiter nutzt, ist allerdings nicht öffentlich bekannt.

---

[41] Kollmann (2009):§1.5.2.

[42] Bauer et al. (2015):S. 81.

[43] Lanier (2014):S. 318ff.

[44] `https://de-de.facebook.com/legal/terms`, `http://www.google.com/intl/de/policies/privacy/`, `https://twitter.com/privacy?lang=de`, `https://www.whatsapp.com/legal/`

[45] `https://help.instagram.com/478745558852511`

Moral Hazard

Unter *moralischem Risiko* oder *Moral Hazard* versteht man leichtfertiges Verhalten in dem Bewusstsein, dass die Allgemeinheit oder jemand anderes die Kosten im Schadensfall trägt[46],[47], siehe auch Seite 128. Moral Hazard bewirkt zwar einerseits eine Risikoreduzierung für den leichtfertig Handelnden, führt aber am Ende stets zu einer Erhöhung der Kosten des Gesamtsystems[48], wie beispielsweise bei der Finanzkrise 2008, siehe Seite 183. Viele Unternehmen der digitalen Ökonomie nutzen diesen Effekt und wälzen etwaige Risiken auf die Nutzer ab, indem sie sich auf eine reine Vermittlungsfunktion zurückziehen und sich damit jeglicher rechtlicher Verantwortung entledigen. Ein Beispiel ist YouTube, wo man zwar kostenlos Filme einstellen kann, die Nutzer allerdings alle Kosten zur Erstellung eines Films selbst tragen müssen[49].

Nun nutzt YouTube das Moral Hazard zu einem recht geringen Grad, da andererseits weder für die Veröffentlichung der Inhalte noch für deren Betrachten Gebühren anfallen und daher der Erlös für YouTube verschwindend ist. Ganz offensiv wird Moral Hazard jedoch für die Dienste der *Shared Economy* ausgenutzt, die wie `airbnb.com` Zimmer oder wie `uber.com` Taxifahrten über ein Netzwerk vermitteln und damit Sozialabgaben und Versicherungskosten einsparen, die Risiken übernehmen die Nutzer[50][51].

[46] Bofinger (2007):S. 256.
[47] Lanier (2014):S. 88, 353f.
[48] Lanier (2014):S. 88ff.
[49] Lanier (2014):S. 355.
[50] `https://www.uber.com/legal/deu/terms`, `https://www.airbnb.de/terms`
[51] Lanier (2014):S. 355.

Auf die IBM-Ära mit den Großrechnern folgte die Microsoft-Ära mit den Personalcomputern, und nun sind wir in der Internet-Ära angekommen, die man besser Google-Ära nennen sollte.

Tim O'Reilly in der FAZ, 20.11.2006

23 Google und seine Kerntechnologien

Kapitelübersicht

Als Larry Page und Sergey Brin Ende der 1990er Jahre die Firma Google gründeten, boten sie als zentrale Dienstleistung die Suche nach Begriffen an. Diese Dienstleistung würde ohne das Internet gar nicht existieren, d.h. Google stellte die zu dieser Zeit modernste Art des Wirtschaftens dar. Dabei produzierte Google zunächst nichts wirklich, es verarbeitete nur Inhalte anderer. Die eine geniale Idee der Firmengründer Page und Brin war es, die Suche nach Inhalten besser durchzuführen, als es vorher möglich war. Ferner war die Erkenntnis entscheidend, dass die Suche nach Information ein *zentraler Bedarf* für die Nutzer des Internets war und bleiben würde. Eine Einschätzung, die Ende der 1990er Jahre IT-Firmen wie Yahoo! oder Microsoft übrigens nicht teilten, die Internetsuche erschien ihnen eher als ein Zusatzangebot, nie als zentrale Dienstleistung[1].

Allerdings ließ sich Google diese Dienstleistung *nicht* bezahlen. Was auf den ersten Blick keine geniale Idee zu sein schien, war jedoch der Verbreitung des Dienstes sehr zuträglich. Erfolgreich war zudem die auf die Firma GoTo.com zurückgehende Idee, genau diejenige Information, die zur Verbesserung der Suche benötigt wurde, und die Popularität des Dienstes zur kontextsensitiven Werbung einzusetzen und *damit* Geld zu verdienen. Damit ähnelt Googles Geschäftsmodell demjenigen des Privatfernsehens, das seine eigentliche Dienstleistung, nämlich die Übertragung von Unterhaltungs- und Nachrichtensendungen, kostenlos anbietet, und durch separate Werbesendungen verdient.

23.1 Wirtschaftliche Kennzahlen

Google Inc. ist ein Unternehmen mit Sitz in Mountain View, Kalifornien, das am 7.9.1998 von den beiden Informatikern Larry Page und Sergey Brin mit einem Startkapital von 1,1 Mio US$

[1] Vise und Malseed (2006):S. 91.

in einer Garage in Menlo Park gegründet und am 15.9.1998 mit einer Suchmaschine unter der Domain google.com ans Netz ging.[2] Nach einer Umstrukturierung des Konzerns gehört Google Inc. seit dem 2. Oktober 2015 zur Alphabet Inc. (`http://abc.xyz`).[3]

Google wird seit dem 19. August 2004 an der Börse gehandelt, damals mit einem Preis von $ 85 je Aktie und mit 19 605 052 Aktien, was einen Börsenwert von $ 1,67 Milliarden ergab. Am 11. September 2009 lag der Kurs für eine Google-Aktie an der Nasdaq bei $ 472,14, was mit damals 316,57 Mio Aktien einem Börsenwert von $ 149,47 Milliarden entsprach.[4] Damit hatte Google am 11. September 2009 an der Nasdaq einen höheren Börsenwert als die Deutsche Telekom ($ 60 Milliarden) und die Daimler AG ($ 51 Milliarden) zusammen.

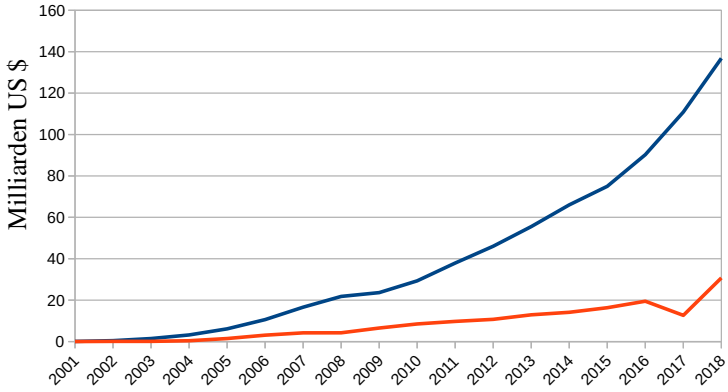


Abbildung 23.1: Umsatz und Gewinn von Google. Quelle: `https://abc.xyz/investor/`.

Nach dem Geschäftsbericht vom 1. Februar 2019[5] erwirtschafteten Ende 2018 weltweit 98 771 Beschäftigte *(employees)* bei Alphabet 2018 einen Umsatz *(revenues)* von $ 136,819 Milliarden und einen Gewinn *(net income)* von $ 30,7 Milliarden.

Betrachtet man die Marktanteile für Googles zentrale Dienstleistung, die Websuche, so erkennt man, dass Google den Markt der Suchmaschinen weltweit stark dominiert.

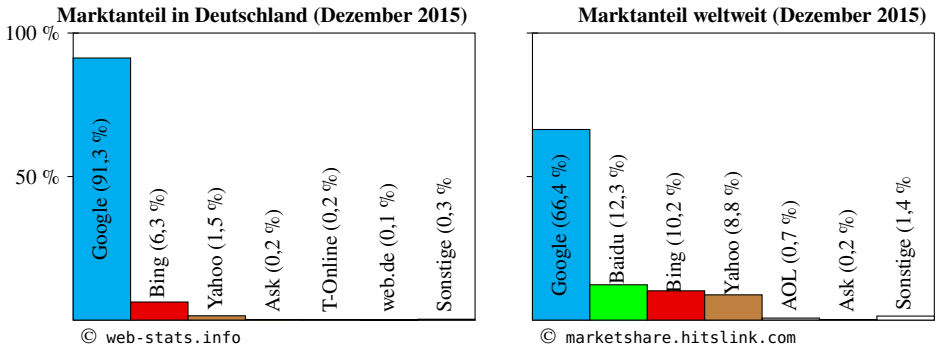


Abbildung 23.2: Marktanteile der Suchmaschinen.

[2] `http://en.wikipedia.org/wiki/Google`

[3] `http://www.heise.de/newsticker/meldung/Revolution-bei-Google-Abspaltungen-und-eine-Holding-2775977.html`

[4] `http://finance.yahoo.com/q/ks?s=Goog` [2009-09-12]

[5] `https://abc.xyz/investor/`, Reiter *Income Statement*

23.2 Suchmaschinen

Eine *Suchmaschine (search engine)* ist ein Programm zur Recherche elektronischer Dokumente, die in einem Computer oder einem Computernetzwerk gespeichert sind. Suchmaschinen im Internet wurden seit Mitte der 1990er Jahre entwickelt und eingesetzt. Sie erstellen einen Index von Schlüsselworten für die Dokumentbasis, um Suchanfragen über Schlüsselwörter mit einer nach Relevanz geordneten Trefferliste zu beantworten. Nach Eingabe eines Suchbegriffs liefert eine Suchmaschine auf diese Weise eine Liste von Verweisen auf möglicherweise relevante Dokumente, meistens dargestellt mit Titel und einem kurzen Auszug des jeweiligen Dokuments.

Die Suche in elektronischen Dokumentenetzwerken kann nach unterschiedlichen Methoden durchgeführt werden und gründet sich auf der Theorie des *Information Retrievals*, einem Fachgebiet der Informatik, das sich mit der Suche nach komplexen Inhalten beschäftigt, vor allem in Texten, aber auch anderen Medien wie Bildern, Tondateien, Videos oder Datenbanken.

Eine Websuchmaschine sucht nach Dokumenten und Informationen speziell im WWW und stellt die Ergebnisse auf einer Suchergebnisseite *SERP (search engine result page)* dar. Die wesentlichen Prozesse einer Websuchmaschine sind:

- *Web crawling*, also das Auffinden der zu durchsuchenden URLs und deren Inhalte und das Abspeichern der Daten in einer Datenbank.
- *Indexing*, also die Erstellung eines Indexes, also einer Datenbank mit Informationen und Metadaten über Dokumente.
- Verarbeiten von Suchanfragen und Aufbereitung der Ergebnisse in einer SERP.

Siehe dazu auch Abbildung 23.3. Ein *Crawler*, oft auch *Spider* genannt, ist hierbei ein Programm,

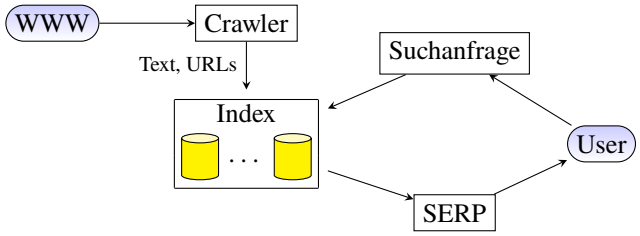


Abbildung 23.3: Prozesse einer Websuchmaschine.

das von einer gegebenen Menge initialer Webseiten startet und allen weiteren Hyperlinks auf andere Seiten folgt. Auf diese Weise werden alle Webseiten gefunden, die mit den Initialseiten direkt oder indirekt verlinkt sind, allerdings bleiben nicht mit ihnen verlinkte Webseiten verborgen.

23.3 Der PageRank-Algorithmus

Wie ordnet Google Webseiten zu eingegebenen Suchbegriffen? Die Kernidee ist der *PageRank-Algorithmus* der Firmengründer Page und Brin (initiiert von Page, daher auch „PageRank“). Er wurde 1998 veröffentlicht[6]. Im Wesentlichen bewertet er alle Knoten (Webseiten) eines Graphen (Internet) durch die gewichtete Anzahl aller gerichteten Kanten (Links) auf die Knoten, wobei das Gewicht einer Kante umso höher ist, je höher die Bewertung des Knotens ist, von dem sie kommt[7]. Mathematisch ist dieses rekursive Problem für N Knoten formulierbar als eine lineare

[6] Brin und Page (1998); Page et al. (1999).
[7] Easley und Kleinberg (2010):§14.3.

Matrizengleichung für die Unbekannte $\vec{x}$, einen N-Vektor, dessen i-ter Eintrag der gesuchte PageRank x_i des Knotens i ist[8]:

$$\vec{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}, \tag{23.1}$$

Je höher der PageRank x_i für Knoten i, desto höher ist seine Bewertung. Die zu lösende Gleichung hat die Form $\vec{x} = dA\vec{x} + \vec{b}$, vgl.[9], oder umgestellt

$$(I - dA)\,\vec{x} = \vec{b}, \tag{23.2}$$

wobei $0 < d < 1$ ein Dämpfungsfaktor ist (üblicherweise empirisch auf $d = 0{,}85$ gesetzt), I die $(N \times N)$-Einheitsmatrix, $\vec{b}$ der N-dimensionale Vektor mit den gleichen Einträgen $\frac{1-d}{N}$ und A die „modifizierte Adjazenzmatrix" des Graphen, also

$$I = \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix}, \qquad \vec{b} = \frac{1}{N}\begin{pmatrix} 1-d \\ \vdots \\ 1-d \end{pmatrix}, \qquad A = \begin{pmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{pmatrix}, \tag{23.3}$$

mit

$$a_{ij} = \begin{cases} 1/k_j^{\text{out}}, & \text{wenn Knoten } j \text{ nach Knoten } i \text{ verweist,} \\ 0 & \text{sonst.} \end{cases} \tag{23.4}$$

Hier ist k_j^{out} der *Außengrad* von Knoten j, also die Anzahl aller Kanten, die von Knoten j wegweisen. In der Spalte j der Matrix A stehen also nur Nullen oder der gleiche Wert $1/k_j^{\text{out}}$.

Generell ist nach Konstruktion eine Spaltensumme der Matrix A entweder 0 oder 1. Verweist ein Knoten auf keinen anderen Knoten, so heißt er *Sackgasse (dead end)*. Ist Knoten j keine Sackgasse (d.h. $k_j^{\text{out}} > 0$), so ergibt seine Spaltensumme in der Matrix A genau 1,

$$\sum_{i=1}^{N} a_{ij} = 1 \qquad \text{für jedes feste } j \text{ mit } k_j^{\text{out}} > 0. \tag{23.5}$$

Gibt es in einem Netzwerk also keine Sackgassen (d.h. $k_j^{\text{out}} > 0$ für alle Knoten j), so ist A eine „stochastische Matrix"[10]. Ferner kann man zeigen, dass in einem solchen Netzwerk die Summe aller PageRanks genau 1 ergibt (z.B. das fünfte Netz in Tab. 23.1).

Die Mathematik von PageRank ist gut erklärt in[11] oder (wenn auch mit einem mit dem Faktor N multiplizierten PageRank $\vec{x} \mapsto N\vec{x}$) auf der Webseite

`http://www.suchmaschinen-doktor.de/algorithmen/pagerank.html` [2019-05-02]

Für einige „Mini-Webs" sind die Page-Ranks in Tabelle 23.1 exemplarisch berechnet. Natürlich ist eine Lösung über inverse Matrizen für das Internet mit mehreren Milliarden Webseiten ($N \approx 4 \cdot 10^{10}$, siehe [12]) nicht mehr praktikabel, selbst nicht mit den riesigen Rechner-Clustern von Google. Allerdings benötigt Google auch gar nicht die algebraisch exakte Lösung, sondern es genügt eine Näherung. Ein einfaches numerisches Näherungsverfahren geht auf Gauß zurück, der es aber nie veröffentlicht hat, und welches später der Mathematiker und Astronom Ludwig

[8] Easley und Kleinberg (2010):§14.6.
[9] Newman (2010):(7.16).
[10] Zeidler (1996):§6.4.2.
[11] Brandes und Dorfmüller (2008).
[12] `http://www.worldwidewebsize.com/` [2016-02-01]

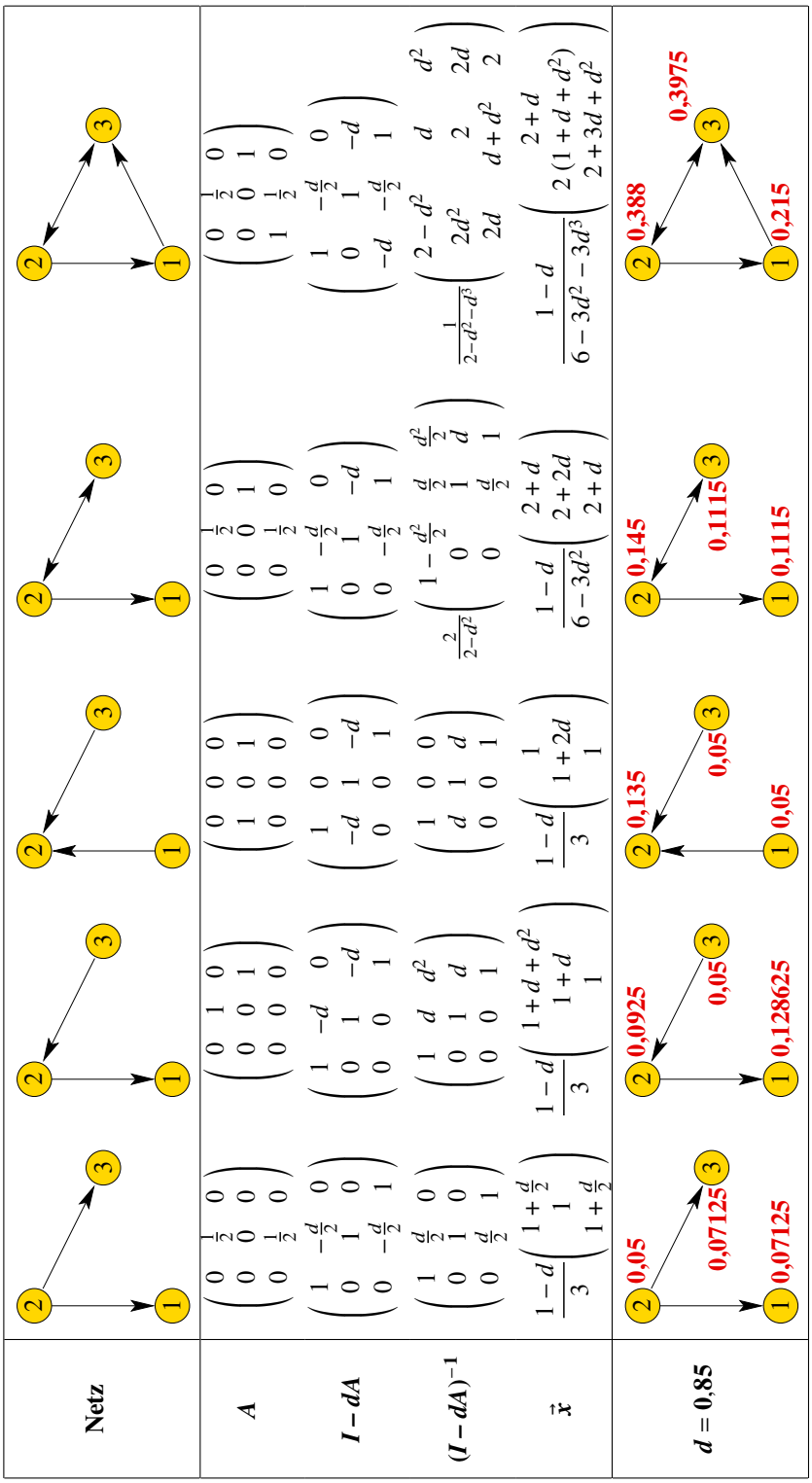


Tabelle 23.1: Beispiele einiger Netze mit $N = 3$ Webseiten und deren PageRanks für den Wert $d = 0{,}85$.

Seidel wiederentdeckte. Es wird daher heute *Gauß-Seidel-Verfahren* genannt und kann auf eine allgemeine Klasse von mehrdimensionalen Gleichungssystemen angewandt werden[13] [14], beim PageRank eben auf das Gleichungssystem (23.2). Angewandt auf den PageRank ergibt es den folgenden Algorithmus[15]:

Algorithmus 23.1: Der PageRank-Algorithmus

algorithm pageRank(A, d) {
  input: die modifizierte Adjazenzmatrix und der Dämpfungsfaktor d
  output: der Vektor $\vec{x} = (x_1, ..., x_N)$ mit dem Eintrag x_i als PagerRank der Seite i

  // 1. initialisierung aller PageRanks:
  $b \leftarrow \frac{1-d}{N}$;
  for (i: 1 **to** N) { $x_i \leftarrow b$; }
  // 2. Iterative Berechnung:
  while(gewünschte Näherung der PageRanks $\vec{x}$ nicht erreicht) {
    for (i: 1 **to** N) {
      $x_i \leftarrow b + d \cdot \sum_{j \in \{1,2,...,N\}} a_{ij} x_j$;
    }
  }
  return $\vec{x}$;
}

Hierbei bedeutet die gewünschte Näherungsgenauigkeit der PageRank-Werte als Bedingung für die äußere Schleife entweder eine zu erreichende Schwankungsbreite um einen kleinen Wert $\varepsilon > 0$ oder eine vorgegebene Anzahl an Iterationen. Das Netz kann hinreichend mit seiner modifizierten Adjazenzmatrix A repräsentiert werden. In dem Algorithmus werden für x_j die bis dahin neu berechneten Werte verwendet (also für $j < i$). Nimmt man für jede der Iterationsrunden der äußeren Schleife komplett die x-Werte der vorherigen Runde, so handelt es sich um das *Jacobi-Verfahren*, das ebenfalls zur Lösung führt[16].

Am Ende des Algorithmus hat also jede Webseite ihren festen individuellen PageRank-Wert, der im Moment der eigentlichen Suchanfrage nicht mehr berechnet werden muss. Zwei Nachteile des PageRanks sind jedoch offensichtlich[17]: Der PageRank bevorzugt Webseiten, die allgemein populär sind, egal ob sie für die Suchanfrage relevant sind. Zudem bewertet PageRank nur einzelne Seiten, nicht einen gesamten Webauftritt; so kann es dazu kommen, dass die Summe der PageRanks eines Webauftritts insgesamt sehr hoch ist, aber die Einzelseiten jeweils nicht viel verlinkt sind, so dass sie geringe PageRanks haben.

Google versucht, diese bekannten Nachteile zu umgehen. Wie genau Google die Trefferliste zu eingegebenen Begriffen bildet und sortiert, ist allerdings nicht öffentlich bekannt[18]. Neben dem PageRank-Algorithmus kommt dabei den Text-Matching-Verfahren eine besondere Rolle zu, die grob gesagt versuchen, den Übereinstimmungsgrad des Suchbegriffs mit den Informationen auf den Webseiten selbst zu bestimmen. PageRank nimmt dann eine Relevanzbewertung dieser Seitenliste vor. Google hat sowohl den Suchalgorithmus als auch den PageRank-Algorithmus

[13] http://mathworld.wolfram.com/Gauss-SeidelMethod.html
[14] Zeidler (1996):S. 1107.
[15] Brandes und Dorfmüller (2008):S. 100.
[16] Zeidler (1996):S. 1106.
[17] Kaumanns und Siegenheim (2007):S. 22.
[18] Kaumanns und Siegenheim (2007):S. 23.

kontinuierlich weiterentwickelt und so zum Beispiel ein Patent auf eine PageRank-Variante erhalten, die zusätzlich die Aktualität des Dokuments und die Historie der Links auf diese Seite bewertet. Wahrscheinlich hängt der Suchalgorithmus von Google derzeit von 200 Variablen ab, sogenannten *Signalen*[19]. Zunehmend wird dabei auf individuelle Signale Wert gelegt, die Rückschlüsse auf das Onlineverhalten des Nutzers erlauben. Zu diesem Zweck dienen Dienste wie E-Mail, Terminplanung, Fotoalben (Picasa) oder vernetzte Dokumentenspeicherung (Google Docs), für die man sich anmelden muss. Jedoch auch ein anonymer Nutzer verrät viel über sich, mit etwa 50 Signalen kann Google erkennen, wo in etwa er sich befindet, auf welche Sprache sein Rechnersystem eingestellt ist oder welches Gerät er benutzt.

PageRank wurde Mitte der 1990er Jahre von Larry Page als Student der Universität Stanford entwickelt, 1997 zum Patent angemeldet und 2001 als Patent erteilt. Das Patent gehört der Universität Stanford, bis 2011 hat Google die alleinigen Nutzungsrechte. Die Lizenz hat der Universität Stanford hunderte Millionen Dollar Einnahmen eingebracht[20].

Bemerkung 23.1. Der PageRank lässt sich als stabiler Endzustand einer Markow-Kette beschreiben, also eines stochastischen Prozesses. Bis auf eine für alle PageRanks gleiche Konstante ist x_i interpretierbar als die Wahrscheinlichkeit, sich auf der Webseite Nummer i von insgesamt N Webseiten zu befinden, wenn man eine sehr lange Zeit zufällig den Links folgt *(„Random Surfer")*, unter der Voraussetzung, dass man von einer Seite j auf jede von ihr verlinkten Seiten mit der Übergangswahrscheinlichkeit $\frac{d}{k_j^{\text{out}}} + \frac{1-d}{N}$ kommt, und mit der Wahrscheinlichkeit $\frac{1-d}{N}$ auf eine andere nichtverlinkte Seite. (Hierbei ist, wie oben, k_j^{out} der Außengrad, also die Anzahl der von der Seite j ausgehenden Links.) Mit anderen Worten: Bildet man aus dA die Matrix

$$M = \begin{pmatrix} m_{11} & \cdots & m_{1N} \\ \vdots & \ddots & \vdots \\ m_{N1} & \cdots & m_{NN} \end{pmatrix} \quad \text{mit} \quad m_{ij} = \begin{cases} \dfrac{1-d}{N} + da_{ij}, & \text{wenn } k_j^{\text{out}} > 0, \\ \dfrac{1}{N} & \text{sonst.} \end{cases} \tag{23.6}$$

so sind die Spaltensummen von M allesamt eins, d.h. M ist eine stochastische Matrix. Der Prozess des *Random Surfers* stellt eine „Markow-Kette" dar, in der er von der Webeite j mit der Wahrscheinlichkeit m_{ij} auf Webseite i kommt. Unter Ausnutzung der Identität $\vec{1}^T \cdot \vec{x} = \sum x_i = \text{const}$ kann Gleichung (23.2) umgeformt werden zu

$$M\vec{x} = \vec{x}. \tag{23.7}$$

Das ist ein sogenanntes Markow'sches Eigenvektorproblem. Beispielsweise gilt für das erste Netz in Tabelle 23.1

$$M\vec{x} = \begin{pmatrix} \frac{1}{3} & \frac{d}{2} + \frac{1-d}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1-d}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{d}{2} + \frac{1-d}{3} & \frac{1}{3} \end{pmatrix} \begin{pmatrix} 1 + \frac{d}{2} \\ 1 \\ 1 + \frac{d}{2} \end{pmatrix} = \begin{pmatrix} 1 + \frac{d}{2} \\ 1 \\ 1 + \frac{d}{2} \end{pmatrix} = \vec{x}, \tag{23.8}$$

wie man direkt (wenn auch etwas mühsam) nachrechnet. Der Lösungsvektor $\vec{x}$ ist also der Eigenvektor der Übergangsmatrix M zum Eigenwert eins. Gleiches gilt entsprechend für die anderen Netze. □

23.4 Googleware und Cloud Computing

Google has the capability to read information at a speed of around 583 megabytes a second and responds to more than 1 billion [d.h. 1 Milliarde, Anm. d. Autors]

[19] Kaumanns und Siegenheim (2007):S. 23.
[20] Kaumanns und Siegenheim (2007):S. 19.

> *queries per 24 hours. (. . .) Google has a supercomputer that delivers applications. Some of these applications are free for the user; for example, search. Other applications are for Google's 4,000 employees; for example, the programmers who craft applications for the Googleplex and employees who use the formidable number-crunching capabilities of the Googleplex to figure out what users are doing.* [21]

Google benötigt sowohl gigantische Rechenleistung als auch riesige Speicherkapazität. Realistisch ergeben sich folgende Schätzungen[22]:

- Derzeit existieren über acht Milliarden Webseiten, die jeweils mit durchschnittlich etwa 10 kB PageRank indiziert werden; das ergibt eine Speicherkapazität von etwa 80 Terabyte ($8 \cdot 10^{13}$ Byte) allein für die Indizierung.
- Pro Rechner-Cluster wird etwa ein Petabyte (10^{12} Byte) an Daten verwaltet, und jeder Cluster hat einen Datendurchsatz von 2 Gbit/sec.
- Das Rechnernetzwerk von Google besteht insgesamt aus geschätzt bis zu 450 000 Rechnern.[23]

Den letzten kompletten Systemausfall von Google hat es im Februar 2000 gegeben, er dauerte knapp eine Stunde.[24]

Wie gelingt es Google, diesen enormen Anforderungen an die Hardware zu genügen? Bereits ganz zu Anfang traten die Probleme auf, wenn auch in viel kleineren Größenordnungen. Aus Geldmangel kauften Brin und Page Einzelteile und gebrauchte PCs und bauten daraus ihre eigenen Server zusammen. Als Betriebssystem wurde von Anfang an Linux verwendet und im Rechnerverbund um ein selbstentwickeltes Dateimanagementsystem, das Google File System GFD, erweitert.

Wohl also eher aus der Not geboren, stellten sich jedoch schnell zwei sehr wichtige Vorteile dieser Herangehensweise gegenüber einem Konzept mit Superrechnern und Spezialhardware heraus: erstens die flexible Skalierbarkeit der „Googleware", also der optimalen Verzahnung von Software, Rechenleistung und Performanz, denn es brauchten im Prinzip einfach nur mehr PCs eingesetzt werden; zweitens die hohe Ausfallsicherheit, da schon aus Performanzgründen die Daten redundant gespeichert werden mussten.

23.4.1 Das Google File System GFS

Ein Webdienstanbieter wie Google oder YouTube hat prinzipiell das Problem, eine riesige Datenmenge von hunderten Terabytes auf einem Rechnercluster mit Hunderten von Rechnern und bis zu tausenden Festplatten für mehrere hundert gleichzeitig aus dem Web zugreifende Clients bereitzustellen. Traditionelle netzwerkbasierende Dateisysteme wie NFS erwiesen sich für diese Zwecke als unzulänglich, und Google entwickelte zu Beginn der 2000'er Jahre ein neues Dateisystem, das *Google File System GFS* [GFS]. Es basiert auf den folgenden Designkriterien:

- Fehler in den beteiligten Komponenten sind die Regel, nicht die Ausnahme. Es werden stets massiv Probleme festgestellt durch Bugs in Programmen, menschliche Fehler, Fehler von Betriebssystemen, Speichern, Festplatten oder Netzwerken, aber auch Stromausfälle.

[21] `http://www.hstoday.us/content/view/62/111/` [13.9.2008]

[22] Kaumanns und Siegenheim (2007):S. 25.

[23] `http://www.nytimes.com/2006/06/14/technology/14search.html?ei=5090&en=d96a72b3c5f91c47&ex=1307937600&adxnnl=1&pagewanted=2&adxnnlx=1222009358-hec9l6NoP1uJ6H3tzbIc8Q` [7.10.2010]

[24] Kaumanns und Siegenheim (2007):S. 25.

Daraus ergeben sich als Anforderungen an das Dateisystem die grundsätzlichen Eigenschaften permantentes Monitoring und Fehlererkennung, Fehlertoleranz und automatische Datenwiederherstellung.

- Die zu speichernde Dateien sind sehr groß. Die vielen einzelnen zu speichernden Objekte wie HTML-Dokumente sind zu solchen großen Dateien zusammengefasst, denn die Verwaltung von Milliarden relativ kleiner Dateien mit wenigen KB Größe wäre zu aufwändig.

- Die bei weitem meisten Dateien werden höchstens durch Anhängen neuer Objekte verändert, kaum durch Überschreiben bereits existierender Daten. Auch Schreibzugriffe innerhalb von Dateien kommen praktisch nicht vor. Sind sie einmal geschrieben, so werden sie fast nur noch gelesen, meist sogar nur sequentiell, sei es durch Programme zur Datenanalyse, zur Erzeugung von Datenströmen (z.B. Filme), zum Archivieren oder zur Zwischenspeicherung bei der verteilten Datenverarbeitung auf mehreren Rechnern.

- Hohe Übertragungsraten sind wichtiger als kurze Antwortzeiten. Die Google-Anwendungen verarbeiten vorwiegend sehr große Datenpakete *(„data bulk")* dagegen kaum zeitkritische individuelle Lese- oder Schreibzugriffe.

Architektur des GFS. Ein *GFS-Cluster* ist ein Rechner-Cluster, der aus einem einzelnen *Master* und mehreren *Chunkservers*[25] besteht (Abb. 23.4). Dateien sind in *Chunks* fester Größe (64

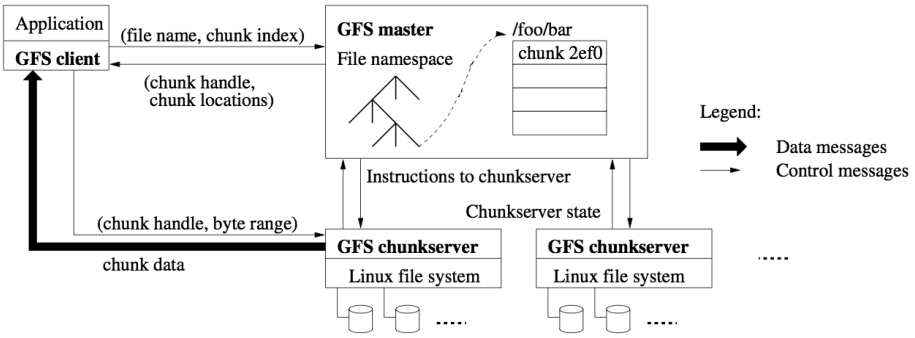


Abbildung 23.4: Die GFS-Architektur; Abb. aus [GFS].

MB) aufgeteilt, die auf den Chunkservern auf lokalen Festplatten als Linux-Dateien gespeichert werden. Jeder Chunk ist global eindeutig durch einen 64 Bit langen *Chunk-Handle* identifiziert; er wird von dem Master für jeden Chunk bei dessen Erzeugung vergeben und nicht wieder geändert. Ein Chunkserver liest und schreibt seine Chunks anhand seines Chunk-Handles und seines Bytebereichs *(byte range)*. Zum Schutz vor Datenverlust wird jeder Chunk auf standardmäßig auf drei verschiedenen Chunkservern repliziert, die Anzahl kann jedoch durch die Clientanwendung verändert werden.

Der Master verwaltet in seinem Arbeitsspeicher alle Metadaten, wie Namensraum, Zugriffsinformationen, die Zuordnung von Dateien zu Chunks und die aktuellen Speicherorte der Chunks. Er steuert sämtliche systemweiten Aktivitäten, wie Speicherplatzverwaltung der Chunks, Garbage Collection verwaister Chunks, Fehlerbehebungen oder Chunkmigrationen zwischen Chunkservern. Der Master kommuniziert mit jedem einzelnen Chunkserver regelmäßig über *HeartBeat*-Nachrichten, um Anweisungen zu geben und dessen Zustand abzufragen. Das einzige, was der Master persistent auf seine Festplatte speichert, sind Log-Dateien der

[25] *chunk* (engl.): Brocken, Klotz

Namenräume und der Datei-Chunk-Zuordnungstabellen, so dass das Gesamtsystem nach einem Absturz des Masters wiederherstellbar wird. Insbesondere werden die Speicherorte der Chunks nicht vom Master gespeichert, diese werden bei jedem Neustart des Masters oder bei Hinzufügen eines neuen Chunkservers von den Chunkservern abgefragt.

Grundsätzlich sendet der Master niemals Dateien an die Client-Anwendung, stets nur Steuerungsinformationen. Der eigentliche Datenverkehr findet nur zwischen dem Client und den Chunkservern statt. Auf diese Weise wird der Master entlastet und wird somit nicht zum Engpass, wenn mehrere Clients gleichzeitig auf Dateien zugreifen wollen. Zudem kann er seine Ressourcen vollständig für diejenigen Aktivitäten verwenden, die globales Wissen über den Cluster benötigen, wie eine intelligente Speicherverwaltung der Chunks oder Replikationsentscheidungen.

Um einen Eindruck über die Größenordnungen eines typischen Rechner-Clusters zu bekommen, sei auf Tabelle 23.2 verwiesen.

Cluster	A	B
Chunkserver	342	227
Verfügbarer Plattenplatz	72 TB	180 TB
Verwendeter Plattenplatz	55 TB	155 TB
Anzahl Dateien	735.000	737.000
Anzahl toter Dateien	22.000	232.000
Anzahl Chunks	992.000	1,55 Mio
Metadaten auf Master	48 MB	60 MB
Metadaten auf Chunkservern	13 GB	21 GB

Tabelle 23.2: Zwei typische GFS Rechner-Cluster gemäß [GFS].

Typischer Ablauf eines Lesezugriffs im GFS. In Abildung 23.4 lässt sich der folgende Ablauf eines Lesezugriffs erkennen. Zunächst berechnet der GFS-Client anhand der festen Chunkgröße und des Dateinamens und der von der Anwendung spezifizierten Offset-Adresse einen Chunk-Index. Mit den beiden Angaben Dateiname und Chunk-Index ruft er den Master auf, der den entsprechenden Chunk-Handle und die Speicherorte zurück gibt. Diese Information speichert sich der Client im Cache und ruft den Chunkserver direkt auf mit dem betreffenden Chunk und dem gewünschten Byte-Bereich als Eingabe.

Der Client kann sogar nach mehreren Chunks gleichzeitig nachfragen und erhält vom Master die Informationen so in einem einzigen Aufruf.

Energiestromdichte. Der technologische Ansatz der Googleware auf Basis von Servern aus Standard-PC-Elementen erfordert eine große Menge an Energie. Achtzig mittelgroße PCs in einem Rack, von denen ein paar Dutzend in einem Google-Rechenzentrum sind, produzieren eine physikalische Flächenleistungsdichte von 1,5 kW/m^2, haben also salopp gesprochen einen Energieausstoß von 1500 Watt je Quadratmeter Fläche. Das ist mehr als die Solarkonstante $I_0 = 1,367$ kW/m^2, also die Intensität der Sonnenstrahlung in Erdnähe außerhalb der Atmosphäre[26]. Typischerweise haben Rechenzentren eine Leistungsdichte von etwa 0,6 kW/m^2, manche bis 0,9 kW/m^2[27].

Ein typischer Google-Cluster mit einer Größe von etwa zwei Fußballfeldern *(American Football)*, also etwa 11 000 m^2, verbrauchte damit eine Leistung von 16,5 MW. Das entspräche etwa

[26] Unsöld und Baschek (1999):§6.1.3.

[27] Kaumanns und Siegenheim (2007):S. 25.

der Leistung eines kleinen Heizkraftwerks wie etwa des Kraftwerks Bochum (21 MW Nettoleistung) oder der Biomasse-Verstromungsanlage in Hagen-Kabel (20 MW). (Zum Vergleich: Das Pumpspeicherkraftwerk Koepchenwerk am Hengsteysee in Herdecke erzeugt eine Leistung von etwa 153 MW, das Steinkohlekraftwerk Werdohl-Elverlingsen der Mark-E hat eine Nennleistung von etwa 693 MW.)[28]

23.5 Maschinelles Lernen mit TensorFlow

TensorFlow ist eine quelloffene Programmierschnittstelle für verteilte Systeme mehrerer CPUs und GPUs, die maschinelles Lernen mit Hilfe neuronaler Netze ermöglicht. Es basiert auf Python und ermöglicht die Programmierung hardwarenaher Datenoperationen mit C++, soll aber auch für Java und R verfügbar werden [TF]. Ursprünglich von Google intern entwickelt, wurde es im November 2015 auf GitHub veröffentlicht.

Neuronale Netze sind Netzwerke, die nach Vorbild biologischer Gehirne aus simulierten Neuronen als Knoten und Synapsen als Kanten bestehen. Ein einzelnes Neuron kann dabei ein Signal an alle seine benachbarten Neuronen feuern, wenn es seinerseits eine genügend große Signalmenge von anderen Neuronen empfangen hat. Neuronale Netze werden auf dem Gebiet der Künstlichen Intelligenz (KI) bereits seit vielen Jahre betrachtet, konnten jedoch lange Zeit aufgrund der hohen benötigten Rechenressourcen kaum eingesetzt werden.

Ein erfolgreicher Ansatz wird mit dem so genannten das *Deep Learning* verfolgt, bei dem mehrere neuronale Netze in Schichten hintereinander geschaltet werden. Jede dieser Schichten führt dabei eine spezialisierte Teilaufgabe wie das Wahrnehmen von Signalen aus der Umwelt oder Erkennen von Strukturen. Zum Beispiel kann ein neuronales Netz als erste Ebene der Bilderkennung sich mit einzelnen Pixeln und der Erkennung von Helligkeit und Farbe beschäftigen, während ein weiteres Netz als nächste Ebene zur Erkennung von Linien, Kanten und Flächen verwendet wird. Deep Learning ist die Basis für digitale Assistenten, die Kommunikation verstehen sollen, wie Apples Siri, Cortana von Microsoft oder Google Now.

Beispiel 23.2. *(Zeitliche Ebenen der Spracherkennung)*[29] In einer Unterhaltung stehen meist mehrere aufeinanderfolgende Sätze in einem inhaltlichen Zusammenhang. Informationen referenzieren dabei auf vorher Gesagtes und manchmal auch auf Sachverhalte, die erst später gesagt werden: „Er kam auf mich zu. Er war klein, hieß Felix und bellte laut.“ In diesen beiden Sätzen wird erst am Ende aufgelöst, dass es sich um einen Hund handelt. Entsprechend wandelt sich nachträglich unser Verständnis des ersten Satzes „Er kam auf mich zu“. □

TensorFlow löst Google-intern nach und nach das Projekt DistBelief ab und soll für alle Sparten wie Search, AdWords, YouTube und Gmail eingesetzt werden[30]. TensorFlow erlaubt es, beliebige neuronale Netze durch gerichtete kreisfreie Graphen darzustellen, sogenannte *Data Flow Graphs*. Die Kanten bilden die Eingabe und die Ausgabe der einzelnen Rechenschritte ab, die Knoten die jeweilige Verarbeitung der Eingaben zur Ausgabe. Um TensorFlow zu verwenden, muss ein Programm einen solchen Graphen konstruieren.

Innerhalb von TensorFlow werden die Daten als mehrdimensionale Arrays gespeichert. Solche Gebilde heißen in der Mathematik *Tensoren*. Um Input zu erzeugen, lässt sich beispielsweise die gesprochene Sprache über Sampling, also dem Abgreifen von Klangwerten in kurzen periodischen Abständen, in einen Vektor überführen, d.h. einen „Tensor 1. Stufe“. Entsprechend

[28] `https://de.wikipedia.org/wiki/Kraftwerk_Bochum`, `http://www.standardkessel-baumgarte.com/neuanlagen-und-komponenten/referenz/bva-hagen-kabel.html`, `https://de.wikipedia.org/wiki/Koepchenwerk`, `https://de.wikipedia.org/wiki/Kraftwerk_Werdohl-Elverlingsen` [2016-01-25]

[29] Fondermann (2016).

[30] Fondermann (2016).

lässt sich ein Schwarz-Weiß-Bildausschnitt als Matrix der Pixel darstellen, d.h. als ein „Tensor 2. Stufe“, und ein Farbbild als drei solcher Pixelmatrizen, also ein „Tensor 3. Stufe“. Tensoren sind insbesondere interessant, da Grafikkarten darauf optimiert sind, sehr schnell extrem viele Berechnungen auf ihnen ausführen zu können. Daher unterstützt TensorFlow GPU-Computing.

24 Facebook

Kapitelübersicht

Wie kann man einem etablierten Suchdienst Konkurrenz bieten? Für ein knappes Jahrzehnt schien keine Antwort auf das Monopol der Firma Google möglich zu sein, aber etwa seit 2010 erscheint die Idee realistisch, ein Parallelweb neben dem Internet aufzubauen, auf das Suchmaschinen keinen wesentlichen Zugriff haben und das die Eigenschaften und Vorlieben seiner Nutzer kennt. Weltweit hat Facebook seit 2012 mehr als 1 Milliarde Mitglieder,[1] das

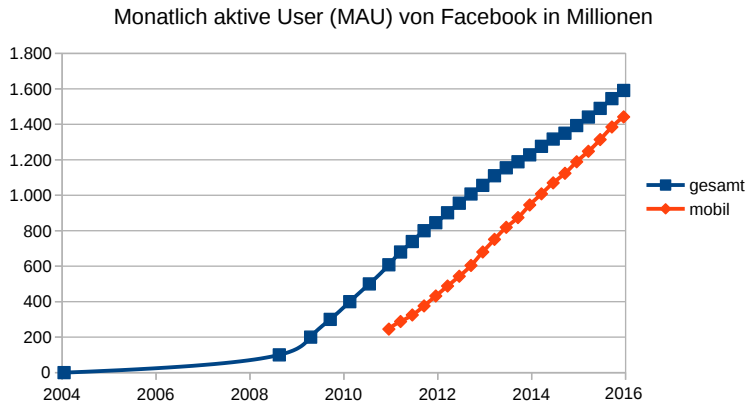


Abbildung 24.1: Monatlich aktive User (MAU) von Facbook. Quellen: `http://en.wikipedia.org/wiki/Facebook`, Annual Reports 2012 (S. 39f), 2014 & 2015 (jeweils S. 35f), `http://investor.fb.com/sec.cfm` [2016-02-08]

heißt mehr als die Hälfte aller Internetnutzer weltweit ist aktiv in Facebook. In den USA, Südamerika und Afrika sollen es über 80% der Internetnutzer sein, in Europa etwa 65% und in

[1] `http://newsroom.fb.com/Key-Facts` [2012-11-12]

Asien 25%[2]. Im Juni 2014 hatte Facebook über 1,3 Milliarden aktive Nutzer.[3]

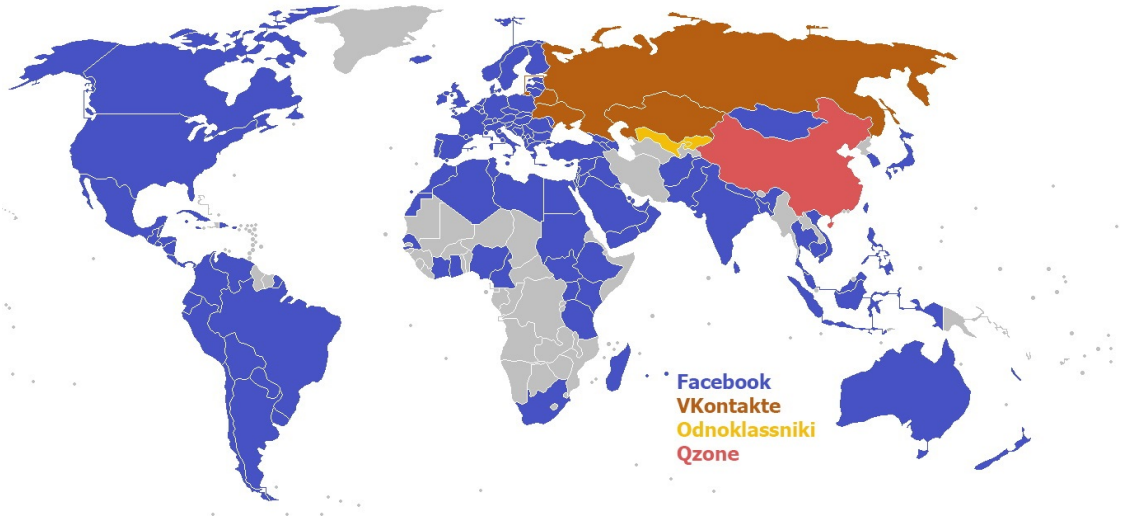


Abbildung 24.2: Die in den jeweiligen Ländern beliebtesten sozialen Netze. Quelle: `http://commons.wikimedia.org/wiki/File:Social_networks.jpg` [2015-04-08]

Facebook kaufte am 19. Februar 2012 den Instant-Messaging-Dienst WhatsApp für etwa 19 Milliarden US-Dollar und im April 2012 den Foto- und Video-Sharing-Dienst Instagram für 1 Milliarde US-Dollar. Seit dem 18. Mai 2012 ist Facebook an der Börse notiert.

24.1 Geschäftsmodell

„Our mission is to give people the power to share and make the world more open and connected. Our business focuses on creating value for people, marketers, and developers.“[4] Nach Sheryl Sandberg, Geschäftsführerin (COO) von Facebook, hat Facebook eine klare Vision: „Wir wollen, dass jeder, der Produkte herstellt, Facebook nutzt.“ Der Facebook-Auftritt soll also den Webauftritt eines Unternehmens ersetzen. Tatsächlich wird in vielen Fernsehwerbespots nicht mehr die Webadresse angezeigt, sondern nur noch das Logo von Facebook, ein kleines weißes „f“ auf blauem Quadrat. Dieser Button ist das Kernelement von Facebooks Geschäftsmodell. Er hilft dem Unternehmen, sämtliche Vorlieben seiner Nutzer auf der eigenen und auf fremden Seiten zu erfassen, die ihn eingebettet haben. Da Facebook eher ein Forum ist, auf dem Unternehmen mit Nutzern in einen Dialog treten können, die wiederum das Netzwerk eher privat nutzen, um mit Freunden, Bekannten oder Interessenverwandten in Verbindung zu treten, ist klassische Werbung dort allerdings eher unerwünscht. Damit muss Facebook den Konflikt bewältigen, einerseits das attraktive Werbeumfeld für die Unternehmen auszureizen, andererseits den Nutzern eine Atmosphäre zu bieten, in der sie sich gern lange aufhalten und vertrauensvoll viel von sich preisgeben.

Seit Anfang 2012 setzt Facebook *Featured Ads* ein, also hervorgehobene Anzeigen. Marken werden dadurch Freunden gleichgestellt und erscheinen in deren *Newsfeed*, dem zentralen Feld

[2] Rink (2012):S. 38.

[3] `http://newsroom.fb.com/company-info/` [2015-04-07]

[4] Facebook Annual Report 2014, p. 5 `http://investor.fb.com/sec.cfm` [2015-04-10]; gegenüber der Einleitung zu Geschäftsbericht 2013 fehlt übrigens der Absatz *“We build technology to enable faster, easier and richer communication. Hundreds of millions of people use Facebook’s websites and mobile applications every day to stay connected with their friends and family, to discover and learn what is going on in the world around them, and to share and express what matters to them to the people they care about.”*

des eigenen Profils, das Neuigkeiten anzeigt. Sie sind daher für die Nutzer des Netzwerks an der zentralen Stelle sichtbar, an der sie die meiste Zeit verbringen: Neben der Mitteilung, wer mit wem befreundet ist, steht dort auch eine Firmenwerbung, sofern der Nutzer oder seine Freunde zuvor den „Gefällt mir"-Button für die Marke gedrückt haben. Facebook bewertete bis etwa 2011 die Einträge mit dem EdgeRank-Algorithmus (`http://edgerank.net`) mit drei Bewertungskriterien pro Eintrag, mittlerweile ist er durch einen bislang nie veröffentlichten Algorithmus mit angeblich hunderttausenden von Parametern pro Eintrag ersetzt worden.[5]

Ein zentrales von Facebook eingeführtes und in den Geschäftsberichten als betriebswirtschaftliche Kennzahl verwendetes Maß ist MAU (*monthly active users*), also die Anzahl monatlich aktiver User, definiert als die Anzahl in Facebook registrierter User, die in den jeweils letzten 30 Tagen des Messdatums in Facebook eingeloggt waren und Facebook über dessen Webseite oder ein Mobilgerät besucht oder eine Aktion zum Teilen von Inhalten oder Aktivitäten mit den Facebook-Kontakten unternommen haben. *"MAUs are a measure of the size of our global active user community."*[6]

24.2 WhatsApp und die Internet.org Vision

Am 19. Februar 2012 kaufte Facebook für etwa 19 Milliarden Dollar das US-Unternehmen WhatsApp Inc., das 2009 in Santa Clara, Kalifornien, von Jan Koum und Brian Acton gegründet wurde. Das Unternehmen bietet den internetbasierten Instant-Messaging-Dienst *WhatsApp* auf Basis von XMPP an, der den Austausch von Textnachrichten, Standortinformationen sowie Bild-, Video- und Ton-Dateien zwischen Benutzern von Mobilgeräten. Ein Benutzer des Dienstes muss hierzu die App *WhatsApp Messenger* als Client auf seinem Endgerät installieren und sich mit seiner Telefonnummer am Server von WhatSapp registrieren. WhatsApp Inc. betreibt den Server des Dienstes und entwickelt und vertreibt die Clientanwendung *WhatsApp Messenger*. Sie liest Daten aus dem Adressbuch des Telefons aus und gleicht sie auf dem Server ab. Informationstechnisch ist der Dienst WhatsApp ein auf einer angepassten Version von XMPP basierendes Netzwerk und stellt die Kommunikation zwischen Sendern und Empfängern über den Server `1234@s.whatsapp.net` her. Die Kennung eines Benutzers ist dessen Telefonnummer, also nach XMPP-Format z.B.

`01234@s.whatsapp.net.`

Im Juni 2009 wurde mit WhatsApp 2.0 für das iPhone die erste Version mit einer Nachrichtenkomponente in Apples App Store veröffentlicht, durch die die Zahl der Netzwerkteilnehmer schnell auf 250 000 stieg.[7] Im Januar 2010 wurde eine App für BlackBerrys veröffentlicht, im Mai 2010 für Symbian OS und im August 2010 für Android OS.

Etwa 200 Millionen aktive Benutzer waren im Februar 2013 erreicht, im April 2014 bereits 500 Millionen mit insgesamt etwa 10 Milliarden Nachrichten pro Tag.[8] Zu Beginn des Jahres 2015 waren 700 Millionen Benutzer aktiv und versendeten täglich über 30 Milliarden Nachrichten.[9] Für die zeitliche Entwicklung der monatlich aktiven User (MAU)[10] siehe Abbildung

[5] `http://marketingland.com/edgerank-is-dead-facebooks-news-feed-algorithm-now-has-close-to-100k-weight- factors-55908` [2016-01-25]

[6] Facebook 2012 Annual Report, p. 36 `http://investor.fb.com/downloads.cfm` [2016-02-09]

[7] `http://www.forbes.com/sites/parmyolson/2014/02/19/exclusive-inside-story-how-jan-koum-built-whatsapp- into-facebooks-new-19-billion-baby/`

[8] `http://www.forbes.com/sites/amitchowdhry/2014/04/22/whatsapp-hits-500-million-users/`

[9] `http://www.businessinsider.in/WhatsApps-Insane-Growth-Continues-100-Million-New-Users-in-4-Months/articleshow/45786867.cms`

[10] `http://en.wikipedia.org/wiki/Monthly_active_users`

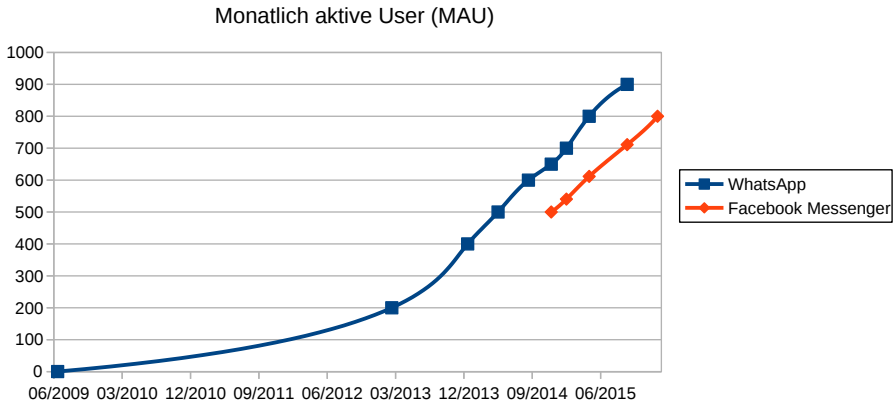


Abbildung 24.3: WhatsApp: Zeitliche Entwicklung der Anzahl monatlich aktiver User (MAU). Quelle: Wikipedia `http://en.wikipedia.org/wiki/WhatsApp#History` [2016-01-25]

24.3. Seit Anfang 2013 ist demnach ein nahezu lineares Wachstum zu erkennen, ein deutlich stärkeres Wachstum als von 2009 bis 2013. Die erheblichen Sicherheitslücken, die im Laufe der Zeit entdeckt wurden, haben der Verbreitung des Dienstes also keinen Abbruch getan. So wurde im Mai 2011 eine Sicherheitslücke entdeckt, die die Übernahme von Benutzerkonten durch Angreifer ermöglichte,[11] im Januar 2012 eine weitere, die es einem Angreifer ermöglichte, bei Kenntnis lediglich der Telefonnummer, den Status des angegriffenen Kontos zu ändern,[12] und im Juli 2013 eine, die es im Zahlungsprozess erlaubte, in den Besitz von Zahlungsdaten von Google Wallet oder Paypal zu kommen.[13]

Im Februar 2014 gab Zuckerberg als Grund für den bislang größten Kauf in der Unternehmensgeschichte Facebooks an, dass WhatsApp eine Schlüsselrolle in dem von Facebook geführten Projekt Internet.org spielt, das als Ziel die Anbindung der Entwicklungsländer an das Internet hat.[14]

24.2.1 Facebook Messenger

Facebook Messenger ist eine Anwendung für Text- und Audio-Kommunikation, die Facebook am 9. August 2011 für iOS und Android eingeführt hat und am 11. Oktober 2011 für BlackBerry OS und am 5. März 2014 für Windows Phone[15] verfügbar waren. Technisch basiert es auf dem offenen Nachrichten-Protokoll MQTT und integriert Facebooks Web-Chat-Funktion.[16] Am 11. November 2014 berichtete das Unternehmen von 500 Millionen, am 7. Januar 2016 von bereits 800 Millionen Nutzern.[17] Das sind fast soviele Nutzer wie WhatsApp.

Am 28. April 2015 startete Facebook eine Konkurrenz für Skype bzw. Facetime, denn ab

[11] `http://thenextweb.com/apps/2011/05/23/signup-goof-leaves-whatsapp-users-open-to-account-hijacking/`

[12] `http://www.iphone-ticker.de/whatsapp-sicherheitslucke-erlaubt-status-anderungen-fremder-nummern- entwickler-ignorieren-hinweise-29416/`

[13] `http://www.zdnet.de/88163371/sicherheitsfirma-uber-whatsapp-lucke-lassen-sich-paypal-und-google- konten-ausspionieren/`

[14] `http://www.techradar.com/news/internet/web/mark-zuckerberg-whatsapp-is-worth-more-than-19-billion-1227925`, `http://techcrunch.com/2014/02/24/whatsapp-is-actually-worth-more-than-19b-says-facebooks-zuckerberg/` [2016-01-25]

[15] `http://www.engadget.com/2014/03/04/facebook-messenger-arrives-for-windows-phone/`

[16] `https://www.facebook.com/notes/facebook-engineering/building-facebook-messenger/10150259350998920`

[17] `http://www.bbc.com/news/technology-29999776`, `http://www.heise.de/newsticker/meldung/Facebooks-Messenger-hat-jetzt-800-Millionen-Nutzer-3065800.html`

diesen Zeitpunkt kann man über den Messenger in 18 Testländer (Deutschland ist nicht dabei) kostenlose Videoanrufe starten

24.3 Wirtschaftliche Kennzahlen

Facebook Inc. ist ein Unternehmen mit Sitz in Menlo Park, Kalifornien, das am 4. Februar 2004 von Mark Zuckerberg und seinen Kommilitonen Eduardo Saverin, Dustin Moskovitz and Chris Hughes an der Harvard-Universität in Cambridge, Massachusetts, gegründet wurde. Der Umsatz betrug 2012 insgesamt 5,1 Milliarden US$, davon 89% durch Werbung, und davon wiederum

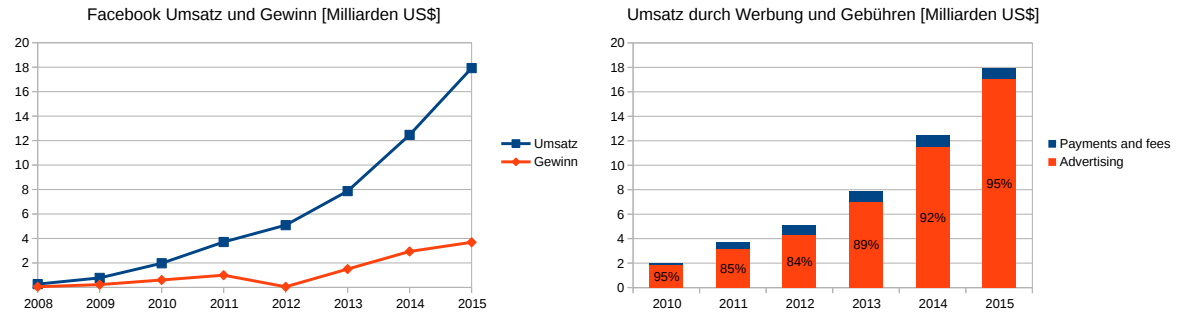


Abbildung 24.4: Umsatz und Gewinn von Facebook. Quelle: Facebook Annual Reports 2012–2015 `http://investor.fb.com/sec.cfm` [2016-02-08]

über die Hälfte von außerhalb der USA (Abbildung 24.4). Daneben konnte Facebook von 2008 bis 2013 über Zahlungen von Anwendungen von Fremdanbietern in seiner Kunstwährung „Facebook Credits“ (10 Credits = 1 US$) bei einer Umsatzbeteiligung von 30% je Einkauf etwa 557 Millionen US$ umsetzen[18]. Zum 12. September 2013 stellte Facebook die Währung wieder ein.

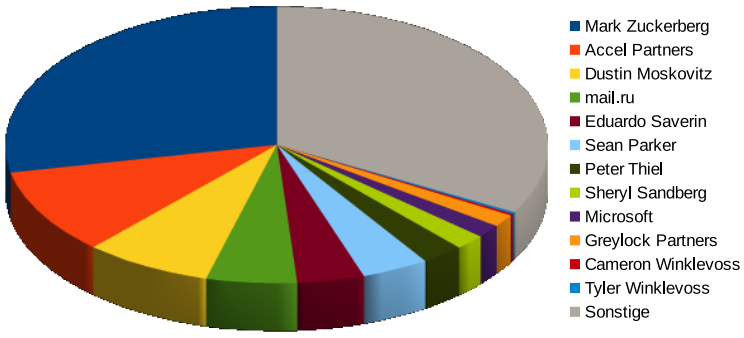


Abbildung 24.5: Facebook: Eigentümerstruktur im Mai 2012. Quelle: Handelsblatt `http://www.handelszeitung.ch/bildergalerie/die-facebook-eigner` [2016-01-25]

[18] Rink (2012):S. 30.

24.4 Informationstechnik

Die Website von Facebook ist in PHP programmiert, das mit dem von Facebook entwickelten Konvertierungsprogramm *HipHop for PHP* optimiert nach C++ transformiert wird.[19] Seit März 2014 wird zunehmend die Programmiersprache *Hack* verwendet, die Facebook selbst entwickelt und im März 2014 veröffentlicht hat. Die Syntax von Hack ähnelt derjenigen von PHP, allerdings werden die Programme nach dem Vorbild der Java VM in einen Bytecode kompiliert, der auf der HipHop Virtual Machine (HHVM) abläuft. Weitere Details siehe unter `http://hacklang.org/`.

Zusammen gefasst ist Facebook also eine monolithische Anwendung, der gesamte PHP-Quelltext wird in eine einzige binäre Datei kompiliert.[20] Insbesondere werden alle Daten und die Kommunikation zwischen Mitgliedern zentral von Facebook kontrolliert und durchgeführt. Zur Verarbeitung der Daten werden Java-Programme von PHP aufgerufen und mit HBase in Rechnerclustern verteilt gespeichert,[21] bis 2011 mit Apache Cassandra und MySQL.[22]

Das Betriebssystem von Facebook ist CentOS, eine für Großunternehmen konzipierte freie Linux-Distribution. Für Entwickler bietet Facebook auf seiner Webseite `https://developers.facebook.com/docs/` APIs und SDK's für JavaScript und PHP, für die gängigen App-Programmiersprachen, für die Spiel-Engine Unity und für Apple TV an.

[19] `https://developers.facebook.com/blog/post/2010/02/02/hiphop-for-php--move-fast/` [2016-01-25]

[20] `http://arstechnica.com/business/2012/04/exclusive-a-behind-the-scenes-look-at-facebook-release-engineering/1/` [2015-04-09]

[21] `http://highscalability.com/blog/2011/3/22/facebooks-new-realtime-analytics-system-hbase-to-process-20.html` [2015-04-09]

[22] `http://mvdirona.com/jrh/TalksAndPapers/KannanMuthukkaruppan_StorageInfraBehindMessages.pdf` [2015-04-09]

I'm also curious about whether there is a fundamental mathematical law underlying human social relationships that governs the balance of who and what we all care about. I bet there is.

Mark Zuckerberg am 30. Juni 2015 (*https://facebook.com/zuck/posts/10102213601037571*)

25 Netzwerkeffekte und Netzwerkanalyse

Kapitelübersicht

25.1 Definition elektronischer sozialer Netzwerke

Ein *soziales Netzwerk* ist definiert als eine endliche Menge von Akteuren und die Menge der direkten Beziehungen zwischen ihnen. Ein Akteur kann hierbei beispielsweise ein Individuum sein, eine Gruppe oder auch eine Organisation, und die direkte Beziehung zwischen zwei Akteuren zeigt an, ob sie direkt miteinander interagieren, direkten Kontakt haben oder sozial verbunden sind durch zufällige Bekanntschaft oder Verwandtschaft[1]. Ein soziales Netzwerk

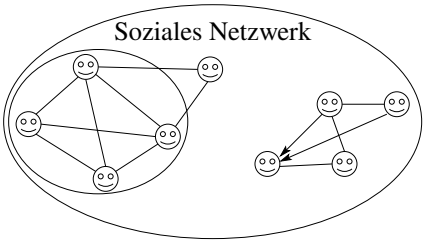


Abbildung 25.1: Ein soziales Netzwerk als Graph. Der innere Kreis umschließt eine Clique.

[1] Barnes (1954); Wasserman und Faust (1994).

kann also natürlicherweise durch einen Graph dargestellt werden, in dem jeder Knoten einen Akteur wiedergibt und jede Kante eine direkte Beziehung. In der Graphentheorie spricht man statt von der „Bekanntschaft“ von dem *Grad* eines Knotens. In sozialen Netzwerken heißt eine bidirektionale Beziehung *Kontakt*, eine gerichtete Kante wird oft *Link* genannt.

Eine *Clique* ist eine Gruppe von Akteuren, in der jeder jeden kennt. Stellt der gesamte Graph eine Clique dar, so heißt er *vollständig*.

Empirisch scheint die durchschnittliche Anzahl der Beziehungen eines Individuums in einem biologischen sozialen Netzwerk positiv korreliert zu sein mit der Größe seines Neocortex, dem evolutionsgeschichtlich jüngsten Teil des Großhirns. In menschlichen (nichtelektronischen) sozialen Netzwerken beträgt die maximale Anzahl direkter Beziehungen etwa 150 Personen („Dunbar'sche Zahl“), und die durchschnittliche Anzahl etwa 124 Personen.[2]

Ein *elektronisches soziales Netzwerk* ist definiert als ein Netzwerk von mindestens drei Akteuren, die wesentlich, wenn auch nicht ausschließlich, elektronische Geräte und Medien zur Kommunikation verwenden. Beispiele für elektronische soziale Netze sind Internet Communities wie Facebook oder XING, aber auch Onlinespiele oder virtuelle Welten wie World of Warcraft oder FarmVille.

25.2 Netzwerkstrukturen

25.2.1 Zufallsnetze und skalenfreie Netze

Die Theorie der Netzwerke ist die von Euler 1736 mit der Lösung des Königsberger Brückenproblems begründete Graphentheorie. Zwei wesentliche Klassen von Netzwerken bilden die Zufallsnetze und die skalenfreie Netze. Zufallsnetze sind mathematisch gut beschreibbar, kommen jedoch in der Realität selten vor. In der Realität sind skalenfreie Netze die Regel, sie sind

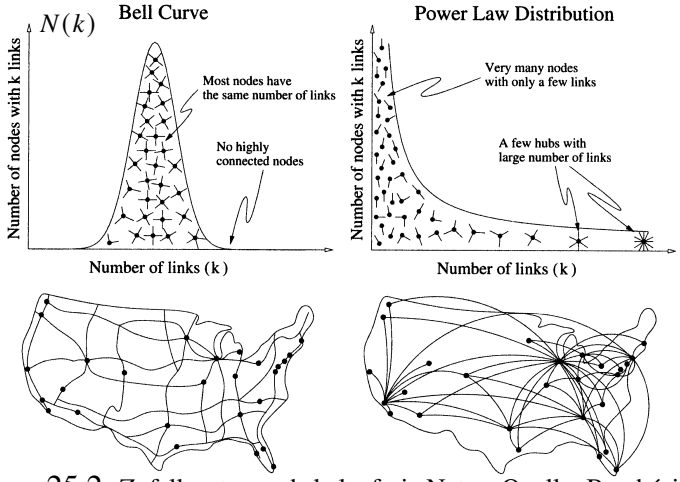


Abbildung 25.2: Zufallsnetze und skalenfreie Netze. Quelle: Barabási (2003:S. 71)

mathematisch jedoch schwerer zu beschreiben und zu identifizieren. Der Begriff des Zufallsgraphen entstand durch den russischen Mathematiker und Biologen Anatol Rapoport Anfang der 1950er Jahre, bevor Erdős und Rényi 1960 die heute nach ihnen benannte Kategorie von Graphen einführten: Ein *Zufallsnetz* oder *Erdős-Rényi-Graph* ist eine Menge von isolierten Knoten, die sukzessive paarweise mit einer gleichverteilten Wahrscheinlichkeit verbunden werden. In einem Zufallsgraphen ist also die Anzahl k der direkten Kontakte der einzelnen Knoten binomialverteilt um einen Durchschnittswert, siehe Abb. 25.2 links. In der Graphentheorie spricht man bei

[2] Hill und Dunbar (2003).

der Anzahl der direkten Kontakte eines Knotens von seinem *Grad*[3],[4],[5].

Der Mittelwert k der Kontakte pro Knoten ist für real existierende Netzwerke (Kontakte, Internet, Ausbreitung von Epidemien oder Computerviren, Verkehrsnetze, . . .) allerdings nicht besonders aussagekräftig: Solche Netzwerke bauen sich zumeist mit der Zeit sukzessive auf und weisen eine Struktur aus, in der einige Knoten mehr Verbindungen haben als andere. Solche Knoten heißen *Hauptknotenpunkte (hubs)*. Typische Vertreter solcher Netze sind *skalenfreie* oder *skaleninvariante Netze*, die man am besten mithilfe der Anzahl $N(k)$ der Knoten mit k Beziehungen charakterisiert, siehe Abbildung 25.2. Im Gegensatz zu einem Zufallsnetz, in

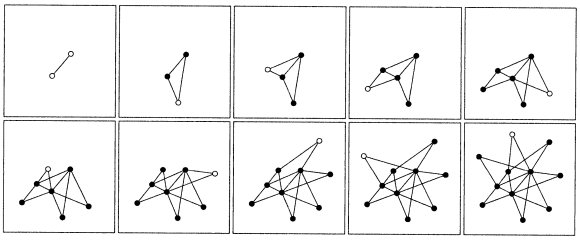


Abbildung 25.3: Entstehung eines skalenfreien Netzes nach dem Barabási-Albert-Modell. In jedem Schritt fügt sich ein neuer Knoten (leerer Kreis) in das Netz ein, indem er sich bevorzugt mit bereits gut vernetzten Knoten verknüpft. Quelle:[6]

dem die Funktion $N(k)$ eine Binomialverteilung („Glockenkurve“) um den Mittelwert $\bar{k}$ der Beziehungen darstellt, zeichnet sich die Verteilungsfunktion eines skalenfreies Netzes durch ein Potenzgesetz

$$N(k) \propto k^{-\gamma} \qquad (k \gg 1) \tag{25.1}$$

für ein $\gamma > 0$ aus: viele Knoten mit sehr wenig Beziehungen, wenige mit vielen Beziehungen. Die Bezeichnung „skalenfrei“ rührt daher, dass auch bei Vergrößerung oder Verkleinerung der Knotenzahl die Verteilung gleich bleibt. Die Ursache ist die Neigung neu hinzukommender Knoten, sich eher mit bereits gut vernetzten Knoten zu verknüpfen (Abbildung 25.3). Dieses Bildungsprinzip wird auch *Simon-Mechanismus* oder *preferential attachment* genannt[7].

Zahlreiche reale Netzwerke sind skalenfrei, so beispielsweise das Web mit seiner Hyperlinkstruktur[8], das Netzwerk von Filmschauspielern, die miteinander Filme gemacht haben, und die Zitiernetze wissenschaftlicher Publikationen. Sie weisen alle einen Exponenten von $\gamma = 2.3$ auf[9].

Bemerkung 25.1. In der Mathematik spricht man bei der diskreten Wahrscheinlichkeitsverteilung

$$P_{n,\gamma}(k) = \frac{c_{n,\gamma}}{k^{\gamma}} \tag{25.2}$$

mit den Parametern $n \in \mathbb{N}$ und $\gamma \in (0, \infty)$ und der Konstanten $c_{n,\gamma} = 1/\sum_k^n k^{-\gamma}$ von einer *Zipf-Verteilung* für eine Grundgesamtheit der Größe n und der Potenz γ. Hierbei bezeichne $P_{n,\gamma}(k)$ die Wahrscheinlichkeit, dass ein Knoten den Grad k hat. Die Zipf-Verteilung ist damit die diskrete Variante der Pareto-Verteilung. Für $\gamma > 1$ ist der Grenzwert $n \to \infty$ wohldefiniert und es gilt

$$P_{\gamma}(k) = \frac{1}{\zeta(\gamma)} \frac{1}{k^{\gamma}} \tag{25.3}$$

[3]Diestel (2000):§1.2.

[4]Kaderali und Poguntke (1995):§4.1.8.

[5]Newman (2010):§6.9, (6.19) & (6.25).

[7]Newman (2010):§14.1.

[8]Die Asymmetrie, dass eine Webseite auf eine andere verweist, aber nicht umgekehrt, ändert daran nichts Wesentliches (Newman, Barabási et al. (2006):S. 338)

[9]Newman, Barabási et al. (2006):S. 335f.

für $k = 1, 2, 3, \ldots$, wobei $\zeta(\gamma)$ die Riemann'sche Zetafunktion

$$\zeta(\gamma) = \sum_{k=1}^{\infty} \frac{1}{k^\gamma} \tag{25.4}$$

ist. Vgl.[10]. □

25.2.2 Kleine-Welt-Netze

In einem sozialen Netzwerk mit n Teilnehmern habe jeder im Durchschnitt k Kontakte. Dann sind im Schnitt k^2 Teilnehmer über einen Teilnehmer dazwischen verbunden. Mit anderen Worten sind also insgesamt etwa k^2 Teilnehmer über zwei Links mit einem gegebenem Teilnehmer verbunden. Allgemeiner sind unter der Annahme, dass die Bekanntschaften sich kaum überschneiden, von einem gegebenem Teilnehmer etwa k^d Teilnehmer d Links entfernt, jeder hat also k^d *Kontakte d-ten Grades*. Andersherum kann in einem solchen Netzwerk mit n Teilnehmern die Anzahl der k^d Kontakte d-ten Grades ja höchstens n sein, [11] also $k^d \lesssim n$, oder äquivalent

$$d \lesssim \log_k n. \tag{25.5}$$

Das bedeutet, dass ein Teilnehmer jeden anderen Teilnehmer des Netzwerks über höchstens $\log_k n$ Grade kennt. Natürlich ist in realen sozialen Netzwerken die Überschneidungen der

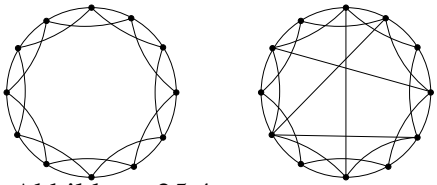


Abbildung 25.4: Kleine-Welt-Netze.

Bekanntschaften der Teilnehmer nicht selten, es gibt sehr viele Cliquen, Cluster und Quervebindungen[12]. Die Abschätzung (25.5) ist dennoch richtig, wenn man k eben so wählt, dass es die durchschnittliche Anzahl der Kontakte bezeichnet, die sich nicht gegenseitig kennen.

Beispiel 25.2. Für die Menschheit mit $n = 7 \cdot 10^9$ Personen und durchschnittlich $k = 50$ Kontakten je Person (ohne Bekanntschaftsüberschneidungen) folgt $d < 6$: Über höchstens sechs Grade ist also jeder mit jedem auf dieser Erde bekannt![13] Interessanterweise stimmt das mit den Bekanntschaftsgraden in Twitter überein.[14] □

Man definiert die *Distanz* zwischen zwei Knoten in einem Graphen als die kürzest mögliche Entfernung zwischen ihnen. Bezeichnet d die maximale Distanz zwischen zwei Knoten eines Netzes („Durchmesser des Netzes"[15]) oder alternativ die durchschnittliche Distanz zweier Knoten eines Netzes mit n Knoten, so drückt die logarithmische Beziehung (25.5) das *Kleine-Welt-Phänomen (small world phenomenon)* aus, vgl. Abbildung 25.4. Ein Netzwerk, das diese Eigenschaft besitzt, heißt entsprechend *Kleine-Welt-Netz*. Nicht alle Netzwerke sind Kleine-Welt-Netze. Empirisch belegt ist es zwar für viele reale Netzwerke, beispielsweise das Web, Stoffwechselnetze oder Zitiernetzwerke[16]. Auch in den webbasierten sozialen Netzwerken ist es oft zu sehen, wie die Kontaktzahlen eines typischen Nutzers in Abbildung 25.5 zeigen.

[10] Newman (2010):§§8.4.2, 13.1.2.

[11] Genau genommen gilt $k^d \leq n - \sum_{i=0}^{d-1} k^i = n - \frac{k^d - 1}{k-1}$, also $k^{d+1} \leq n(k-1) + 1 \leq nk$, oder eben $k^d \leq n$.

[12] Easley und Kleinberg (2010):§20.2.

[13] Barabási (2003):S. 29.

[14] `http://www.sysomos.com/insidetwitter/sixdegrees/` [2012-10-23]

[15] Diestel (2000):S. 9.

[16] Newman, Barabási et al. (2006):§3.

Abbildung 25.5: Kontakte unterschiedlichen Grades des Autors im Netzwerk XING.

Beispiel 25.3. *(Kleine Welt Facebook)* In einer Veröffentlichung vom 4. Februar 2016 schätzten fünf Wissenschaftler von Facebook statistisch ab, dass in diesem sozialen Netzwerk jeder der (damals) knapp 1,6 Milliarden Mitglieder von allen anderen im Durchschnitt nur durch 3,57 Kontakte getrennt ist,[17] siehe Abbildung 25.6. Fünf Jahre vorher, 2011, waren es im Durchschnitt

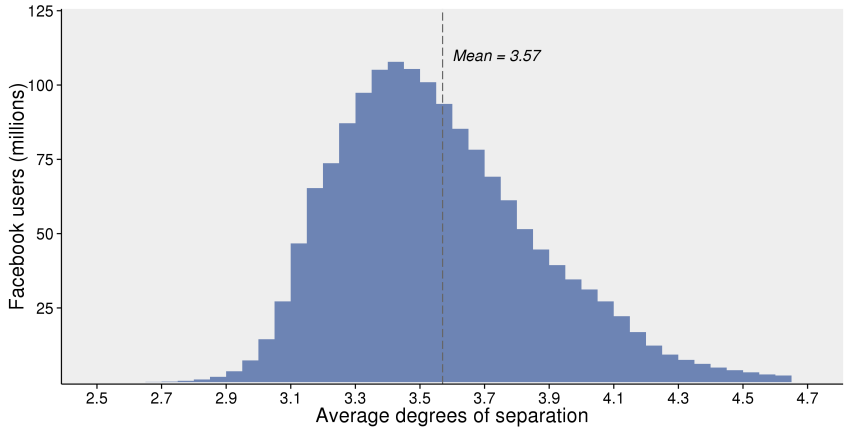


Abbildung 25.6: Geschätzter mittlerer Abstand der Facebookmitglieder zu allen anderen Mitgliedern. Bildquelle:[18].

3,74 Kontakte bei 721 Millionen Mitgliedern[19]. □

Beispiel 25.4. *(Gehirn)* Die Neurowissenschaftler Patric Hagmann und Olaf Sporns kartierten in den 2000er Jahren die Verknüpfungen der Neuronen des menschlichen Gehirns. Dabei zeigte sich, dass das entsprechende Netzwerk nicht gleichmäßig ist, also kein Zufallsnetz ist, sondern skalenfrei ist. Es gibt also einige besonders stark vernetzte Hauptknotenpunkte, über die ein

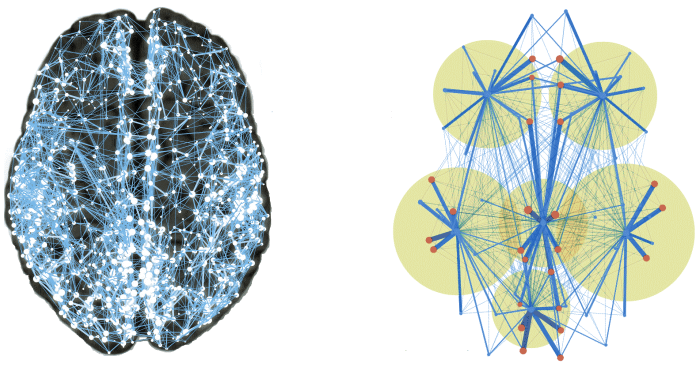


Abbildung 25.7: Die neuronalen Verbindungen eines Gehirns bilden ein skalenfreies Kleine-Welt-Netz. Quelle:[20]

Großteil der neuronalen Signale läuft. Solche Kleine-Welt-Netze ermöglichen eine schnelle

[17] `https://research.facebook.com/blog/three-and-a-half-degrees-of-separation/`
[19] Kang et al. (2011).

und effiziente Informationsverarbeitung bei verhältnismäßig geringem Aufwand für Aufbau und Betrieb der Neuronenstruktur. □

Mathematisch nachgewiesen jedoch ist das Kleine-Welt-Phänomen bislang nur für Zufallsgraphen[21], siehe Abbildung 25.2 links. Im allgemeinen ist es schwierig, für ein gegebenes Netzwerk dessen „Kleine-Welt-Grad" zu bestimmen, also ein Maß für die Kleine-Welt-Eigenschaft zu definieren. Üblicherweise werden dafür zwei messbare globale Eigenschaften eines Netzwerkes verwendet, die *mittlere Weglänge L* und der Grad *C* der Clusterung[22].

Bezeichnet d_{ij} die Distanz zwischen den Knoten i und j eines gegebenen Netzwerkes mit n Knoten, so ist die *mittlere Weglänge L* des Netzes definiert als die durchschnittliche Distanz über alle möglichen Verbindungen:

$$L = \frac{1}{n\,(n-1)} \sum_{i \neq j} d_{ij}. \tag{25.6}$$

(Denn es gibt $\binom{n}{2} = \frac{1}{2}\, n(n-1)$ Paare = Verbindungen in dem Netz, wobei jede Verbindung zwei Richtungen hat.) Der *Clusterkoeffizient C* eines Netzwerk ist definiert als der Quotient

$$C = \frac{3 \cdot \text{Anzahl Dreiecke}}{3 \cdot \text{Anzahl Dreiecke} + \text{Anzahl offener Tripletts}} = \frac{3N_{\triangle}}{3N_{\triangle} + N_{\wedge}}, \tag{25.7}$$

wobei ein Triplett drei verbundene Knoten sind und ein ein Zyklus (Dreieck) aus drei Knoten ist[23],[24]. In ungerichteten Graphen kann bei der Zählung der Dreiecke und Wege die Reihenfolge der Knoten vernachlässigt werden. Allgemein gelten für den Wert des Clusterkoeffizienten die Ungleichungen $0 \leqq C \leqq 1$. Für ein Netzwerk, das sich als eine lineare Kette oder geschlossener Kreis von Knoten darstellen lässt (jeder kennt höchstens zwei andere), gilt $C = 0$, und für eines, in dem jeder jeden kennt (also einer Clique oder einem vollständigen Graph), gilt $C = 1$. Zwar ist klar, dass je größer der Clusterkoeffizient eines Netzes, desto stärker das Kleine-Welt-Phänomen: Im ungünstigsten Fall ist das Netz eine nichtgeschlossene Kette von n Knoten, dann ist $C = 0$ und die maximale Distanz ist $d = n - 1$. In einer Clique mit n Knoten dagegen sind alle Tripletts geschlossen. d.h. $C = 1$ und die maximale Distanz $d = 1$. Allerdings zeigen die beiden Netze in Abbildung 25.4, dass ein höherer Clusterkoeffizient nicht notwendig eine niedrigere Maximaldistanz impliziert.

Definition 25.5. [25] Ein Netzwerk mit n Knoten und m Kanten ist ein *Kleine-Welt-Netz*, wenn für seinen *Kleine-Welt-Koeffizient* die Ungleichung $S > 1$ gilt. Dieser Koeffizient ist durch die mittlere Weglänge L und den Clusterkoeffizient C des Netzes über die Gleichung

$$S = \frac{C}{C^{\mathrm{ER}}_{n,m}} \cdot \frac{L^{\mathrm{ER}}_{n,m}}{L} \tag{25.8}$$

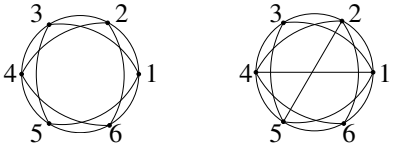
definiert, mit den entsprechenden charakteristischen Koeffizienten

$$L^{\mathrm{ER}}_{n,m} = \frac{\ln(2m + 1 - n)}{\ln(2m/n)} - 1 \qquad \text{und} \qquad C^{\mathrm{ER}}_{n,m} = \frac{2m}{n^2} \tag{25.9}$$

[21] Newman, Barabási et al. (2006):S. 286.
[22] Varshney et al. (2011).
[23] Newman (2010):(7.41).
[24] Newman, Barabási et al. (2006):S. 287.
[25] Humphries und Gurney (2008).

eines Zufallsnetzes („Erdős-Rényi-Graph“, siehe Seite 116) mit n Knoten und m Kanten (Abbildung 25.2 links).[26] □

Beispiel 25.6. Das Prüfen eines Netzwerks auf das Kleine-Welt-Phänomen ist sehr mühsam, wie das folgende Beispiel zeigt. Gegeben seien die beiden Netzwerke:



Die maximale Distanz in beiden Netzen ist $d = 2$, beispielsweise ist $d_{3,6} = 2$ zwischen Knoten 3 und 6. Zu Ermittlung der mittleren Weglängen erstellen wir zunächst die Distanzmatrizen d_{ij} und d'_{ij} der Distanzen zwischen Knoten i und j:

$$d_{ij} = \begin{pmatrix} 0 & 1 & 1 & 2 & 1 & 1 \\ 1 & 0 & 1 & 1 & 2 & 1 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 2 & 1 & 1 & 0 & 1 & 1 \\ 1 & 2 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 \end{pmatrix}, \quad d'_{ij} = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 2 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 2 & 1 & 1 & 0 \end{pmatrix}. \tag{25.10}$$

Damit sind die mittleren Weglängen $L = \frac{12\cdot 1+3\cdot 2}{15} = \frac{6}{5} = 1{,}2$ und $L' = \frac{14\cdot 1+1\cdot 2}{15} = \frac{16}{15} = 1{,}0\overline{6}$. Bezeichnen wir mit (x, y, z) = ein offenes, also nichtgeschlossenes Triplett der Knoten x, y, z gegen Uhrzeigersinn und mit $\overline{(x, y, z)}$ = ein Dreieck. Für das linke Netzwerk erhalten wir dann: $\overline{(1, 2, 3)}$, $(1, 2, 4)$, $(1, 3, 4)$, $\overline{(1, 3, 5)}$, $\overline{(2, 3, 4)}$, $(2, 3, 5)$, $(2, 4, 5)$, $\overline{(2, 4, 6)}$, $\overline{(3, 4, 5)}$, $(3, 4, 6)$, $(3, 5, 6)$, $\overline{(4, 5, 6)}$, $(4, 5, 1)$, $(4, 6, 1)$, $\overline{(5, 6, 1)}$, $(5, 6, 2)$, $(5, 1, 2)$, $\overline{(6, 1, 2)}$, $(6, 1, 3)$, $(6, 2, 3)$. Zählen wir ohne Berücksichtigung der Reihenfolge der Knoten, so erhalten wir den Clusterkoeffizienten

$$C = \frac{3 \cdot 8}{3 \cdot 8 + 12} = \frac{2}{3}. \tag{25.11}$$

Im Netzwerk rechts sind von den 12 offenen Tripletts des linken Netzes nun geschlossen: $\overline{(1, 2, 4)}$, $\overline{(1, 3, 4)}$, $\overline{(1, 2, 5)}$, $\overline{(1, 4, 5)}$, $\overline{(1, 4, 6)}$, $\overline{(2, 3, 5)}$, $\overline{(2, 4, 5)}$, $\overline{(2, 5, 6)}$, die einzigen verbliebenen offenen Tripletts: $(3, 4, 6)$, $(3, 5, 6)$, $(6, 1, 3)$, $(6, 2, 3)$. Damit gilt für den Clusterkoeffizienten C' des rechten Netzes

$$C' = \frac{3 \cdot 16}{3 \cdot 16 + 4} = \frac{12}{13} = 0{,}923. \tag{25.12}$$

Da das linke Netzwerk $n = 6$ Knoten und $m = 12$ Kanten hat und das rechte $n = 6$ und $m = 14$, gilt für die entsprechenden Zufallsgraphen mit (n, m) gemäß (25.8)

$$L^{\mathrm{ER}}_{6;12} = 1{,}124, \quad C^{\mathrm{ER}}_{6;12} = \frac{2}{3}, \quad L'^{\mathrm{ER}}_{6;14} = 1{,}035, \quad C'^{\mathrm{ER}}_{6;14} = 0{,}\overline{7}. \tag{25.13}$$

Für die Kleine-Welt-Koeffizienten der beiden Netzwerke folgt damit schließlich

$$\boxed{S = \tfrac{2/3}{2/3} \cdot \tfrac{1{,}124}{1{,}2} \approx 0{,}94 < 1} \quad \text{und} \quad \boxed{S' = \tfrac{12/13}{0{,}77} \cdot \tfrac{1{,}035}{1{,}066} \approx 1{,}15 > 1.} \tag{25.14}$$

Das linke Netz ist also kein Kleine-Welt-Netz, das rechte dagegen ist eines. Natürlich ist diese Aussage für so kleine Netzwerke mit $n = 6$ Teilnehmern nicht sehr tiefsinnig, aber es sollen ja auch nur die durchzuführenden Berechnungen an diesem einfachen Beispiel klargemacht werden. □

[26] Nach (Newman, Strogatz et al. (2001):Eqs. (5), (11), (12), (50) und (53)) gilt für ein Zufallsnetz mit binomialverteilter Kontaktanzahl $z = 2m/n$ je Knoten wegen $z_1 = z$ und $z_2 = z^2$ die Gleichung für $L^{\mathrm{ER}}_{n,m}$ in (25.9), während nach (Watts und Strogatz (1998)), (Newman (2010):Eq. (6.23)) und (Newman, Barabási et al. (2006):S. 288) mit $z = 2m/n$ die Gleichung für den Clusterkoeffizient $C^{\mathrm{ER}}_{n,m}$ folgt.

Beispiel 25.7. Im linken Netzwerk in Abbildung 25.4 ist jeder Knoten i „Startpunkt“ (im Uhrzeigersinn) von insgesamt 4 Tripletts,[27] wovon nur eines geschlossen ist ($(i, i{+}1, i{+}3)$); da das für jeden Knoten gilt und dabei kein Triplett doppelt gezählt wird, gilt für den Clusterkoeffizient $C = \frac{3n}{4n} = \frac{3}{4}$. Das rechte Netzwerk in Abbildung 25.4 dagegen hat zusätzlich vier Kanten, die jeweils vier weitere Tripletts beinhalten: $C' = \frac{3n}{4n+4\cdot 4} = \frac{3n}{4(n+1)} < \frac{3}{4}$, mit $n = 12$, also $C' = \frac{9}{13}$. Beide Netzwerke in Abbildung 25.4 haben eine maximale Knotendistanz von $d = 3$ ($= \lfloor \log_2 n \rfloor$, mit $n = 12$), obwohl $C' < C$. Zur Berechnung der mittleren Weglänge im rechten Netzwerk von Abbildung 25.4 bestimmt man zunächst die durchschnittliche Distanz $\bar{d}_i$, die der Knoten i mit den anderen Knoten hat,

$$\bar{d}_i = \frac{4 \cdot 1 + 4 \cdot 2 + 3 \cdot 3}{11} = \frac{21}{11},$$

denn die kürzesten Distanzen nach $i + 1$, $i + 2$, $i + 10$ und $i + 11$ sind jeweils 1, die nach $i + 3$, $i + 4$, $i + 8$ und $i + 9$ jeweils 2, und die nach $i + 5$, $i + 6$ und $i + 7$ jeweils 3. Damit folgt

$$L = \frac{12 \cdot 21}{12 \cdot 11} = \frac{21}{11} = 1{,}\overline{90}. \tag{25.15}$$

Für das rechte Netzwerk in Abbildung 25.4 werden durch die vier zusätzlichen Kanten drei Wege der Länge 3 und ein Weg der Länge 2 zu jeweils einer Direktverbindung verkürzt, also dagegen

$$\bar{d}'_i = \begin{cases} \frac{5\cdot 1+4\cdot 2+2\cdot 3}{11} = \frac{19}{11} & \text{für } i = 0, 1, 3, 6, 10, \\ \frac{5\cdot 1+3\cdot 2+3\cdot 3}{11} = \frac{20}{11} & \text{für } i = 4, \\ \frac{6\cdot 1+3\cdot 2+2\cdot 3}{11} = \frac{18}{11} & \text{für } i = 8, \\ \frac{4\cdot 1+4\cdot 2+3\cdot 3}{11} = \frac{21}{11} & \text{für } i = 2, 5, 7, 9, 11, \end{cases}$$

also

$$L' = \frac{5 \cdot 19 + 20 + 18 + 5 \cdot 21}{11 \cdot 12} = \frac{119}{66} = 1{,}803.$$

Da mit $n = 12$ und $m = 24$ für die Zufallsnetzgrößen $L^{\mathrm{ER}}_{12,24} = \frac{\ln 37}{\ln 4} - 1 = 2{,}60$ und $C^{\mathrm{ER}}_{12,24} = \frac{1}{3}$ folgt, ist $S = \frac{3\cdot 3}{4} \cdot \frac{2{,}60}{1{,}\overline{90}} = 3{,}06$ und $S' = \frac{9\cdot 3}{13} \cdot \frac{2{,}60}{1{,}803} = 2{,}995$. □

Beispiel 25.8. Das Kleine-Welt-Phänomen ist durch die Neurowissenschaften für Gehirne von Säugetieren und Vögeln nachgewiesen. So hat selbst eine Taube nach unserer Formel (25.8) einen Kleine-Welt-Koeffizienten

$$S_{\text{Taube}} = 1{,}0356. \tag{25.16}$$

vgl. Shanahan et al. (2013). Eine Taube hat etwa 80 Millionen Gehirnzellen. Demgegenüber hat ein Rabe etwa 1,2 Milliarden Gehirnzellen und zeigt bereits sehr hohe kognitive Fähigkeiten, wie Onur Güntürkün von der Ruhr-Universität Bochum mit seinem Team nachweisen konnte[28], auch wenn er deutlich weniger Gehirnzellen als ein Mensch mit 16 Milliarden. Güntürkün begründet dies mit der zu Säugetieren unterschiedlichen Hirnarchitektur von Vögeln, die für dieselbe kognitive Leistung weniger Zellen und weniger Hirnmasse benötigt. □

Das Kleine-Welt-Phänomen wird von einigen Methoden des Online-Marketings ausgenutzt, insbesondere dem viralen Marketing[29].

[27] $(i, i + 1, i + 2)$, $(i, i + 1, i + 3)$, $(i, i + 2, i + 3)$, $(i, i + 2, i + 4)$, wobei die Addition hier modula $n = 12$ zu verstehen ist.

[28] Güntürkün (2021).

[29] Fischer (2009):§2.

25.3 Wachstum von Netzen

Eines der Modelle, die das Wachstum eines Netzes beschreiben, ist das *Bianconi-Barabási-Modell* von 2001[30],[31]. Es hängt ab von der vorgegebenen Anzahl m, mit der ein jeweils neuer Knoten sich beim Einfügen vernetzt, und der Wahrscheinlichkeitsdichte $\rho(\eta)$ dafür, in dem Netz einen Knoten mit der Fitness $\eta \in [0,1]$ zu finden. Hierbei ist „Fitness“ ein Maß für die Attraktivität eines Knotens. Nach dem Modell ergibt sich aus diesem Parameter dann eine bestimmte Funktion $f(\eta) \in [0,1)$ mit Definitionsbereich $[0,1]$, die wiederum die Anzahl $k_i(\eta_i, t, t_i)$ der Kontakte von Knoten i mit Fitness η_i und Eintrittszeitpunkt t_i zur Zeit $t \geqq t_i$ durch

$$k_i(\eta_i, t, t_i) = m \left(\frac{t}{t_i}\right)^{f(\eta_i)}, \tag{25.17}$$

bestimmt, und ebenso die Konnektivitätswahrscheinlichkeit $P(k)$, d.h. die Wahrscheinlichkeit, dass (zu einem gegebenen Zeitpunkt t) ein Knoten des Netzes k Kontakte hat, durch

$$P(k) \propto k^{-\gamma} \cdot O(\tfrac{1}{\log k}) \qquad \text{mit} \quad \gamma = 1 + \frac{1}{f(1)}. \tag{25.18}$$

Beispielsweise ergibt sich für den speziellen Fall, dass alle Knoten dieselbe Fitness $\eta = 1$ haben,

$$f(\eta) = 1/2, \qquad P(k) \propto k^{-3} \tag{25.19}$$

das *Barabási-Albert-Modell* von 1999[32], siehe Abb. 25.3. Für die Gleichverteilung $\rho(\eta) = 1$ dagegen erhalten wir

$$f(\eta) = \frac{\eta}{1.255}, \qquad P(k) \propto \frac{1}{k^{2.255} \log k}. \tag{25.20}$$

Hier ist $x_* \approx -\frac{1}{1.255}$ die Näherungslösung der transzendentalen Gleichung $\mathrm{e}^{2x} = 1 + x$.

Beispiel 25.9. [33],[34] Für $\rho(\eta) = (1-\eta)^\lambda$ mit dem Parameter $\lambda \in [0, \infty)$ sagt das Bianconi-Barabási-Modell zwei verschiedene, von λ abhängende Phasen voraus.

(i) $\lambda < 1$: Die Phase „Je fitter, desto reicher“ (Fit-get-rich). Da $\rho'(\eta) = -\lambda(1-\eta)^{\lambda-1} < 0$ und $\rho''(\eta) = \lambda(\lambda-1)(1-\eta)^{\lambda-2} < 0$, ist ρ streng monoton fallend und konkav. Insbesondere

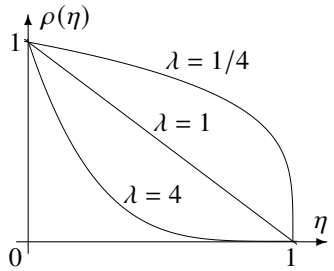


Abbildung 25.8: Graphen $\rho(\eta) = (1-\eta)^\lambda$ für $\lambda = \frac{1}{4}, 1, 4$.

ist $\rho'(\eta) \to -\infty$ für $\eta \to 1$ (Abb. 25.8), d.h. es gibt in dem Netzwerk sehr viele Knoten, die die höchsten Fitnesswerte haben. Die Funktion $f(\eta)$ ist dann streng monoton steigend, d.h. mit (25.17) wächst die Konnektivität der fitteren Knoten schneller als die der weniger fitten Knoten. In dem Netzwerk bilden sich mehrere Hauptknotenpunkte *(hubs)*.

[30] Bianconi und Barabási (2001b).
[31] Newman, Barabási et al. (2006):S. 361ff.
[32] Newman, Barabási et al. (2006):S. 349ff.
[33] Bianconi und Barabási (2001a).
[34] Albert und Barabási (2002):pp 41.

(ii) $\lambda > 1$*: Bose-Einstein-Kondensat.* In diesem Fall ist $\rho(\eta)$ monoton fallend und konvex, d.h. es gibt nur einen (bzw. nur sehr wenige) Knoten mit maximaler Fitness („*Superhubs*“[35]). In diesem Falle brechen zudem die mathematischen Voraussetzungen zur Gültigkeit der Gleichungen (25.17) zusammen. Der Knoten mit maximaler Fitness entwickelt sich zum einzigen Hauptknotenpunkt, da *jeder* neu hinzukommende Knoten sich mit ihm verbindet *(Winner takes all)*. □

25.4 Netzwerkeffekte

Je größer ein soziales Netzwerk, desto größer seine Potenziale für Synergie, Kreativität und Innovation. Die Menschheit brachte in ihrer Entwicklung immer größere soziale Strukturen hervor. Wanderte der Homo sapiens vor 100 000 Jahren noch in kleinen Herden in den Savannen Ostafrikas, entstanden mit der Sesshaftwerdung („Neolithische Revolution“) vor etwa 10 000 Jahren erste Dörfer, um 2500 v. Chr. die ersten Hochkulturen und Staaten sowie die Schrift. Die Entwicklung setzte sich bis heute fort zur Bildung von Megastädten (seit 2008 lebt die Mehrheit der Menschheit in Städten[36]) und durch das Internet zu weltweit vernetzten sozialen Gruppen.

Die Netzwerkökonomie beschäftigt sich mit den Auswirkungen, die speziell auf die soziale Vernetzung der wirtschaftlichen Akteure zurückzuführen sind. Ein wesentliches Phänomen bilden dabei Netzwerkeffekte. Grob gesagt handelt es sich dabei um sich verstärkende oder auch abschwächende Rückkopplungen, bei denen die Handlungsentscheidung der einzelnen Netzteilnehmer durch das tatsächliche oder erwartete Verhalten der anderen Netzteilnehmer beeinflusst wird. In den Wirtschaftswissenschaften werden Netzwerkeffekte zu den Externalitäten gezählt. Wir werden uns in diesem Abschnitt vor allem mit den quantitativ erfassbaren Aspekten von Netzwerkeffekten beschäftigen.

Ein eher schwer quantitativ bewertbarer Netzwerkeffekt ist in der Arbeit von Teams das Phänomen „das Ganze ist mehr als die Summe seiner Teile“, also der Entstehung qualitativ neuer Potenziale und Fähigkeiten durch soziale Vernetzung. Dadurch kann eine Gruppe eine Innovationsfähigkeit erzielen, die jeder Einzelne für sich allein nicht erreicht würde. Einer der Ersten, die diese Eigenschaft von Teams systematisch ausnutzte, war Thomas Alva Edison, der bereits 1875 eine Entwicklungsabteilung zur industriellen Forschung gründete und so zu neuen Erfindungen wie die Glühbirne, die Tonaufnahme und die Filmkamera kam[37].

25.4.1 Nutzenfunktionen von Netzwerken

Zur Quantifizierung des durch dieser Effekte bewirkten ökonomischen Nutzen gibt es im wesentlichen zwei Ansätze, die Metcalfe'sche Nutzenfunktion und die Reed'sche Nutzenfunktion.

Der US-amerikanische Elektroingenieur Robert M. Metcalfe definierte den Nutzen *(utility)* u_{M} eines sozialen Netzwerks als die Anzahl der möglichen Verbindungen, die sich in dem Netzwerk bilden können.

Satz 25.10 (Metcalfe'sches Gesetz). *Bei einem Netzwerk mit n Teilnehmern ist gemäß Metcalfe der Nutzen* $u_{\mathrm{M}}(n)$ *gegeben durch*

$$u_{\mathrm{M}}(n) = \frac{n^2 - n}{2}. \tag{25.21}$$

Beweis. Die größtmögliche Anzahl an Verbindungen von n Knoten in einem Graphen beträgt $\binom{n}{2} = n(n-1)/2$. □

[35] Newman (2010):§14.4.4.
[36] Hayden (2008):S. 17.
[37] Gaede (2008):S. 15.

Dagegen definierte der US-amerikanische Informatiker David P. Reed den Nutzen u_R eines sozialen Netzwerks als die Anzahl der möglichen Cliquen mit mindestens zwei Teilnehmern, die sich in dem Netzwerk bilden können.

Satz 25.11 (**Reed'sches Gesetz**). *Bei einem Netzwerk mit n Teilnehmern ist gemäß Reed der Nutzen $u_R(n)$ gegeben durch*

$$u_R(n) = 2^n - n - 1. \tag{25.22}$$

Beweis. Eine Clique von Netzteilnehmern ist nach obiger Definition eine m-elementige Teilmenge der Netzteilnehmer, die mindestens zwei Teilnehmer hat, also $m \geqq 2$ erfüllt. Die Anzahl aller möglichen Teilmengen einer n-elementigen Menge ist nun 2^n: das wird sofort klar, wenn man die Teilnehmer durchnummeriert und eine Teilmenge (eineindeutig!) durch einen Binärstring („BitSet") der Länge n betrachtet, dessen k-te Stelle mit 0 oder 1 anzeigt, ob Netzteilnehmer Nummer k zu der Teilmenge gehört oder nicht; die Anzahl aller möglichen Binärstrings mit n Stellen (n Bits!) ist aber genau 2^n. Diese Gesamtheit aller Teilmengen umfasst allerdings auch die leere Menge und die n einelementigen Teilmengen, die man also von 2^n abziehen muss. □

Welche der beiden Nutzenfunktionen beschreibt den ökonomischen Nutzen eines sozialen Netzwerk besser? Die Frage ist nicht eindeutig zu beantworten, da „der" ökonomische Nutzen dazu erst streng definiert werden müsste. Aus Sicht des Marketings oder Vertriebs eines Unternehmens ist sicher die Anzahl der möglichen Cliquen eine wichtige Größe, d.h. deren Nutzen beschreibt eher die Reed'sche Nutzenfunktion. Allerdings impliziert sie, dass für ein sehr großes n jeder weitere Netzteilnehmer den Nutzen jeweils nahezu verdoppelt.

Beispiel 25.12. Um die Nutzenfunktion eines sozialen Netzwerks zu überprüfen, kann der Wert von Facebook abhängig von der Anzahl seiner Nutzer betrachtet werden. Als Maßzahl soll dabei die im jährlichen Geschäftsbericht von Facebook angegebene Anzahl MAU der monatlich aktiven User verwendet werden, als Wert des Netzwerks das ebenfalls darin dokumentierte Anlagevermögen *(total assets)*. Für die Geschäftsjahre 2008 bis 2020 sind die Werte in Tabelle 25.1 aufgelistet. Mit Hilfe einer Regressionsanalyse kann man nun die beiden Nutzenfunktionen

Geschäftsjahr	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
MAU	100	300	608	845	1056	1228	1393	1591	1860	2129	2320	2498	2797
Anlagevermögen	0,505	1,009	2,990	6,331	15,103	17,895	39,966	49,407	64,961	84,524	97,334	133,376	159,316

Tabelle 25.1: Anzahl monatlicher User (MAU) in Mio und des jeweiligen Anlagevermögens in Milliarden US$. Quellen: `https://investor.fb.com/financials`, Annual Reports 2012 (S. 41/45), 2015 (S. 30/34), 2020 (S. 50/55)

u_M und u_R an die Berichtsdaten anpassen („fitten"), vgl. Abbildung 25.9[38]. Die Metcalfe'sche Nutzenfunktion modelliert mit einem Bestimmtheitsmaß $R^2_M = 98{,}48\%$ den Zusammenhang von MAU und Wert des Netzwerks besser als der Reed'sche Ansatz mit dem Bestimmtheitsmaß $R^2_R = 92{,}52\%$. Auch optisch scheint die Metcalfe'sche Nutzenfunktion den Zusammenhang besser zu modellieren als die Reed'sche. Da schließlich das Metcalfe'sche Modell mit nur einem Parameter sogar noch einfacher als das Reed'sche mit zwei, ist sie zu bevorzugen. □

Beide Nutzenfunktionen – so wie im Übrigen auch andere in der Fachliteratur[39] diskutierten Nutzenfunktionen für Netzwerke – widersprechen damit dem ökonomischen Gesetz vom abnehmenden Grenznutzen, dem „1. Gossen'schen Gesetz"[40]. Es besagt, dass der durch eine

Gesetz vom abnehmenden Grenznutzen

[38] de Vries (2020):Aufgabe 1.6.1.

[39] Clement und Schreiber (2016):Abb. 3.10.

[40] Bofinger (2007):§6.3.

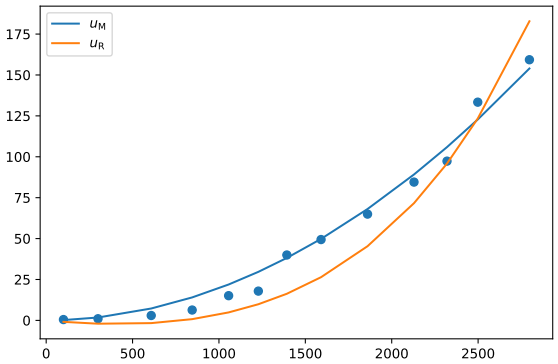


Abbildung 25.9: Plot der Nutzenfunktionen u_M und u_R von Facebook und des Anlagevermögens in Milliarden US$ gegen die Anzahl MAU in Millionen.

gegebene Zusatzmenge eines Gutes oder einer Dienstleistung bewirkte Zusatznutzen, also der Grenznutzen $\frac{\Delta u}{\Delta n}$ für die zusätzliche Menge Δn, mit zunehmender Menge immer kleiner wird, wie in Abbildung 25.10 schematisch dargestellt. Solche Nutzenfunktionen werden auch als „degressiv steigend“ bezeichnet[41]. Mathematisch bedeutet das, dass eine solche Nutzenfunktion

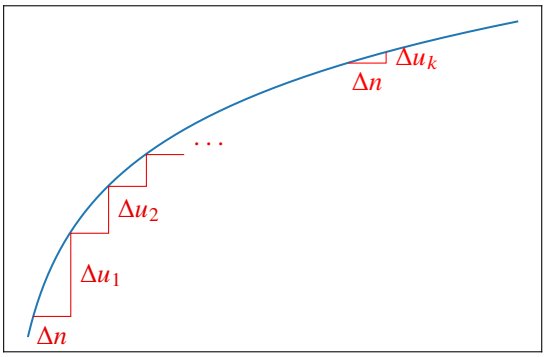


Abbildung 25.10: Abnehmender Grenznutzen: Die Nutzenfunktion $u(n)$ ist konkav, d.h. $\frac{\Delta u_1}{\Delta n} > \frac{\Delta u_2}{\Delta n} > \ldots > \frac{\Delta u_k}{\Delta n}$

$u(n)$ muss konkav sein!

$u(n)$ als Funktion der Menge n konkav ist.[42] Ist u zweimal differenzierbar, so ist u genau dann konkav, wenn $u''(n) < 0$. Sowohl die Metcalfe'sche als auch die Reed'sche Nutzenfunktion sind jedoch konvex, denn es gilt $u''_M(n) = 1 > 0$ und $u''_R(n) = (\ln 2)^2\, 2^n > 0$. Sie sind entsprechend „progressiv steigende“ Nutzenfunktionen: Der n-te Netzteilnehmer stiftet einen höheren Nutzen als der $(n-1)$-te. Beide Nutzenfunktionen messen den Nutzen des Netzwerk in Abhängigkeit von seiner Größe, und hier nimmt der Grenznutzen bei zusätzlicher Netzvergrößerung eben nicht ab, sondern im Gegenteil zu, siehe Abbildung 25.11. Diese Eigenschaft des *zunehmenden* Grenznutzens ist die wesentliche Ursache der meisten Netzwerkeffekte.

Zunehmender Grenznutzen: eine Ursache von Netzwerkeffekten

[41] Buxmann et al. (2015):§2.2.6.

[42] Im Allgemeinen heißt eine Funktion $f : (a, b) \to \mathbb{R}$ (mit $a < b$) *konkav*, wenn

$$f\big(tx + (1-t)\,y\big) > t\,f(x) + (1-t)\,f(y) \tag{25.23}$$

für alle $t \in [0, 1]$ und alle $x, y \in (a, b)$ gilt. Das ist äquivalent mit der Bedingung

$$\frac{f(y) - f(x)}{y - x} > \frac{f(z) - f(y)}{z - y} \qquad \text{für } x < y < z \tag{25.24}$$

vgl. (Bröcker (1995):S. 144). Speziell für $y = x + \Delta x$ und $z = x + 2\Delta x$ folgt dann $\frac{\Delta f(x)}{\Delta x} > \frac{\Delta f(y)}{\Delta x}$ für ein kleines $\Delta x > 0$. Entsprechend heißt f *konvex*, wenn überall „>“ durch „<“ ersetzt wird.

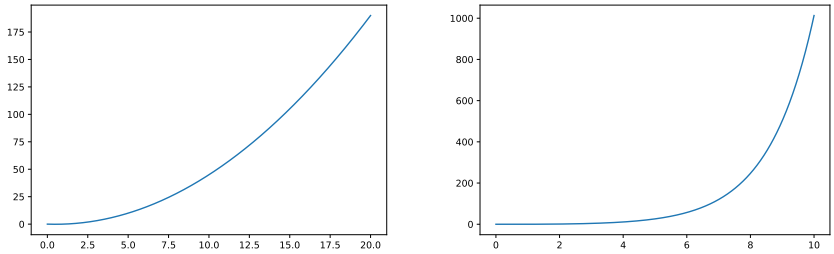


Abbildung 25.11: Die konvexen Nutzenfunktion $u_{\mathrm{M}}(n)$ und $u_{\mathrm{R}}(n)$

Satz 25.13. *Der Nutzwert zweier getrennter Netzwerke ist stets kleiner als der Nutzwert des gemeinsam gebildeten Netzwerks.*

Beweis. Seien n_1 und $n_2 \geqq 1$ die Anzahl der Teilnehmer von Netzwerk 1 bzw. 2. Dann gilt für die Metcalfe'sche Nutzenfunktion

$$\begin{aligned} u_{\mathrm{M}}(n_1) + u_{\mathrm{M}}(n_2) &= \frac{n_1^2 + n_2^2}{2} - \frac{n_1 + n_2}{2} \\ &< \frac{n_1^2 + 2n_1n_2 + n_2^2}{2} - \frac{n_1 + n_2}{2} = \frac{(n_1 + n_2)^2}{2} - \frac{n_1 + n_2}{2} \\ &= u_{\mathrm{M}}(n_1 + n_2), \end{aligned} \tag{25.25}$$

und entsprechend für die Reed'sche Nutzenfunktion

$$\begin{aligned} u_{\mathrm{R}}(n_1) + u_{\mathrm{R}}(n_2) &= 2^{n_1} + 2^{n_2} - (n_1 + n_2) - 2 \\ &< 2^{n_1} \cdot 2^{n_2} - (n_1 + n_2) - 1 = 2^{n_1+n_2} - (n_1 + n_2) - 1 \\ &= u_{\mathrm{R}}(n_1 + n_2). \end{aligned} \tag{25.26}$$

Die Differenzen sind für große n_1 und n_2 beträchtlich, für den Metcalfe'schen Nutzwert gilt $u_{\mathrm{M}}(n_1 + n_2) = u_{\mathrm{M}}(n_1) + u_{\mathrm{M}}(n_2) + n_1n_2$, für den Reed'schen Nutzwert sogar $u_{\mathrm{R}}(n_1 + n_2) = u_{\mathrm{R}}(n_1) + u_{\mathrm{R}}(n_2) + 2^{n_1+n_2} - (2^{n_1} + 2^{n_2}) + 1$. □

Beispiel 25.14. Das Netzwerk des Apple-Stores mit n_1 Teilnehmern und das Amazon-Netzwerk mit n_2 Mitgliedern hätten einen weit größeren Wert, wenn sie fusionieren würden.[43] □

Für eine vollständige wirtschaftliche Betrachtung müssen dem Nutzen die Kosten einer Netzwerkvergrößerung gegenüber gestellt werden. Weder die Metcalfe'sche noch die Reed'sche Nutzenfunktion berücksichtigen die Kosten der Vergrößerung des Netzwerks. Mit einer Kostenfunktion, die bei steigender Netzwerkgröße stärker steigt als die anzusetzende Nutzenfunktion, wird der Nutzengewinn am Ende vernichtet und es kommt zu keinen Netzwerkeffekten. Bei digitalen Gütern wie Software, Social Media oder Browserspielen jedoch, die über das Internet verbreitet werden können, sind die Grenzkosten praktisch gleich null, denn sowohl die Vervielfältigung des Guts als auch die Infrastruktur des Netzwerks sind fast gratis. Bei Konsolenspielen dagegen ist die Betrachtung nicht so einfach, denn die Infrastruktur, also die Verteilung der Konsolen, erzeugt zusätzliche Kosten, um nur ein Beispiel zu nennen[44].

25.4.2 Externalitäten

In der Volkswirtschaftslehre ist eine *Externalität* oder ein *externer Effekt* ein Verlust oder Ertrag einer ökonomischen Handlung, die daran Unbeteiligte erfahren:[45][46] Fliegt man zum Beispiel

[43] Lanier (2014):S. 436.
[44] Buxmann et al. (2015):§2.2.5.
[45] Bofinger (2007):§14.3.
[46] Clement und Schreiber (2016):§3.2.1.

Externalität = soziale vs. private Kosten / Erträge

von Frankfurt nach New York, so bezahlt man lediglich diejenigen Kosten, die die Fluggesellschaft decken muss, nicht aber diejenigen, die dadurch für die Umwelt wegen Lärm oder Luftverschmutzung entstehen. Allgemein definiert man die *privaten Kosten* einer wirtschaftlichen Handlung als die Kosten, die das Unternehmen übernimmt, die *sozialen Kosten* dagegen sind die gesamten anfallenden Kosten der wirtschaftlichen Handlung. Zu *negativen Externalitäten* (NE) kommt es dabei, wenn die sozialen Kosten höher sind als die privaten, und zu *positiven Externalitäten* (PE), wenn die privaten Erträge geringer sind als die sozialen Erträge,

$$\begin{aligned} \text{NE} &= \text{soziale Kosten} - \text{private Kosten} \\ \text{PE} &= \text{soziale Erträge} - \text{private Erträge}. \end{aligned} \tag{25.27}$$

(Beachte, dass stets NE, PE $\geqq 0$, da nach Definition die sozialen Kosten und Erträge die privaten jeweils mit umfassen.) Ein Beispiel für eine positive Externalität ist die innerbetriebliche Ausbildung, durch die der Betrieb zwar produktiver wird und private Erträge erwirtschaftet, aber das Risiko trägt, dass der oder die Ausgebildete früher oder später zu einem anderen Arbeitgeber wechselt: Das auszubildende Unternehmen kann in diesem Fall nur teilweise oder überhaupt nicht von den Ausbildungskosten profitieren, die sozialen Erträge sind höher als die privaten. Auch Forschung und Entwicklung eines Unternehmens ist eine positive Externalität, wenn daraus neue Produkte entstehen, die der Allgemeinheit nutzen.[47]

Positive Externalitäten: Betriebliche Ausbildung und Forschung und Entwicklung

Wichtige Beispiele für negative Externalitäten sind die Verschmutzung der Umwelt und die Erzeugung von CO_2 bei der Bereitstellung von Produkten bzw. Dienstleistungen; in beiden Fällen werden die Folgekosten von der Allgemeinheit oder nachfolgenden Generationen übernommen. Eine weitere Externalität ist das *moralische Risiko (moral hazard)*, das ein leichtsinniges oder verantwortungsloses Handeln bezeichnet, welches eingegangen wird in der Gewissheit, dass im Falle des Gelingens die Gewinne erlangt werden, das Risiko eines Misslingens jedoch großenteils oder sogar ganz von der Allgemeinheit übernommen wird[48]. Beispielsweise ist dies der Fall bei Finanzspekulationen von Banken und Finanzinstituten, für die das Systemrisiko im Falle eines Zusammenbruchs der Staat übernimmt.[49] So wie bei den großen Finanzkrisen 1929 und 2008 geschehen, gemäß dem Motto „Privatisierung der Gewinne, Sozialisierung des Risikos".

Negative Externalitäten: Umweltverschmutzung und moralisches Risiko (*moral hazard*)

Zusammengefasst wird allgemein ein Unternehmen Projekte mit positiven Externalitäten eher *nicht* durchführen, obwohl die Allgemeinheit einen Nutzen daraus ziehen würde. Ein Projekt mit negativen Externalitäten dagegen wird sicher umgesetzt, gerade weil es auf Kosten der Umwelt oder der Allgemeinheit geht. Der Markt berücksichtigt also ohne weitere Rahmenbedingungen nicht die sozialen Kosten und mindert die sozialen Erträge einer Unternehmung, beides bleibt „extern". Externalitäten „stellen damit einen fundamentalen Organisationsdefekt des Marktes dar"[50], also ein Versagen des Marktes; vgl. dazu auch[51] oder[52]. Während im Fall negativer Externalitäten ein übermäßig hoher Verbrauch öffentlicher Güter stattfindet, unterbleibt das Angebot eines öffentlichen Gutes bei positiven Externalitäten. Entsprechend sollte der Staat regulierend in den Markt eingreifen, um Externalitäten zu „internalisieren" und so deren Folgen zu vermeiden. Beispiele dafür sind Schadstoffsteuern in der Umweltpolitik („Pigou-Steuer") oder staatliche Auflagen wie Emissionsgrenzwerte oder CO_2-Zertifikatehandel.[53]

Der Markt selbst schafft keine Anreize zur Vermeidung sozialer Kosten und zur Mehrung sozialer Erträge

[47] Samuelson und Nordhaus (1995):S. 32.
[48] Bofinger (2007):S. 256.
[49] Sinn (2009):S. 96.
[50] Bofinger (2007):S. 273.
[51] P. R. Krugman und Wells (2006):S. 125.
[52] Samuelson und Nordhaus (1995):S. 32.
[53] Bofinger (2007):S. 272ff.

25.4.3 Netzwerkeffekte als Externalitäten

Externalitäten treten auch in sozialen Netzwerken auf. So ist nach Gleichung (25.21) der Nutzen eines sozialen Netzwerks direkt abhängig von der Anzahl der Netzteilnehmer. Da somit der Eintritt eines neuen Mitglieds neben seinem privaten Nutzen einen Nutzen für die Allgemeinheit hat, handelt es sich um eine positive Externalität. Ein Beispiel für eine negative Netzexternalität ist das Braess-Paradoxon, das wir weiter unten betrachten werden.

In einem Markt ohne Externalitäten ergibt sich die Nachfrage nach einem Gut oder einer Dienstleistung in Abhängigkeit von der Nachfragemenge n anhand einer Nachfragekurve $r(n)$, die durch eine monoton fallende Funktion r gegeben ist. (Hierbei steht r für „Reservierungspreis“, also den Preis, für den ein Nachfrager das Angebot kaufen würde.) Dadurch steigt die nachgefragte Menge mit sinkendem Angebotspreis p und umgekehrt[54], siehe Abbildung 25.12 links. Im Zusammenhang mit Netzwerken wird hierbei die Nachfragemenge oft einfach durch die Anzahl der Nachfragenden dargestellt[55]. Näherungsweise verschwinden die variablen Kosten

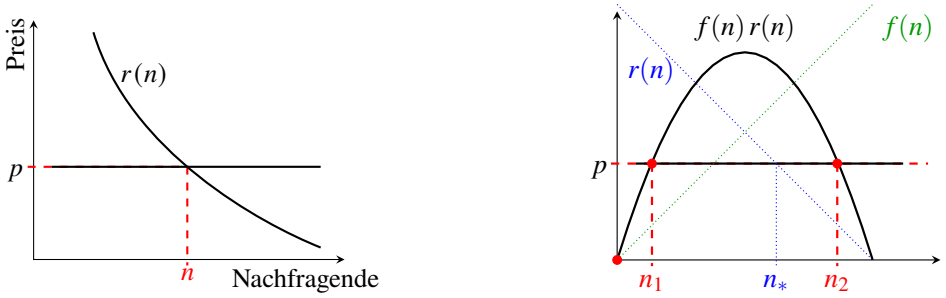


Abbildung 25.12: Links: Nachfragekurve $r(n)$ in Abhängigkeit der Anzahl n der Nachfragenden und bei konstantem Angebotspreis p. Rechts: Nachfragekurve $f(n)\,r(n)$ bei einem Netzwerkeffekt $f(n)$, hier $f(n) = n$ und $r(n) = n_{max} - n$ für ein Netzwerk mit $n_{\max}$ Teilnehmern; in blau ist die Nachfragemenge n_* bei dem Angebotspreis p ohne Netzwerkeffekt dargestellt.

bzw. die Produktionskosten zur Erstellung der Angebotsmenge in elektronischen Netzwerken, denn die Distribution oder das Kopieren des Produkts ist so gut wie umsonst. Die Angebotskurve, die im Allgemeinen monoton steigend ist, ist in einem solchen Markt also konstant. Ist p der Marktpreis und ist n so, dass der Nachfragepreis dem Angebotspreis entspricht, also

$$p = r(n) \tag{25.28}$$

gilt, so kaufen n Netzteilnehmer die Leistung. Ein solches n beschreibt dann ein *Marktgleichgewicht*.

Ein Netzwerkeffekt tritt nun ein, wenn die Nachfrage neben dem Preis zusätzlich durch die Anzahl der Käufer beeinflusst wird. Wird die Attraktivität der Leistung durch eine hohe Kaufrate noch gesteigert, so spricht man von einem *positiven Netzwerkeffekt*, wird sie durch eine hohe Kaufrate dagegen gemindert, so handelt es sich um einen *negativen Netzwerkeffekt*. Ein einfaches Modell für einen Netzwerkeffekt in Märkten ist eine monotone Funktion $f(n)$, die mit dem Reservierungspreis $r(n)$ multipliziert wird, $f(n)\,r(n)$, und so die Nachfragekurve unter Einbeziehung des Netzwerkeffekts darstellt.[56] Zur Modellierung eines positiven Netzwerkeffekts ist f monoton steigend, für die eines negativen Netzwerkeffekts monoton fallend. Liegt ein

[54] Bofinger (2007):§2.

[55] Easley und Kleinberg (2010):§17.

[56] Da der Netzwerkeffektfaktor f im Allgemeinen eine Funktion der *erwarteten* Kaufrate z ist, und nicht der *tatsächlichen* Kaufrate n, müssten wir eigentlich $f(z)\,r(n)$ betrachten, wie es in (Easley und Kleinberg (2010):§17.2) geschieht. Für unsere Zwecke zur Betrachtung positiver Netzwerkeffekte ist jedoch die Gleichsetzung der Größen eine zulässige Vereinfachung.

positiver Netzwerkeffekt vor, so beschreibt bei einem gegebenen Angebotspreis p ein n, das die Gleichung

$$p = f(n)\, r(n) \tag{25.29}$$

erfüllt, ein *Gleichgewicht der selbsterfüllenden Erwartungen (self-fulfilling expactations equilibrium)*[57]. Liegt dagegen ein negativer Netzwerkeffekt vor, so liefert ein Gleichung (25.29) genügendes n ein *Gleichgewicht der selbstnegierenden Erwartungen*[58].

Beispiel 25.15. *(Das soziale Unternehmensnetzwerk)*[59] Das Management eines Unternehmens mit 100 Beschäftigten richtet für die Belegschaft eine Netzwerkseite zur Organisation des Workflows ein. Für die oder den einzelnen Beschäftigten wird dieses Netzwerk attraktiv, wenn mindestens 60 Beschäftigte sich daran beteiligen. Da bei diesem Szenario die Attraktivität bei hoher Nachfrage steigt, handelt es sich um einen positiven Netzwerkeffekt. Nehmen in diesem Fall an, dass alle Beschäftigten dieselben Erwartungen zur Beteiligung an dem Netzwerk haben (*shared expectations*): Nehmen alle an, dass mehr als 60 sich beteiligen, so werden sich mehr als 60 (nämlich 100) daran beteiligen; nehmen jedoch alle an, dass sich weniger als 60 beteiligen, so werden sich weniger als 60 (nämlich 0) daran beteiligen. Es handelt sich also um selbsterfüllende Erwartungen. Positive Netzwerkeffekte werden wir im Folgenden näher betrachten. □

Beispiel 25.16. *(Die El Farol Bar)*[60] In der El Farol Bar in Santa Fe wird jeden Donnerstag abend Livemusik gespielt. Die Bar hat Platz für 60 Leute, es gibt jedoch 100 Interessierte. Erwartet ein Interessent mehr als 60 Gäste, so bleibt er lieber zu Hause, erwartet er weniger, so geht er ins El Farol. Alle Interessierten müssen sich zum selben Zeitpunkt entscheiden, ob sie ausgehen oder nicht. Da bei diesem Szenario die Attraktivität bei hoher Nachfrage *sinkt*, handelt es sich um einen negativen Netzwerkeffekt. Nehmen in diesem Fall alle Beschäftigten an, dass mehr als 60 Gäste kommen, so werden weniger als 60 (nämlich 0) kommen; nehmen jedoch alle an, dass weniger als 60 kommen, so werden mehr als 60 (nämlich 100) kommen. Es handelt sich also um selbstnegierende Erwartungen. Oder wie es der Baseballstar Yogi Berra formulierte: *„Oh that place: It's so crowded that nobody goes there anymore."*[61] (In der Realität wird es jedoch nicht gemeinsame, sondern unterschiedliche Erwartungen geben, so dass im Schnitt eher um die 50 Gäste anwesend sind.) □

Charakteristisch für Märkte mit positiven Netzwerkeffekten ist, dass es mehrere Gleichgewichte gibt, da die Nachfragekurve $f(n)r(n)$ in der Regel nicht mehr monoton fallend ist. In Abbildung 25.12 rechts ist als Beispiel $f(n) = n$ und $r(n) = (n_{\max} - n)$ für ein Netzwerk mit $n_{\max}$ Teilnehmern dargestellt. Der Netzwerkeffektfaktor $f(n)$ bewirkt, dass der Markt komplizierter ist als der entsprechende Markt ohne Netzwerkeffekt. Insbesondere können nun mehrere Marktgleichgewichte existieren, hier eines bei einer sehr kleinen Nachfrage n_1 und eines bei einer sehr hohen Nachfrage n_2. Das erste Gleichgewicht für n_1 ist bei einer weit geringeren Nachfrage als n_* bei einem entsprechenden Markt ohne Netzwerkeffekt, während die Nachfrage n_2 im zweiten Gleichgewicht größer ist.

Betrachtet man die Dynamik eines solchen Marktes mit Netzwerkeffekten, so erkennt man die unterschiedlichen Eigenschaften der drei Marktgleichgewichte $n = 0$, $n = n_1$ und $n = n_2$. Für einen Nachfrager $n < n_1$ ist wegen $f(n)\, r(n) < p$ der Marktpreis p zu hoch, d.h. es existiert *keine* Nachfrage für das Intervall $[0, n_1)$. Ebenso verhält es sich für das Intervall $(n_2, n_{\max}]$. Für das Intervall dazwischen, (n_1, n_2), ist dagegen der Marktpreis niedriger als der Nachfragepreis,

[57] Easley und Kleinberg (2010):S. 454f.
[58] Easley und Kleinberg (2010):S. 472f.
[59] Easley und Kleinberg (2010):§17.7.
[60] Arthur (1994); Easley und Kleinberg (2010):§17.7.
[61] Arthur (1994).

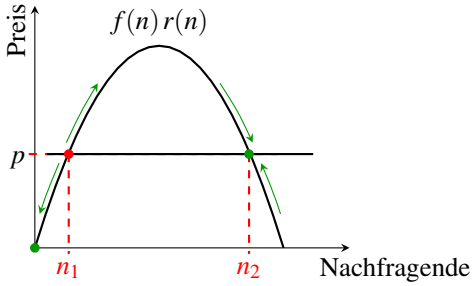


Abbildung 25.13: Dynamik eines Marktes mit Netzwerkeffekt: Das Gleichgewicht für $n = n_1$ ist labil, die Gleichgewichte für $n = 0$ und $n = n_2$ dagegen sind stabil.

d.h. der Markt wird bei der Nachfrage $n = n_2$ ein stabiles Gleichgewicht erlangen[62]. Siehe dazu Abbildung 25.13. Das labile Gleichgewicht $n = n_1$ ist ein *kritischer Punkt* für das Produkt, denn erst wenn es gelingt, die Nachfrage auf $n > n_1$ zu steigern, ist es erfolgreich, im anderen Fall ist es ein Flop.

Bemerkung 25.17. Positive Netzwerkeffekte implizieren die Tendenz zu einem natürlichen Monopol, da der Anbieter mit den meisten Anwendern auch den höchsten Nutzen bietet und sich daher am Markt durchsetzt.[63]. Solche Monopole nennt Jaron Lanier „Sirenenserver“ und zeichnen sich extreme Informationsasymmmetrie aus, da sie über Big Data sehr viel über ihre Nutzer wissen, die jedoch über den Sirenenserver so gut wie nichts[64]. Bislang haben jedoch technologische Veränderungen und neue Trends dazu geführt, dass solche Monopole in vielen Fällen nur von kurzer Dauer waren[65]. □

Das Braess-Paradox

Ein Beispiel für eine negative Externalität in Netzwerken ist das *Braess-Paradox*. Es kommt ursprünglich aus der Verkehrsplanung und beschreibt die paradoxe Situation, dass der Ausbau eines Straßennetzes um eine neue Schnellstraße bei insgesamt gleichem Verkehrsaufkommen zu einer Erhöhung der Fahrdauer für alle Fahrzeuge führt. Aus Sicht eines individuellen Verkehrsteilnehmers kommt es also aufgrund des (ganz rationalen) Verhaltens der anderen zu einem negativen externen Effekt, obwohl das öffentliche Gut „Straßennetz“ ausgebaut wurde. Als idea-

Abbildung 25.14: Braess-Paradox: Kapazitätserhöhung führt zu schlechterer Netzauslastung

lisiertes Beispiel betrachten wir dazu vier Städte A, B, C, D, die wie in Abbildung 25.14 links dargestellt durch vier Straßen verbunden sind. Die Fahrdauer in Minuten auf den Straßen A–C und B–D seien stets 50 Minuten, wobei die Fahrdauer auf den Teilstrecken A–B und C–D von dem Verkehrsaufkommen x gemäß

$$t_{AB}(x) = t_{CD}(x) = 40x, \qquad t_{BD} = t_{AC} = 50 \tag{25.30}$$

[62] Easley und Kleinberg (2010):§17.3.
[63] Dillerup und Stoi (2013):S. 776.
[64] Lanier (2014):Kap. 5.
[65] Dillerup und Stoi (2013):S. 776.

abhängt. Das Verkehrsaufkommen x ist hier eine Zahl von 0 bis 1 und gibt denjenigen Anteil aller Autos wieder, die auf dem betreffenden Teilstück fahren. Die Fahrzeiten auf den Teilstrecken A–C und B–D sind also – unabhängig vom Verkehrsaufkommen – immer gleich, während die Zeiten auf den Teilstrecken A–B und C–D proportional mit dem Verkehrsaufkommen zunehmen. Der Wert $x = 0$ bedeutet dann, dass kein Auto über das Teilstück fährt, für $x = 1$ wird dagegen die maximale Verkehrsdichte und erreicht und die Fahrzeit ist am längsten. Man überlegt sich schnell, dass die für alle Teilnehmer kürzeste Fahrzeit sich genau dann ergibt, wenn die eine Hälfte der Autos den Weg A–B–D, die andere den Weg A–C–D wählt. In diesem Falle beträgt die Fahrdauer für alle

$$t_{AB}(0{,}5) + t_{BD} = 20 + 50 = 70 \text{ min}. \tag{25.31}$$

Diese Lösung stellt das Nash-Gleichgewicht des Systems dar[66]. Auf diese Weise führt ein rationales und nur das eigene Interesse beachtendes Verhalten der einzelnen Akteure zu einer, global gesehen, optimalen Situation.

Erweitert man das Straßennetz um eine mehrspurige Schnellstraße zwischen B und C, die beispielsweise einen Fluss überbrückt und soweit ausgebaut ist, dass unabhängig vom Verkehrsaufkommen jedes Auto nur noch eine Fahrdauer von $t_{BC} = 5$ Minuten benötigt, so ergibt sich folgender optimaler Zustand des Gesamtsystems, also des Nash-Gleichgewichts: Jedes Fahrzeug fährt nun die Strecke A–B–C–D mit einer Fahrzeit von

$$t_{AB}(1) + t_{BC} + t_{CD}(1) = 40 + 5 + 40 = 85 \text{ min}. \tag{25.32}$$

Wir haben also die paradoxe Situation, dass eine Kapazitätserhöhung des Straßennetzes zu einer optimalen Fahrdauer zwischen A und D für alle Autos führt, die größer ist als im optimalen Gleichgewicht des Netzes ohne die Schnellstraße. Dieses Paradox wurde 1968 von dem deutschen Mathematiker Dietrich Braess veröffentlicht[67].

> *There are many settings in which adding a new strategy to a game makes things worse for everyone. [...] We all have an informal sense that "upgrading" a network has to be a good thing, and so it is surprising when it turns out to make things worse.*
>
> Easley und Kleinberg (2010:S. 210)

Eines der bekanntesten Bestätigungen des Braess-Paradoxons ist das Renaturierungsprojekt des Flusses Cheonggyecheon im Zentrum Seouls. Der Abriss einer 1968 errichteten Schnellstraße über den Fluss zwischen 2003 und 2005 führte zu einer Beschleunigung des Verkehrsflusses in der Stadt insgesamt. Heute ist das Flussufer ein beliebter Treffpunkt.[68]

25.5 Systemische Risiken in Netzen

Die Gesellschaft für Informatik (`http://gi.de`) sieht die Beherrschung systemischer Risiken in weltweiten Netzen als eines der fünf größten Herausforderungen für die Informatik an.[69] Ein Risiko ist *systemisch* für ein Netzwerk, wenn es das gesamte Netz oder dessen Funktionsfähigkeit gefährdet. Wichtige Maßnahmen zum Schutz vor systemischen Risiken in Netzen identifizieren und begrenzen rechtzeitig Schocks, die es komplett durchlaufen können. Immer noch basieren solche Schutzmaßnahmen nur auf starren Trennlinien, fixen Puffern oder persönlichen Überprüfungen, für die jeder einzelne Knoten betrachtet werden muss. In sehr komplexen

[66] Easley und Kleinberg (2010):§8.1.

[67] Braess (1968).

[68] `https://grist.org/infrastructure/2011-04-04-seoul-korea-tears-down-an-urban-highway-life-goes-on/`

[69] `http://www.gi.de/themen/grand-challenges-der-informatik.html` [2016-02-08]

und dynamischen Netzwerken sind diese Maßnahmen allerdings nicht mehr realisierbar, da die einzelnen Knoten nicht in der erforderlichen Schnelligkeit überprüft werden können. Die Herausforderung an die Informatik ist es also, effiziente Interventionssysteme zur Identifikation systemrelevanter Knoten und zur Eindämmung entstehender systemischer Gefahren in komplexen und dynamischen Netzwerken zu entwickeln.

Ein wichtiges Beispiel für systemische Risiken sind Bankenkrisen und Börsenkrache. Finanzmärkte und Finanzinstitute bilden hochkomplexe und extrem dynamische Netzwerke, und gerade die Informatik hat mit Algorithmen und Infrastruktur seit Mitte der 1980er Jahre zur Komplexifizierung und Dynamisiserung beigetragen (siehe Abschnitt 28 ab Seite 179), seit 2010 beispielsweise durch den Hochfrequenzhandel. Im Zuge der Finanzkrise von 2008 führten Ausfälle US-amerikanischer Hypothekendarlehen zum Zusammenbruch von Finanzinstituten und sogar ganzer Staaten. Einserseits wäre durch eine korrekte Einschätzung der Lehman Bank im September 2008 als systemrelevanter Knoten der Ausbruch der weltweiten Bankenpanik vielleicht vermeidbar gewesen,[70], andererseits hätte es in der Folge sicher effizientere Unterstützungen für betroffene Unternehmen und Staaten gegeben als die durchgeführten Maßnahmen nach dem „Gießkannenprinzip“. In der aktuellen wirtschaftswissenschaftlichen Bankenforschung wird das systemische Risiko nach dem anfänglichen Zusammenbruch einzelner Unternehmen mit „Ansteckungsgefahr“ bezeichnet.[71] Für weitere Informationen zu systemischen Risiken an Finanzmärkten siehe auch[72], für spieltheoretische Untersuchungen.[73]

Ein weiteres Beispiel aus der Betriebswirtschaft sind Lieferketten (Supply Chains) bzw. Liefernetze, die Netzwerke von Gütertransporten darstellen. Ein charakteristisches systemisches Risiko von Liefernetzen ist der *Peitscheneffekt* (Bullwhip-Effekt), durch den geringe Nachfrage- oder Angebotsfluktuationen an den Enden des Liefernetzes aufgrund der Prognosen der einzelnen Akteure sich entlang einer Lieferkette in einer Art Resonanzkatastrophe zu extremen Fluktuationen aufschaukeln kann. Daneben kann ein Extremereignis mit niedriger Eintrittswahrscheinlichkeit, beispielsweise ein Erdbeben oder ein schweres Unwetter, systemische Schäden in dem Netzwerk bewirken. Zu weiteren Details dazu siehe Dadfar et al. (2012).

Systemische Risiken spielen desweiteren eine gesellschaftlich wichtige Rolle in Energienetzen. Insbesondere führt eine plötzliche Überlast eines Stromnetzes zum Netzausfall, wobei die Überlast durch erhöhte Nachfrage, durch gesellschaftliche oder globale Ereignisse (Fußball-WM) oder durch Ausfall von Kraftwerksleistung (Unwetter, Störungen, Leitungsunterbrechungen)[74].

25.5.1 Kaskaden und Viralität

Die Verbreitung einer Anwendung, beispielsweise eines Social Games, in einem Netzwerk gleicht der Ausbreitung einer Epidemie in der Bevölkerung oder der Verbreitung eines Gerüchts, einer Mode, einer Innovation oder einer Meinung. Gemäß dem ursprünglich auf Watts[75] zurückgehenden *Kaskadenmodell* stellen die Individuen oder „Aktoren“ Knoten eines gerichteten sozialen Netzwerks dar, die je einen von zwei möglichen Zuständen 0 oder 1 annehmen können, in Symbolen: $a(i) = 0$ oder $a(i) = 1$, wobei ein Aktor im Zustand 1 *aktiv* heißt. Je nach Kontext kann das beispielsweise „infiziert“, „überzeugt“ oder „begeistert“ bedeuten. Eine gerichtete Kante von Knoten i nach Knoten j hat eine Kantenbewertung $w_{ij} \in \mathbb{R}$, die die *Intensität* oder den *Einfluss* der Beziehung von i auf j darstellt. Ein Aktor i kann zu einem gegebenen Zeitpunkt nun nur dann in den Zustand 1 wechseln, wenn die normierte Summe der Intensitäten

[70] Rudolph (2014).
[71] Pflock (2014):S. 87f.
[72] Barth und Schnabel (2013); Barth und Schnabel (2014); Battiston et al. (2012); Mertens und Barbian (2014).
[73] Diamond und Dybvig (1983); Günther (2014).
[74] Beck et al. (2013); Kiyak und de Vries (2015); Kiyak und de Vries (2017); Rohjans et al. (2014).
[75] Watts (2002).

aller auf ihn weisenden aktiven Aktoren einen gegebenen Schwellenwert ϑ_i mit $0 \leqq \vartheta_i \leqq 1$ erreicht oder übersteigt:

$$a(i) \mapsto 1 \quad \Longleftrightarrow \quad \frac{\sum\limits_{j \text{ aktiv}} w_{ji}}{\sum\limits_{\text{alle } j} w_{ji}} \geqq \vartheta_i. \tag{25.33}$$

Betrachten wir beispielsweise den in Abbildung 25.15 skizzierten Ausschnitt aus einem Netzwerk. Hier sind drei der Nachbarn von Aktor i im Zustand 1. Bei einem Schwellenwert von

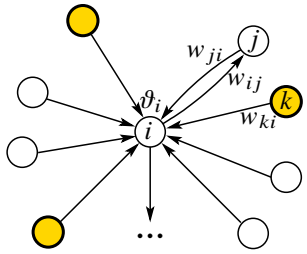


Abbildung 25.15: Drei von acht Nachbarn von Aktor i mit dem Schwellenwert ϑ_i sind aktiv.

$\vartheta_i = \frac{1}{2}$ und gleichen Intensitäten $w_{ji} = w_{ki} = \ldots$ würde also i den Zustand 0 behalten, in Symbolen $x(i) = 0$, bei $\vartheta_i = \frac{1}{3}$ dagegen würde er den Zustand 1 annehmen, $x(i) = 1$, er wäre also aktiv. Zur Bestimmung der individuellen Schwellenwerte ϑ_i kann eine Zufallsverteilung verwendet werden[76], aber auch mit einer Nutzenfunktion gemäß eines vernetzten Koordinationsspiels berechnet werden[77]. In der Regel sind in einem Netzwerk die meisten Aktoren im Zustand 0, nur einige wenige *Innovatoren (initial adopters)* sind im Zustand 1. Sie können im ersten Taktschritt ihre Nachbarn aktivieren, die *Erstanwender (early adopters)*, und die wiederum im nächsten Schritt deren Nachbarn. Auf diese Weise kann nach einigen Schritten eine Kaskade von Aktivierungen oder Infektionen eintreten, also eine Epidemie oder ein Modetrend.

Beispiel 25.18. Gegeben sei das in Abbildung 25.16 skizzierte Netzwerk mit 17 Knoten, wobei die zwei Knoten 9 und 6 aktiv seien, alle anderen inaktiv. Die beiden sind also die Innovatoren der Kaskade. Ferner nehmen wir an, dass die Schwellenwerte alle gleich $\frac{1}{2}$ sind, also $\vartheta_i = \frac{1}{2}$ für

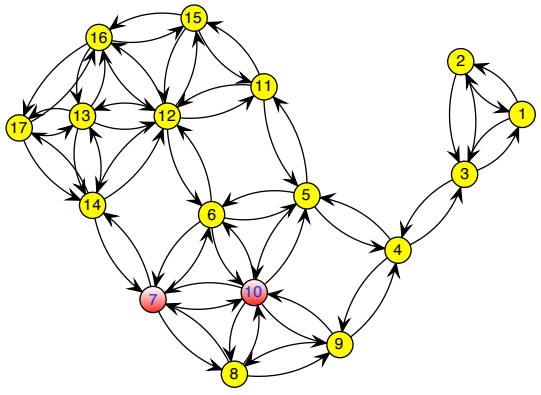


Abbildung 25.16: Netzwerk mit 17 Knoten und zwei Innovatoren. Seien alle Schwellenwerte $\vartheta_i = \frac{1}{2}$ und die Intensitäten aller eingezeichneten Kanten $w_{ij} = 1$.

$i = 1, 2, \ldots, 17$, und ebenso, dass alle nichtverschwindenden Intensitäten gleich 1 sind, d.h. $w_{ij} = 0$ oder 1. Dann werden im ersten Taktschritt Knoten 5 und 7 infiziert, danach 4 und 8, und

[76] Watts (2002).
[77] Easley und Kleinberg (2010):S. 499ff.

schließlich Knoten 3, also

Takt	0	1	2	3	4	⋯
Infizierte Knoten	$\{7, 10\}$	$\begin{Bmatrix}7, 10,\\6, 8\end{Bmatrix}$	$\begin{Bmatrix}7, 10,\\6, 8,\\5, 9\end{Bmatrix}$	$\begin{Bmatrix}7, 10,\\6, 8,\\5, 9,\\4\end{Bmatrix}$	$\begin{Bmatrix}7, 10,\\6, 8,\\5, 9,\\4\end{Bmatrix}$	⋯

(25.34)

In jedem weiteren Taktschritt wird kein neuer Knoten mehr infiziert, die Kaskade ist gesättigt. Die Innovation hat sich nur im Cluster der beiden Innovatoren durchsetzen können. □

Das Kaskadenmodell verallgemeinert klassische epidemiologische Modelle von Infektionsausbreitungen (wo ein einziger infektiöser Nachbar zur Infektion eines Knotens ausreicht, also alle ϑ_i nahe 0 sind), Perkolationsmodelle (wo eine feste Anzahl besetzter Nachbarn zur Besetzung eines Knotens ausreicht) und zufallsverteilte Ising-Modelle oder Mehrheitswahlsysteme (die homogene Netzwerke voraussetzen).

In einem Kaskadennetzwerk wird die *Viralität* v als die Anzahl k der infizierten Knoten pro Zeit t definiert:

$$v = \frac{k}{t}. \tag{25.35}$$

Die Viralität einer Anwendung in einem sozialen Netz, insbesondere eines Social Games, bezeichnet ihre Verbreitungsgeschwindigkeit[78] in dem Netzwerk.

Beispiel 25.19. Gegeben sei das Netzwerk aus Beispiel 25.18. Dann lautet die Viralität v_j im Taktschritt j

$$v_1 = 2, \qquad v_2 = 2, \qquad v_3 = 1, \qquad v_4 = 0, \ldots \tag{25.36}$$

Die durchschnittliche Viralität $v(j)$ nach Takt j beträgt demgegenüber $v(1) = \frac{2}{1} = 2$, $v(2) = \frac{4}{2} = 2$, $v(3) = \frac{5}{3}$, $v(4) = \frac{5}{4}$; ferner $v(j) \to 0$ für $j \to \infty$. □

Beispiel 25.20. *(Pandemie H1N1/2009)* Im März 2009 wurde erstmals von einem neuartigen H1N1-Grippevirus („Schweinegrippe“) berichtet, an dem wahrscheinlich bereits im Januar in Mexiko Menschen erkrankten[79] und im frühen April 2009 erste Todesfälle registriert wurden.[80] Die WHO erklärte am 11. Juni 2009 den H1N1-Ausbruch zur Pandemie,[81] also zu einer weltweiten Epidemie. Am 26. April 2009 wurden weltweit 38 bestätigte Fälle registriert, bis zum 5. Juli 2009 waren es 27 480.[82] Damit betrug die durchschnittliche Viralität

$$v_{\mathrm{H1N1}} = \frac{27\,480 - 7}{72\ \mathrm{d}} = 381{,}57\ \mathrm{d}^{-1}. \tag{25.37}$$

Am 10. August 2010 erklärte die WHO das Ende der Pandemie.[83] Obwohl die Pandemie am Ende weniger Erkrankungen und mit 18 036 Fällen weniger Tote forderte als eine übliche Grippewelle, war die Besorgnis anfangs sehr groß, da es sich bei dem Virus H1N1 um einen engen Verwandten des Erregers der Spanischen Grippe von 1918 handelte, die mit 500 Millionen Fällen fast ein Drittel der damaligen Menschheit infizierte und 50 Millionen, vielleicht sogar 100 Millionen Tote forderte.[84] □

[78] Rink (2012):S. 38.

[79] `http://www.reuters.com/article/2009/06/11/idUSN11399103` [2015-08-29]

[80] `http://www.nytimes.com/2009/04/27/health/27questions.html` [2015-08-29]

[81] `http://www.who.int/mediacentre/news/statements/2009/h1n1_pandemic_phase6_20090611/en/` [2015-08-29]

[82] `http://www.theairdb.com/swine-flu/heatmap.html` [2015-08-29]

[83] `http://www.euro.who.int/en/what-we-do/health-topics/communicable-diseases/influenza/news/news/2010/08/who-director-general-declares-h1n1-pandemic-over` [2015-08-29]

[84] (Johnson und Mueller (2002)), `http://en.wikipedia.org/wiki/2009_flu_pandemic#Comparisons_to_other_pandemics_and_epidemics` [2015-08-27]

Beispiel 25.21. *(FarmVille und FishVille)* Die Firma Zynga veröffentlichte im Juni 2009 auf Facebook die Spiele-App FarmVille, die nach fünf Monaten 25 Millionen Spieler erreichte. Das Nachfolgespiel FishVille kam nach bereits einer Woche auf sechs Millionen Spieler[85]. Die Viralitäten betrugen also durchschnittlich

$$v_{\text{FarmV}} = \frac{25\text{ Mio}}{150\text{ d}} \approx 167\,000\text{ d}^{-1}, \qquad v_{\text{FishV}} = \frac{6\text{ Mio}}{7\text{ d}} \approx 857\,000\text{ d}^{-1}. \tag{25.38}$$

In Einheiten pro Sekunde ausgedrückt betragen die Viralitäten also $v_{\text{FarmV}} \approx 1{,}9\text{ s}^{-1}$ und $v_{\text{FishV}} \approx 9{,}9\text{ s}^{-1}$. □

25.6 * Ramsey-Zahlen

Für $r, s \in \mathbb{N}$ ist die *Ramsey-Zahl* $R(r, s)$ die Mindestgröße einer Gruppe in einem sozialen Netzwerk, so dass entweder mindestens r Akteure sich gegenseitig kennen oder mindestens s Akteure sich gegenseitig nicht kennen. Hierbei bedeutet „gegenseitig kennen“ eine symmetrische Beziehung, das heißt es gibt nicht den Fall, dass ein Akteur einen anderen kennt, der ihn jedoch nicht kennt.[86] Nach dem Satz von Ramsey[87] existiert eine solche Zahl tatsächlich für jedes Paar (r, s). Damit ist das sogenannte *Party-Problem* eindeutig lösbar: *Wieviele Personen muss man zu einer Party einladen, so dass sich r Leute gegenseitig kennen oder s Leute sich gegenseitig unbekannt sind?*

Per Definition gilt

$$R(r, 1) = 1. \tag{25.39}$$

Eine erste wichtige Eigenschaft der Ramsey-Zahlen ist ihre Symmetrie bezüglich r und s,

$$R(r, s) = R(s, r). \tag{25.40}$$

Diese Symmetrie ergibt sich sofort mit dem Argument, dass die Mindestgröße gleich bleibt, wenn man die Bedeutung „kennen“ und „nicht kennen“ austauscht.

Lemma 25.22. *$R(r, 2) = r$ für alle $r \in \mathbb{N}$.*

Beweis. Man sieht sofort ein, dass in einer Gruppe mit r Mitgliedern sich entweder alle kennen oder eben mindestens 2 nicht. Mathematisch bedeutet das $R(r, 2) \leqq r$. Da andererseits für eine Gruppengröße $r - 1$ der Fall eintreten kann, dass sich alle kennen (d.h. wir hätten $(r - 1, 1)$), muss die Gruppengröße auch mindestens r sein, also $R(r, 2) \geqq r$. □

Beispiel 25.23. Mit Lemma 25.22 gilt $R(3, 2) = 3$, d.h. schon für eine Gruppe mit drei Personen kennen sich entweder alle drei oder mindestens zwei kennen sich nicht. In der Tat kann man sämtliche Fälle auflisten:

- Es kennen sich alle drei (erfüllt $r = 3$).
- Eine Person kennt zwei, die sich aber nicht kennen ($s = 2$).
- Zwei Personen kennen sich, aber die Dritte nicht ($s = 2$)

[85] Rink (2012):S. 38.

[86] Mathematisch lässt sich die Ramsey-Zahl $R(r, s)$ für zwei natürliche Zahlen $r, s \geq 2$ präziser definieren als die Mindestgröße, die ein vollständiger Graph, dessen Kanten alle entweder rot oder blau eingefärbt sind, haben muss, so dass er eine rote r-Clique oder eine blaue s-Clique enthält.

[87] F.P. Ramsey (1930): ‘On a problem of formal logic’, *Proc. London Math. Soc. Series 2*, **30**, pp 264–286, doi: 10.1112/plms/s2-30.1.264

- Niemand kennt sich ($s = 3$).

Andererseits wird sofort klar, dass $R(3,2)$ nicht 2 sein kann, denn für den Fall, dass sich die beiden Personen kennen, ist $r = 2$ und $s = 0$. □

Satz 25.24 (Erdős & Szekeres 1935). [88] *Für $r, s \in \mathbb{N}$ mit $r, s \geqq 2$ gilt*

$$\boxed{R(r,s) \leqq R(r-1,s) + R(r,s-1).} \tag{25.41}$$

Beweis. Betrachten wir eine Gruppe V mit $R(r-1,s) + R(r,s-1)$ Akteuren. Für einen Akteur $v \in V$ zerfällt dann die Gruppe in die Menge M der Bekannten und die Menge N der Unbekannten von v, also $V = M \cup N \cup \{v\}$. Die Gruppe hat also

$$R(r-1,s) + R(r,s-1) = |M| + |N| + 1$$

Akteure, d.h. entweder $|M| \geqq R(r-1,s)$ oder $|N| \geqq R(r,s-1)$. Im ersten Fall kennen sich in M entweder mindestens $r-1$ Akteure, also kennen sich alle r in $M \cup \{v\}$, oder es kennen sich s Akteure in M überhaupt nicht; damit gilt insgesamt die Behauptung für diesen Fall. Der zweite Fall $|N| \geqq R(r,s-1)$ ergibt die Behauptung analog. □

Mit $s = 2$ liefert der Satz von Erdős und Szekeres also $R(r,2) = R(r-1,2) + R(r,1)$, d.h. mit Lemma 25.22 $r = (r-1) + R(r,1)$. Die auf den ersten Blick willkürlich festgelegte Definition (25.39) ist also sinnvoll. Man kann die Ungleichung (25.41) etwas verschärfen, es gilt $R(r,s) < R(r-1,s) + R(r,s-1)$, falls $R(r-1,s)$ und $R(r,s-1)$ gerade sind.

Korollar 25.25. *Für $r, s \in \mathbb{N}$ mit $r, s \geqq 2$ gilt*

$$R(r,s) \leqq \binom{r+s-2}{s-1}. \tag{25.42}$$

Beweis. Definieren wir die Funktion $f(r,s) = \binom{r+s-2}{s-1}$. Aus der Rekursionsgleichung für Binomialkoeffizienten,

$$\binom{n+1}{k+1} = \binom{n}{k+1} + \binom{n}{k},$$

also mit $n = r+s-3$ und $k = s-2$

$$\binom{r+s-2}{s-1} = \binom{r+s-3}{s-1} + \binom{r+s-3}{s-2} = \binom{(r-1)+s-2}{s-1} + \binom{r+(s-1)-2}{(s-1)-1},$$

folgt für f die rekursive Gleichung $f(r,s) = f(r-1,s) + f(r,s-1)$. Da zudem $R(2,2) = 2 = f(2,2)$, gilt dann mit Satz 25.24 stets $R(r,s) \leqq f(r,s)$. □

Lemma 25.26. $R(3,3) = 6$.

Beweis. Es ist klar, dass $R(3,3) \geqq 5$ sein muss, denn in jeder Gruppe kann der Fall eintreten, dass sich nur 2 kennen und alle anderen nicht. In einer Gruppe mit 5 Akteuren kann es jedoch zu einer Konstellation kommen, in der jeder genau zwei andere kennt und die verbleibenden beiden nicht, siehe Abb. 25.17. Also gilt sogar $R(3,3) \geqq 6$. Andererseits folgt mit der rekursiven Abschätzung (25.41) mit $r = s = 3$, der Symmetrie (25.40) und Lemma 25.22 sofort $R(3,3) \leqq R(2,3) + R(3,2) = 2\,R(3,2) = 6$, also insgesamt $R(3,3) = 6$. □

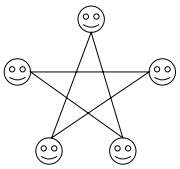


Abbildung 25.17: Eine Gruppe mit fünf Akteuren, in der jeder zwei andere kennt und zwei nicht.

r \ s	3	4	5	6	7	8	9
3	6	9	14	18	23	28	36
4	9	18	25				
5	14	25	?				
6	18			?			
7	23				?		
8	28					?	
9	36						?

Tabelle 25.2: Die bislang bekannten Ramsey-Zahlen für $3 \leq r, s \leq 9$, Stand August 2011. Beachte die Symmetrie bezüglich der Hauptdiagonalen. (Quellen:[89], Stanisław Radziszowski: *Small Ramsey numbers.* `http://www.combinatorics.org/issue/view/Surveys`)

Man ahnt bei der komplizierten Berechnung schon einer solch kleinen Ramsey-Zahl wie $R(3,3)$, wie schwierig die Berechnung für große Ramsey-Zahlen sein muss. In der Tat gibt es für die meisten Ramsey-Zahlen nur Abschätzungen, der genaue Wert ist nur für wenige bekannt, siehe Tabelle 25.2. Man weiß also insgesamt sehr wenig über größere Ramsey-Zahlen. Ihre Berechnung liegt weit jenseits aller heute möglichen Rechenkapazitäten. Die Ursache dafür ist, dass im Wesentlichen kein besserer Algorithmus bekannt ist, als alle möglichen Kombinationen von Bekanntschaften in einer Gruppe der Größe n zu enumerieren. Da es $\binom{n}{2} = n(n-1)/2$ Kanten in einem Graphen der Größe n geben kann und jede einzelne vorhanden oder nicht vorhanden sein kann, erhalten wir für die Überprüfung (!), ob in einer Gruppe der Größe n sich r kennen oder s nicht, eine Zeitkomplexität von

$$T(n) = O(2^{n(n-1)/2}). \tag{25.43}$$

Beispielsweise ist bekannt, dass $43 \leqq R(5,5) \leqq 49$. Um zu zeigen, dass $R(5,5) \geqq 44$, müsste man ein Netzwerk mit $n = 43$ Knoten finden, in dem sich weder 5 Akteure gegenseitig kennen, noch 5 sich unbekannt sind. Da $\binom{43}{2} = 903$, gilt es $2^{903} \approx 10^{271}$ Graphen zu überprüfen. Selbst wenn es gelänge, in einer Sekunde 10^9 Graphen zu prüfen, also pro Jahr $3 \cdot 10^{15}$ Graphen, benötigte man immer noch etwa $3 \cdot 10^{255}$ Jahre! (Das Weltall ist wahrscheinlich etwa 10^{10} Jahre alt.) Für einige Spezialfälle jedoch kennt man immerhin die Abschätzungen

$$\lfloor 2^{k/2} \rfloor \leqq R(k,k) \leqq \lceil 2^{2k-3} \rceil \qquad \text{und} \qquad R(3,k) \leqq \frac{k^2+3}{2} \tag{25.44}$$

[90],[91], die bereits aus den frühen Resultaten von Ramsey (1930) und Erdős (1947) folgten, sowie die etwas verbesserten Abschätzungen

$$\frac{k\,2^{k/2}}{\mathrm{e}\sqrt{2}} < R(k,k) \leqq \frac{4^k}{\sqrt{k}}, \qquad \text{und} \qquad c_1 \frac{k^2}{\ln k} \leqq R(3,k) \leqq c_2 \frac{k^2}{\ln k} \tag{25.45}$$

mit einer (unbekannten) Konstanten $c_1 > 0$ und einer Konstanten $c_2 \leqq \frac{5}{12}$.[88, 92]

to obliterate – auslöschen

Suppose aliens invade the earth and threaten to obliterate unless human beings

[88] Catherine Greenhill (2005): *Ramsey Numbers*, *Parabola* **41**(3) `http://www.parabola.unsw.edu.au/vol41_no3`

[90] Diestel (2000):Sätze 7.1.1, 9.1.3.

[91] Harris et al. (2008):Theorem 1.65.

[92] E.W. Weisstein: "Ramsey Number." `http://mathworld.wolfram.com/RamseyNumber.html`

> *can find the Ramsey number* $[R(5,5)]$ *for red five and blue five. We could marshal the world's best minds and fastest computers, and within a year we could probably calculate the value. If the aliens demanded the Ramsey number* $[R(6,6)]$ *for red six and blue six, however, we would have no choice but to launch a preemptive attack.*
>
> Paul Erdős, `http://en.wikiquote.org/wiki/Paul_Erd%C5%91s`

to marshal – aufstellen

preemptive – Präventiv-

25.6.1 Gerichtete Ramsey-Zahlen

Ramsey-Zahlen können auch für spezielle gerichtete Graphen definiert werden, sogenannte Turniere. Ein *Turnier (tournament)* oder ein *Turniergraph*[93] ist ein Graph, in dem jedes Knotenpaar durch genau eine gerichtete Kante verbunden ist. Ein solcher Graph kann ein Rundenturnier (*round-robin tournament*) ohne Unentschieden repräsentieren, d.h. ein Turnier mit dem Austragungsmodus „jeder gegen jeden". Hierbei wird jeder Spieler bzw. jede Mannschaft als Knoten dargestellt und jedes Spiel als eine auf den Gewinner gerichtete Kante zwischen den beiden Gegnern. Ein Unentschieden kann durch eine ungerichtete Kante dargestellt werden.

Ein Turnier ist genau dann azyklisch, oder transitiv, wenn es keinen 3-Zyklus enthält, sondern für drei Knoten u, v, w stets

$$(u \to v),\ (v \to w)\ \Rightarrow\ (u \to w).$$

gilt. Ein Turnier enthält stets mindestens einen Hamilton-Pfad, ein transitives Turnier dagegen enthält genau einen Hamilton-Pfad. Zudem ist ein transitives Turnier sortierbar: Die Knoten ergeben eine Rangfolge anhand der Anzahl ihrer Außengrade, die wiederum eine strikt monoton wachsende Folge ergibt, die *Punktefolge (score sequence)*.[94]

Beispiel 25.27. In der Gruppenphase einer Fußball-Weltmeisterschaft spielen in acht Gruppen je vier Mannschaften gegeneinander. Jede Gruppe spielt dabei ein Rundenturnier. Es gibt also insgesamt $\binom{4}{2} = 6$ Spiele pro Gruppe, ausgetragen in drei Runden à zwei Spielen. Für die Gruppe

Gruppe E (WM 1994)		
Irland	– Italien	1:0
Norwegen	– Mexiko	1:0
Italien	– Norwegen	1:0
Mexiko	– Irland	2:1
Italien	– Mexiko	1:1
Irland	– Norwegen	0:0

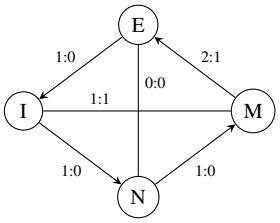


Abbildung 25.18: Turniergraph mit Unentschieden der Gruppe E bei der WM 1994 in den USA. Der Graph ist kein Turniergraph.

E der WM 1994 in den USA beispielsweise mit den Mannschaften Irland (E), Italien (I), Mexiko (M) und Norwegen (N) ergibt der Graph in Abbildung 25.18. Diese Gruppe ist bislang die einzige in der Geschichte der Fußballweltmeisterschaften, in der alle Teams dieselbe Punkteanzahl erreichten. Da es Unentschieden gab, kann das Turnier nicht durch einen Turniergraphen dargestellt werden. Einer der wenigen nichttransitiven Turniergraphen einer WM ist in Abbildung 25.19 wiedergegeben. In diesem Turniergraphen gibt es die drei Hamilton-Pfade G–A–B–N, G–B–N–A und G–N–A–B. Die Punktefolge des Graphen lautet $\{1, 1, 1, 3\}$.

Der einzige Turniergraph der WM 2006 in Deutschland wurde in Gruppe A ausgespielt, wie in Abbildung 25.20 illustriert. Er stellt einen transitiven Graphen dar. Die Punktefolge dieses

[93] Diestel (2000):S. 226.

[94] E. Weisstein, `http://mathworld.wolfram.com/ScoreSequence.html`

Gruppe D (WM 1994)		
Argentinien	– Griechenland	4:0
Nigeria	– Bulgarien	3:0
Argentinien	– Nigeria	2:1
Bulgarien	– Griechenland	4:0
Griechenland	– Nigeria	0:2
Argentinien	– Bulgarien	0:2

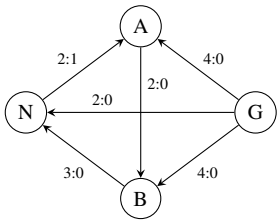


Abbildung 25.19: Turniergraph der Gruppe D bei der WM 1994 in den USA. Der Graph ist nicht transitiv, er enthält einen Kreis.

Gruppe A (WM 2006)		
Deutschland	– Costa Rica	4:2
Polen	– Ecuador	0:2
Deutschland	– Polen	1:0
Ecuador	– Costa Rica	3:0
Ecuador	– Deutschland	0:3
Costa Rica	– Polen	1:2

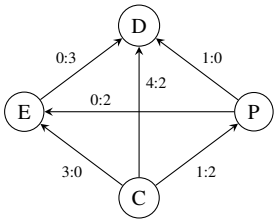


Abbildung 25.20: Der Turniergraph der Gruppe A bei der WM 2006 in Deutschland, der einzigen Gruppe dieser WM, bei der es kein Unentschieden gab. Der Graph ist transitiv.

Turniers ist $\{0, 1, 2, 3\}$ und induziert die Rangfolge der Mannschaften D–E–P–C. Dies ist gleichzeitig der einzige Hamilton-Pfad dieses Turniergraphen. Die Adjazenzmatrizen zu den drei obigen Graphen lauten:

$$\begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix} \tag{25.46}$$

Eine Turniermatrix ist also asymmetrisch und eine transitive Turniermatrix lässt sich durch Zeilenvertauschungen in eine Dreiecksmatrix umformen. □

Die „gerichtete Ramsey-Zahl" $R(k)$, oft auch $\vec{R}(k)$, ist für ein $k \in \mathbb{N}$ definiert als die kleinste Zahl n, so dass jedes Turnier mit $\geqq n$ Knoten ein azyklisches Turnier mit n Knoten enthält. Per Definition gilt

$$R(1) = 1. \tag{25.47}$$

Man kann zeigen, dass jedes Turnier mit n Knoten ein azyklisches Turnier mit $\log_2 n$ Knoten enthält, d.h. $R(k) \leqq 2^{k-1}$. Man kennt die folgenden Werte und Abschätzungen:

k	1	2	3	4	5	6	7
$R(k)$	1	2	4	8	14	28	$[32, 55]$

(25.48)

Googles wichtigstes Produkt wird nicht mehr die Suche sein, sondern Künstliche Intelligenz.

Kevin Kelly (http://www.wired.com/2014/10/future-of-artificial-intelligence/)

26 Künstliche Intelligenz

Kapitelübersicht

Das Gebiet der Künstlichen Intelligenz macht seit Mitte der 2010er Jahre spektakuläre Fortschritte. Im März 2014 stellte Facebook das System DeepFace vor, das aus einer großen Menschenmenge einzelne Gesichter besser erkennen kann als die meisten Menschen, 2015 lernte das Programm Deep Q Network selbständig die Regeln und Gewinnstrategien von 49 Atari-Spielen, im März 2016 schlug AlphaGo den amtierenden Weltmeister in dem Spiel Go, und Programme wie Cortana, Siri oder Google Now können Spracheingaben so gut wie fehlerfrei erkennen. Fast alle diese Programme basieren auf dem Konzept des Deep Learning, bei dem ein vielschichtiges neuronales Netz selbständig aus beobachteten Daten stochastische Vorhersagen über künftige Daten berechnet oder sie kategorisiert. Die meisten dieser neuronalen Netze benötigen dazu Trainingsdaten, die letztendlich nichts weiter als eine durch Menschen geschaffene Wissensbasis sind (AlphaGo beispielsweise wurde mit 30 Millionen von Menschen gespielten Go-Spielen trainiert), manche aber lernen wie ein Lebewesen selbständig und über Rückkopplungen, sich erfolgreich in der Umwelt zurecht zu finden, wie sie sie vorfinden, so zum Beispiel Deep Q

Network. Nach einer Studie des Pentagon werden Computerprozessoren wahrscheinlich in den 2020er Jahren die Rechenleistung des menschlichen Gehirns erreichen. In ein paar Jahrzehnten wird eine einzige relativ billige Maschine über die rohe Rechenleistung der gesamten Menschheit verfügen[1,2].

Diese Entwicklung wirft einige wichtige Fragen auf, eine uralte und zwei aktuelle. Erstens: Was eigentlich ist Intelligenz? Zweitens: Sind die Maschinen und Algorithmen, die wir geschaffen haben und noch schaffen werden, intelligent? Drittens: Welche gesellschaftlichen, politischen und rechtlichen Maßnahmen müssen wir ergreifen, um die Gefahren dieser Entwicklung zu vermeiden und ihre Chancen für das Gemeinwohl zu nutzen? Diese Fragen werden wir in diesem Kapitel behandeln.

26.1 Überblick, Einordnung und Begriffe

"Artificial intelligence is a [...] field that is increasingly utilized in robotic and other control systems, but its focus is on developing intelligent machines or systems, and it is sometimes defined as a branch of computer science. However, today it involves a wide range of research and problem areas, such as reasoning, planning, learning, perception, and environmental and situational awareness, which are studied by a broad range of disciplines, including engineering, psychology, and philosophy. The development of 'weak AI,' namely, the cognitive ability to solve specific problems or perform specific tasks, has been demonstrated in a broad range of applications over several decades. It is becoming increasingly important to facilitate many 'smart' technologies now available commercially. However, the development of 'strong AI,' or true general intelligence and creativity similar to a human brain, is a long-term goal that some believe may never be realized.

Robotics, artificial intelligence, and human augmentation have become very important technologies not only because of their military and industrial applications, but also as a burgeoning economic sector and a potentially transformative social driver."[3]

"Machine learning is at the heart of modern approaches to artificial intelligence. The field posits that teaching computers how to learn can be significantly more effective than programming them explicitly. This idea has revolutionized what computers can do in a wide range of domains, including Intelligence, Surveillance, and Reconnaissance (ISR); Natural Language Processing (NLP); Predictive Analytics; Cyber; and various scientific disciplines. Example applications include: self-driving cars, image search and activity detection, object tracking, topic models, spam filters, recommender systems, predictive databases, and gene sequencing. Unfortunately, building effective machine learning applications currently requires Herculean efforts on the part of highly trained experts in machine learning." [4]

26.2 Was ist Intelligenz?

Intelligenz ist ein schwer zu fassender Begriff. Die Psychologie versucht dieses zentrale Konzept ihrer Wissenschaft in der Regel mit der Messung des Intelligenzquotienten (IQ) zu verstehen, der jedoch auf einer theoretisch eher umstrittenen Mischung intelligenter Fähigkeiten beruht. Viele Wissenschaftler wie Howard Gardner oder Donald Hoffman behaupten, es gebe nicht eine

[1] Kadtke und Wells II (2014):S. 47.
[2] Schlieter (2015):S. 35.
[3] Kadtke und Wells II (2014):S. 43.
[4] `https://www.fbo.gov/utils/view?id=1728bcabdd35609b662d0284e2665f76`

einheitliche Intelligenz, sondern mehrere verschiedene, nämlich neben der rationalen vor allem die emotionale und die visuelle Intelligenz[5].

26.2.1 Denken

Das zentrale Wesen rationaler Intelligenz ist die Fähigkeit zu *denken*. Denken ist eine bewusste geistige Aktivität und umfasst die mentale Entwicklung einer Idee, Einsicht oder Absicht sowie die geistigen Vorgänge, die am Abwägen, Überlegen, Nachsinnen, Reflektieren, Erinnern, Sich-Vorstellen, Planen, Erschaffen, Vorausahnen und Folgern beteiligt sind.[6] Insbesondere setzt Denken also Bewusstsein voraus, kann sehr zielgerichtet oder sehr ungerichtet eingesetzt werden, und viel oder wenig Wissen erfordern. In der Kognitionspsychologie werden oft drei Hauptkategorien des Denkens betrachtet, nämlich Problemlösen, Expertenproblemlösen und logisches Denken.[7]

Zwar wird mit diesen Betrachtungen der Begriff der rationalen Intelligenz und des Denkens konkretisiert und plausibel unterteilt. Aber eine Definition ist das bei genauerem Hinsehen nicht: „Denken ist die Fähigkeit zu bewusster geistiger Aktivität“ reduziert den Begriff „Denken“, und somit auch „Intelligenz“, auf zwei andere Begriffe, nämlich „Bewusstsein“ und „geistige Aktivität“. Da Bewusstsein und geistige Aktivität wiederum offenbar die Fähigkeit zu selbstreflexivem Denken, also Intelligenz, benötigen, drehen wir uns hier logisch im Kreis.

Es erhebt sich sofort die grundsätzliche Frage: Können wir Intelligenz überhaupt mit unserer Intelligenz erklären? Der Informatiker Douglas Hofstadter sieht kein grundsätzliches Hindernis darin: „Es könnte [...] sein, dass unser Gehirn zu schwach ist, um sich selber zu verstehen. Man schaue sich die niedere Giraffe an. Ihr Gehirn ist offensichtlich weit unter dem Niveau, das für Selbsterkenntnis nötig wäre, ist aber unserem Gehirn verblüffend ähnlich. Tatsächlich arbeiten die Gehirne von Giraffen, Elefanten und Büffeln – sogar die Gehirne von Schildkröten oder unbekannten Wesen – vermutlich alle nach denselben Prinzipien. Giraffen finden sich wohl weit unter dem Intelligenzniveau, welches nötig wäre, um zu verstehen, wie diese Prinzipien sich zur Erzeugung von Eigenschaften des Denkens zusammenfügen. Menschen sind dann vielleicht näher an der Schwelle, vielleicht gerade noch unter ihr, vielleicht darüber. Wesentlich ist, dass es keinen *grundsätzlichen* [...] Grund gibt, warum diese Eigenschaften unverständlich sind; vielleicht sind sie intelligenteren Wesen vollkommen klar.“[8] Hofstadter hält eine erweiterte Form der Rekursion („tangled recursion“ – „verwickelte Rekursion“) für ein wesentliches Element jeder Intelligenz und stellt fest, „dass auf geeignete Weise komplizierte rekursive Systeme stark genug sein können, um aus jedem vorgegebenen Muster auszubrechen. Und ist das nicht eines der bestimmenden Merkmale der Intelligenz? Anstatt lediglich die aus Prozeduren zusammengesetzten Programme zu betrachten, die sich rekursiv *aufrufen* können, warum nicht wirklich raffiniert sein und Programme entwerfen, die sich selbst *verändern* können? [...] Diese Art »verwickelter Rekursion« ruht vermutlich im Kern der Intelligenz.“[9]

26.2.2 Hutters Begriff der universellen Intelligenz

Versuchen wir also mit Hofstadters Ermutigung, Intelligenz mit Begriffen der Informatik zu definieren. Ein naheliegender Ansatz könnte lauten: Intelligenz ist die Fähigkeit, Regelmäßigkeiten und Strukturen zu erfassen. Diese Fähigkeit impliziert die Vorhersage zukünftiger Ereignisse und

[5] Hoffman (2000):S. 9f.
[6] J. L. Gould und C. G. Gould (1997):S. 82.
[7] Eysenck und Keane (1995):S. 355ff.
[8] Hofstadter (1986):S. 754.
[9] Hofstadter (1986):S. 164.

bietet so einen evolutionären Vorteil, denn ihr Träger kann sich die erkannten Regelmäßigkeiten durch entsprechend angepasstes Verhalten zu Nutze machen.

Natürlich ist auch diese Aussage über Intelligenz im Kern zirkulär, denn was sind Regelmäßigkeiten oder Strukturen ohne eine Intelligenz, die sie feststellt? In Begriffen der Informatik ausgedrückt jedoch bedeutet die Aussage konkreter (aber auch eingeschränkter): Intelligenz ist die Fähigkeit, Daten zu komprimieren und zukünftige Daten vorauszusagen.

Intelligenz zeichnet sich demnach vor allem durch die Fähigkeit zur Induktion aus, also zum abstrahierenden Schließen vom Beobachteten zu einer Erkenntnis. Ob und in welcher Weise Induktion möglich ist, wurde seit Aristoteles in der Philosophie lange kontrovers diskutiert. Diese Betrachtungen führten vor allem zu Epikurs Prinzip der mehrfachen Erklärungen, Ockhams Rasiermesser und Bayes Regel der bedingten Wahrscheinlichkeiten.

Das Epikur'sche Prinzip der mehrfachen Erklärungen besagt: Wo mehrere Erklärungen von Phänomenen nicht in Widerspruch zur wahrnehmbaren Wirklichkeit geraten, stehen sie gleichberechtigt nebeneinander.[10]

Das Ockham'sche Rasiermesser ist ein erkenntnistheoretisches Prinzip, das besagt: „Von allen mit den Beobachtungen vereinbarenden Erklärungen eines Sachverhalts ist die einfachste zu bevorzugen". In der Sprache der Informatik ausgedrückt lautet es: Von allen Programmen, die eine gegebene Reihe von Daten reproduzieren, ist das kürzeste zu bevorzugen. Die Länge dieses Programms ist die *algorithmische Information*.[11]

Zu Beginn der 1960er Jahre kombinierte Ray Solomonoff diese Prinzipien und Begriffe zu seiner Theorie des universellen induktiven Schließens, das aus der Beobachtung von Daten der Umwelt Vorhersagen über zukünftige Daten macht. Es verknüpft dabei das Epikur'sche und das Ockham'sche Prinzip mit der Bayes'schen Regel zu einer Wahrscheinlichkeitsverteilung möglicher zukünftiger Daten, der *algorithmischen Wahrscheinlichkeit*.

Auf Basis dieses Informationsbegriffs gab der deutsche Informatiker Marcus Hutter 2005 die folgende Definition.

Definition 26.1. (Hutter, 2005) Das *Maß der universellen Intelligenz* ist definiert als der über alle vorstellbaren Umwelten gemittelte Erfolg einer Strategie. □

Diese auf den ersten Blick wenig aufschlussreiche Formulierung wird dadurch interessant, dass sich alle in ihr vorkommenden Begriffe im Sinn der Informatik interpretieren lassen: „Strategie" ist ein beliebig leistungsfähiges Progamm, „Umwelt" sind alle Daten, mit denen sie konfrontiert werden könnte – unter der relativ schwachen Voraussetzung, dass diese Daten nicht vollkommen willkürlich sind, sondern gewissen Gesetzmäßigkeiten gehorchen, so wie unsere natürliche Umwelt den Gesetzen der Physik unterliegt. „Erfolg" schließlich ist zu verstehen als Maximierung einer sinnvoll definierten Zielfunktion, die typischerweise das eigene Überleben und die Erzeugung zahlreicher Nachkommen umfasst[12].

Hutters Definition 26.1 liefert eine strenge Definition von Intelligenz auf der Basis wohldefinierter Konzepte der Informatik. Allerdings reduziert diese Definition den Begriff der universellen Intelligenz auf ein Optimierungsproblem bezüglich einer vorzugebenden Zielfunktion. Konzepte wie Bewusstsein oder Denken spielen dabei keine Rolle. Mit anderen Worten kann ein Softwareprogramm ein hohes Maß an universeller Intelligenz besitzen, ohne bewusst denken zu können. Wir Menschen müssen also akzeptieren, dass Maschinen Denkleistungen auf eine vollkommen andere Art und Weise als wir erzielen, und möglicherweise sind wir nicht einmal in

[10] Titus Lucretius Carus (ca. 50 n. Chr.): *Of the Nature of Things.* Project Gutenberg EBook **785**, Book VI, Line 9549–9560, `http://www.gutenberg.org/ebooks/785`

[11] de Vries (2006).

[12] Delahaye (2016a).

der Lage, sie verstehen[13]. Grundsätzlich scheint jedoch ein Intelligenzbegriff ohne Bewusstsein das Äußerste zu sein, was wir mit unserer Intelligenz definieren können.

26.2.3 Der Turing-Test

Alan Turing schlug 1950 ein ihm zu Ehren später *Turing-Test* genanntes Verfahren vor, um zu überprüfen, ob ein gegebenes System intelligent ist. Eine Reihe von Gutachtern führt dabei über einen elektronischen Kanal – z.B. über das Internet – mit einem unbekannten System einen schriftlichen Dialog. Wenn die Experten nicht unterscheiden könnten, ob ihr Korrespondenzpartner ein Mensch oder eine Maschine ist, müsste man ihm Intelligenz zuschreiben[14].

Auf diese Weise wird das Erkennen von Intelligenz auf die Beobachtung intelligenten Verhaltens reduziert. Da ein Mensch implizit als ein intelligentes Wesen betrachtet wird, wenden wir mit dem Turing-Test dieselben Kriterien zur Bewertung der Intelligenz eines unbekannten Systems an, wie wir es bei anderen Menschen auch tun. Für den Turing-Test benötigen wir keine Definition von Intelligenz.

Bisher hat noch kein künstliches System den Turing-Test bestanden, und es ist auch keines nur knapp gescheitert[15].

26.3 Rechenleistung von Mensch und Computer

Als Maß für die gesamte Rechenleistung eines Rechnersystems, also eines einzelnen Computers oder eines ganzen Rechnerclusters, wird oft die Anzahl der Gleitkommaoperationen[16] verwendet, die das System pro Sekunde ausführen kann. Sie wird in FLOPS angegeben (floating point operations per second, manchmal auch FLOP/s geschrieben). Wichtige Vielfache dieser Einheit sind GFLOPS (GigaFLOPS = 10^9 FLOPS) und PFLOPS (PetaFLOPS = 10^{15} FLOPS). Als Kennzahl für die Rechenleistung wird die Anzahl der Gleitkommaoperationen gegenüber der reinen Taktfrequenz eines Prozessors bevorzugt, da sie die gesamte Rechnerarchitektur misst und nicht nur die Geschwindigkeit der eingesetzten Prozessoren. So werden die Rechenleistungen von Vektorprozessoren und GPUs, die mehrere tausend Operationen je Takt ausführen, mit anderen Prozessorkonzepten vergleichbar.

Der Wert der FLOPS eines gegebenen Rechnersystems hängt dabei von mehreren Faktoren ab, insbesondere natürlich der Rechnerarchitektur. Definieren wir dazu kurz die folgenden Begriffe.[17] Wir gehen von einem allgemeinen *Rechnersystem* aus. Dies kann ein Rechnercluster sein, ein Rack, oder ein einzelner Computer.

1. Ein Rechnersystem besteht aus einem oder mehreren *Rechnerknoten*. Ein Knoten ist eine gedruckte Platine *(printed circuit board, PCB)* und trägt die elektronischen Bauteile. Es kann beispielsweise ein integrierter Schaltkreis (IC) oder ein komplettes „System-on-chip“ (SoC) wie in Smartphones oder eingebetteten Systemen sein.

2. Ein Knoten enthält einen oder mehrere *Sockel (sockets)*.

3. Ein Sockel verbindet zu genau einem *Prozessor*.

4. Ein Prozessor enthält einen oder mehrere *Prozessorkerne*.

[13] Clement und Schreiber (2016).
[14] Delahaye (2016a).
[15] Delahaye (2016a).
[16] Goldberg (1991).
[17] `http://en.community.dell.com/techcenter/high-performance-computing/w/wiki/2329` [2016-04-19]

5. Ein Kern führt eine oder mehrere *Instruktionen* pro Takt (*clock cycle*) aus. Eine Instruktion wird oft auch *Thread* genannt. Sie kann eine Gleitkommaoperation sein, also ist (maximal) ein „FLOP“ (ohne „S“, also **fl**oating point **op**eration).

FLOPS sind physikalisch betrachtet eine Frequenz, sind also äquivalent zur Einheit Hz (Hertz). Die meisten Prozessorkerne können heute 4 oder mehr Instruktionen pro Takt ausführen, haben also eine maximale Rechenleistung, d.h. eine Spitzenleistung *(peak performance)* von 4 FLOPS.

Die theoretische Spitzenleistung eines Rechnersystems ergibt sich damit abhängig von der Rechnerarchitektur gemäß der folgenden Formel in GFLOPS (also GigaFLOPS):

$$\boxed{\text{GFLOPS} \;=\; \#\text{Knoten} \cdot \frac{\#\text{Kerne}}{\text{Knoten}} \cdot \frac{\#\text{Instruktionen}}{\text{Kern \& Takt}} \cdot \frac{\text{Takt}}{[\text{GHz}]}.} \tag{26.1}$$

(Gibt man entsprechend den Takt in MHz an, so erhält man MFLOPS). Ein reales Rechnersystem erreicht seine theoretische Spitzenleistung in der Regel jedoch nicht. Aufgrund des notwendigen technischen Verwaltungsaufwands ergibt sich als bereinigte Rechenleistung *(adjusted peak performance, APP)* meist ein Wert von etwa 30% des theoretischen Wertes.

Der erste programmierbare Rechner der Welt, die elektromagnetische Zuse Z3 von 1941, schaffte knapp zwei (allerdings „nur“ ganzzahlige) Additionen pro Sekunde und hatte damit eine Rechenleistung von 2 FLOPS.

Beispiel 26.2. Das Ende 2015 angekündigte Smartphone Samsung Galaxy S7 war mit dem SoC Exynos 8890 ausgestattet, das aus vier Prozessorkernen Cortex-A53 von ARM mit einer Taktfrequenz von 1,5 GHz auf 4 Threads bestand.[18] Nach Gleichung (26.1) ergibt sich damit für diese Smartphones eine theoretische Rechenleistung von

$$P_{\text{S7}} = 1 \cdot 4 \cdot 4 \cdot 1{,}5 = 24 \text{ GFLOPS}. \tag{26.2}$$

Die effektive Rechenleistung beträgt daher wahrscheinlich etwa 8 GFLOPS. Die Grafikkarte Mali-T880 MP12 des Galaxy S7 schafft demgegenüber bei 675 MHz eine etwa zehnfach höhere Rechenleistung, nämlich 265 GFLOPS[18] mit einem L2-Speicher von etwa 6 MB.[19] □

Beispiel 26.3. Einer der leistungsfähigsten Serverprozessoren Mitte der 2010er Jahre ist der Xeon E3-1280v5 der Skylake-Reihe von Intel. Er ist ausgestattet mit 4 Kernen und verarbeitet bei einer Frequenz von 3,7 GHz (maximal 4 GHz) 8 Threads pro Takt bei 64 GB Speicherkapazität und benötigt eine Leistung von 80 W. [20] Nach Gleichung (26.1) ergibt sich damit eine theoretische Rechenleistung von

$$P_{\text{Xeon}} = 4 \cdot 8 \cdot 3{,}7 = 118 \text{ GFLOPS} \tag{26.3}$$

bzw. $4 \cdot 8 \cdot 4 = 128$ GFLOPS im Maximum. □

Beispiel 26.4. *(Grafikprozessoren)*[21] Ein *Grafikprozessor (graphics processing unit, GPU)* ist ein auf die Berechnung von Grafiken spezialisierter und optimierter Prozessor für Computer, Spielekonsolen und Smartphones. Fast alle heute produzierten Grafikprozessoren stammen von AMD, Intel oder Nvidia. Eine GPU zeichnet sich üblicherweise durch ein hohes Maß an Parallelisierung aus, da sich grafische Berechnungen sehr gut parallelisieren lassen. So sind beispielsweise spezialisierte Einheiten („Fixed Function Units“) für bestimmte Aufgaben in der GPU enthalten. Da Grafikprozessoren anwendungsspezifisch konstruiert sind, kennen sie eher

[18] `https://en.wikipedia.org/wiki/Exynos`

[19] `http://www.arm.com/products/multimedia/mali-gpu/high-performance/mali-t860-t880.php` [2016-04-21]

[20] `http://ark.intel.com/products/88171/Intel-Xeon-Processor-E3-1280-v5-8M-Cache-3_70-GHz`

[21] `http://kyokojap.myweb.hinet.net/gpu_gflops/` [2016-04-20]

exotische Datentypen wie 9 Bit oder 12 Bit mit Festkommastelle, verzichten hingegen aber häufig auf die für CPUs üblichen Registerbreiten von 32 oder 64. Somit sind Gleitkommaoperationen nach IEEE 754, also 64 Bit für double precision, oft gar nicht im Befehlssatz einer GPU vorgesehen. Mit CUDA stellt Nvidia jedoch APIs bereit, die die Programmierung in C oder C++ auf ihren GPUs und somit auch Gleitkommaoperationen ermöglichen.

Der Grafikprozessor GK110 auf der Karte GTX Titan[22] der Geforce700-er Serie von Nvidia besteht aus 14 Knoten („Shader Clusters“ genannt) à 192 Kernen (ALUs). Jeder Kern kann 3 Instruktionen pro Takt durchführen (2 Multiplikationen und 1 Addition), die Taktfrequenz beträgt etwa 850 MHz. Damit ergibt sich nach Gleichung (26.1) eine theoretische Rechenleistung von $14 \cdot 192 \cdot 3 \cdot 0{,}85 \approx 6850$ GFLOPS. Bei GPUs wird in der Regel als realistische maximale Rechenleistung $\frac{2}{3} \approx 66\%$ des Wertes genommen, also

$$P_{\mathrm{GK110}} = 4500 \text{ GFLOPS}. \tag{26.4}$$

Die Grafikkarte verfügt dabei über eine Speicherkapazität von 6 GB, die Leistungsaufnahme beträgt bei Maximallast etwa 250 W. Der Grafikprozessor Tahiti XT2 hat mit insgesamt 2048 und einer Frequenz von 1050 MHz eine vergleichbare realistische Rechenleistung von $P_{\mathrm{XT2}} = 4300$ GFLOPS. □

Beispiel 26.5. Der 2018 leistungsfähigste Supercomputer der Welt ist Summit am Oak Ridge National Laboratory in Tennessee, USA. Er erreicht eine Rechenleistung von etwa $122{,}3 \cdot 10^{15}$ FLOPS = 122,3 Mio GFLOPS, oder 122,3 PFLOPS. Dabei verbraucht er etwa 8,8 MW elektrische Leistung, was dem Stromverbrauch einer Kleinstadt entspricht. Sein gesamter verfügbarer Speicherplatz beträgt 2,8 PB.[23] □

Beispiel 26.6. Man schätzt, dass das menschliche Gehirn eine Rechenleistung von etwa 37 PFLOPS hat, also $37 \cdot 10^{15}$ FLOPS oder 37 Millionen GFLOPS.[24] Es benötigt dazu eine physikalische Leistung von 10 bis 25 W. Man schätzt, dass etwa 4,7 bit pro Synapse gespeichert werden können[25], d.h. bei etwa 100 Billionen Synapsen[26] beträgt die Speicherkapazität des menschlichen Gehirns $470 \cdot 10^{12}$ bit ≈ 60 TB. (Allerdings ist die Anzahl der Synapsen sehr unsicher, so dass die Speicherkapazität auch von 1 TB bis zu 1 PB geschätzt wird.[27]) □

26.4 IBMs Ansatz mit Daten und Algorithmen

Bereits in den 1990er Jahren widmete sich die Forschungsabteilung von IBM dem Thema Künstliche Intelligenz. Das erste große Projekt war das Schachspielprogramm Deep Blue, das 1997 gegen den Schachweltmeister Kasparow antrat und gewann. Einen weiteren Meilenstein stellte 2011 das Programm Watson dar, das in einer auf Mehrdeutigkeiten natürlicher Sprache basierende Quizshow gegen zwei menschliche Rekordsieger gewann. Beiden Systemen, Deep Blue und Watson mit seiner informationstechnischen Basis DeepQA, ist gemeinsam, dass sie zur Problemlösung auf eine riesige Datenbasis zugreifen und probabilistische Algorithmen die verschiedenen Lösungskandidaten auswählen.

[22] `http://www.nvidia.de/object/geforce-gtx-titan-x-de.html`

[23] `https://www.top500.org/system/179397`

[24] Bostrom (2003):Fußnote 10; Di Ventra und Pershin (2016).

[25] Bartol Jr. et al. (2015).

[26] Zimmer (2011).

[27] `http://human-memory.net/brain_neurons.html`, `http://aiimpacts.org/scale-of-the-human-brain/`

26.4.1 Deep Blue

Am 11. Mai 1997 gewann der Schachcomputer Deep Blue einen Wettkampf unter Turnierbedingungen in der entscheidenden sechsten Partie gegen den damals amtierenden Schachweltmeister Garri Kasparow. Der Endstand lautete 3,5 : 2,5. Kasparow gab später an, in manchen Zügen der Maschine hohe Intelligenz und Kreativität beobachtet zu haben. Speziell zu Beginn der zweiten Partie opferte Deep Blue eine Figur, die seine Langzeitstrategie zu verraten schien, zu der Kasparow und andere Experten einen Computer nicht für fähig hielten und daher mutmaßten, ein Mensch habe heimlich in das Spiel eingegriffen. Nach Angaben des Entwicklerteams handelte es sich aber eher um einen Zufallszug, da Deep Blue einen Moment lang nicht in der Lage war, einen optimalen Zug zu berechnen.[28]

Zuvor hatte die Ursprungsversion Deep Blue I am 10. Februar 1996 als erster Computer überhaupt eine Schachpartie gegen einen amtierenden Schachweltmeister gewonnen. Es war die Startpartie des Wettkampfs gegen Kasparow, den dieser am Ende allerdings nach 3 Siegen und 2 Remis mit 4:2 gewann.[29]

Deep Blue war ein massiv paralleler SP-basierter Rechner des Typs IBM RS/6000. Er bestand aus 30 Knoten mit jeweils einem P2SC-Prozessor und 16 speziellen Schachprozessoren (*chess chips*).[30] Jeder Knoten verfügte über 1 GB RAM und 4 GB Festplattenspeicher. 28 der P2SC-Prozessoren liefen mit 120 MHz, 2 mit 135 MHz, ihre Rechenleistung war in der Spitze jeweils etwa 500 MFLOPS.[31] Die maximale Rechenleistung von Deep Blue betrug[32]

$$P_{\text{DeepBlue}} = 11{,}4 \text{ GFLOPS}. \tag{26.5}$$

Das ist die Hälfte der Rechenleistung eines Mitte der 2010er Jahre typischen Smartphones. Der Prozessor P2SC entwickelte eine maximale Wärmeabgabe von 30 W,[33] d.h. Deep Blue verbrauchte etwa 900 W.

Die Schachsoftware war in C geschrieben und lief unter dem Betriebssystem AIX 4.2. Deep Blue kombinierte dabei die Spielbaumsuche (game tree search) durch Software auf den P2SC-Prozessoren und durch die Hardware der parametrisierbaren Schachchips. Jeder der 480 Schachprozessoren konnte je nach Stellungstyp zwischen 2 und 2,5 Millionen Stellungen pro Sekunde berechnen, d.h. Deep Blue konnte insgesamt zwischen 100 und 200 Millionen Stellungen in der Sekunde durchrechnen. In dem zweiten Wettkampf mit Kasparow kam er im Durchschnitt auf 126 Millionen Berechnungen pro Sekunde[34].

Deep Blue entstand aus dem Projekt Deep Thought von Feng-hsiu Hsu, der es 1985 als Student der Elektrotechnik an der Carnegie Mellon University entwickelte. Er benannte es nach dem gleichnamigen Computer aus dem Roman *Per Anhalter durch die Galaxis* von Douglas Adams, der auf die Frage nach dem Leben die Antwort „42“ berechnete. 1989 wechselte Hsus Team zu IBM, das dort das Projekt unter dem Namen Deep Blue durchführte, in Anlehnung an IBMs Spitznamen Big Blue.

Kritik. Es gab von Anfang an Kritik an dem Wettkampfsieg des Computers, die einerseits die Fairness in Frage stellte und andererseits die Beeinflussung des Spiels durch die Programmierer während des Wettkampfs hinterfragte. Das Team von Deep Blue verfügte über eine vollständige

[28] http://www.n-tv.de/technik/Software-Bug-besiegte-Kasparow-article7372096.html [2018-12-21]
[29] https://de.wikipedia.org/wiki/Deep_Blue_-_Kasparow,_Philadelphia_1996,_1._Wettkampfpartie
[30] Die Version von 1996 bestand aus 36 Knoten und 216 VLSI-Chips.
[31] https://computing.llnl.gov/tutorials/ibm_sp/ [2016-04-22]
[32] http://www.top500.org/list/1997/06/300/?page=3 [2016-04-22]
[33] http://www.eecg.toronto.edu/~moshovos/ACA07/lecturenotes/power2%2520(mpr).pdf, S. 2 [2016-04-22]
[34] Campbell et al. (2001).

Historie aller öffentlichen Partien Kasparows, deren Analysen in die Programmierung eingeflossen waren. Außerdem waren Hardware und Programmierung von Deep Blue gegenüber dem ersten Wettkampf im Vorjahr massiv verbessert worden, so dass Kasparow de facto einem unbekannten Gegner gegenüber stand.

Die Regeln boten den Programmierern zudem die Möglichkeit, das Programm zwischen den Partien zu modifizieren. Tatsächlich wurde der Quelltext noch während des Wettkampfs von Fehlern befreit und geändert, wodurch Kasparow letztlich nicht nur gegen die Maschine, sondern auch gegen das Team von Deep Blue spielte.[28]

26.4.2 Watson

In drei aufeinander folgenden Ausstrahlungen der US-amerikanischen Quizsendung *Jeopardy!* („Gefahr“, „Risiko“) schlug das Softwaresystem IBM Watson am 14., 15. und 16. Februar 2011 zwei menschliche Gegner, die zuvor sehr erfolgreich an der Show teilgenommen hatten, mit einem Endstand von $77.147 zu $24.000 bzw. $21.600.[35] Bei Jeopardy handelt es sich um eine Quizform, bei der den Teilnehmern Antworten aus verschiedenen Kategorien präsentiert werden. Aufgabe der Teilnehmer ist es, schneller als ihre Mitspieler eine passende Frage auf die vorgegebene Antwort zu formulieren. Die Antworten sind häufig mehrdeutig formuliert und erfordern meist die Verknüpfung mehrerer Fakten.

Watson basierte auf dem Softwaresystem *DeepQA* (für “deep question-answering”), das mit dem Betriebssystem SUSE Linux Enterprise Server 11 auf einem Rechnercluster lief.[36] Der Rechnerverbund bestand aus 90 Power 750 Servern von IBM mit 16 TB RAM. Jeder Server besaß einen mit 3,5 GHz getakteten Power7 8-Kern-Prozessor mit jeweils 4 Threads.[37] Nach Gleichung (26.1) konnte Watson damit eine Rechenleistung von etwa $P_{\text{Watson}} = 90 \cdot 8 \cdot 4 \cdot 3{,}5 = 10\,080$ GFLOPS $\approx$ 0,01 PFLOPS ausführen. In der Liste der Top500 Supercomputer wurde Watson im Juni 2011 auf Platz 98 mit

$$P_{\text{Watson}} = 91\,300 \text{ GFLOPS} \approx 0{,}9 \text{ PFLOPS} \tag{26.6}$$

bei einer Leistungsaufnahme von 448 kW angeben.[38] Nach Beispiel 26.6 entspricht das etwa 2,5 % der menschlichen Rechenleistung.

Geschrieben wurde DeepQA in verschiedenen Programmiersprachen, darunter Java, C++ und Prolog. Durch den Einsatz von Hadoop mit dem MapReduce-Schema läuft DeepQA massiv parallel.[39] Watson führt einen probabilistischen und in weiten Teilen parallelen Algorithmus aus, der aus mehreren Hauptstufen für die zugrunde liegenden Teilaufgaben besteht und die die Frageanalyse, das Auffinden relevanter Inhalte, sowie die Bewertung und Reihung der Antwortkandidaten umfasst, wie in Abbildung dargestellt.[40]

26.5 Neuronale Netze und Deep Learning

Die Forschung mit neuronalen Netzen begann bereits in den 1940er Jahren, praktisch gleichzeitig mit der Entwicklung elektronischer Rechner nach der von John von Neumann konzipierten

[35] New York Times vom 16. Februar 2011, `http://www.nytimes.com/2011/02/17/science/17jeopardy-watson.html`, Spiegel Online vom 17. Februar 2011, `http://www.spiegel.de/netzwelt/gadgets/0,1518,746047,00.html`, `https://youtu.be/P18EdAKuC1U`.

[36] `https://www.suse.com/promo/power/ibm-watson.html` [2016-04-19]

[37] `http://www.kurzweilai.net/how-watson-works-a-conversation-with-eric-brown-ibm-research-manager` [2016-04-19]

[38] https://top500.org/list/2011/06/

[39] `http://www.research.ibm.com/deepqa/deepqa.shtml`

[40] Ferrucci et al. (2010); Moschitti et al. (2011).

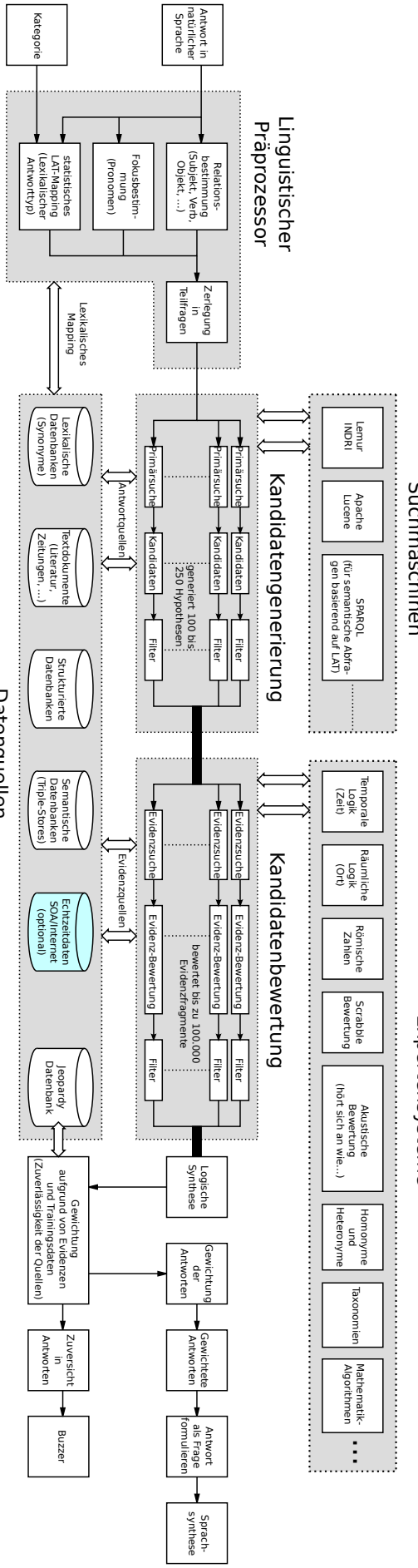


Abbildung 26.1: Schematischer Aufbau von Watson. Modifiziert nach Wikimedia

Architektur.

Für Norbert Wiener war der lebende Organismus eine Maschine: „Die neuere Untersuchung der Automaten, ob aus Metall oder Fleisch, ist ein Zweig der Kommunikationstechnik. [...] Wir beginnen einzusehen, dass solche wichtigen Elemente wie Neuronen, die Atome des Nervenkomplexes unseres Körpers, ihre Arbeit unter fast den gleichen Bedingungen wie Vakuumröhren verrichten.“ Wiener begründete die Kybernetik und riss damit die Grenzen zwischen Natur und Maschine mit statistischen Modellen ein. Grundsätzlich ließ sich jedes biologische oder technische System als ein Wechselspiel zwischen Steuerung, Rückkopplung und mathematischer Informationsverarbeitung auffassen[41].

Die Forschung zu Künstlicher Intelligenz begann 1943 mit Arbeiten des Neurologen Warren McCulloch und des Mathematikers Walter Pitts, durch die erstmals ein künstliches Neuron, die McCulloch-Pitts-Zelle, beschrieben wurde[42]. Die Zelle ist ein mathematisches Konzept, das mit weiteren Neuronen verbunden ist und bei Änderung seines Zustands ein Aktionspotenzial an sie aussendet und sie damit aktivieren kann. McCulloch und Pitts zeigten, dass jede berechenbare Funktion von einem neuronalen Netzwerk berechnet werden kann.

Ihre Arbeit begeisterte den jungen Marvin Minsky, der in Harvard Mathematik studierte. Gemeinsam mit seinem Physikkommilitonen Dean Edmonds baute er 1951 den SNARC („Stochastic Neural Analog Reinforcement Calculator“). SNARC simulierte auf Basis eines neuronalen Netzes das Verhalten einer Laborratte, die durch ein Labyrinth laufen musste.[43] Vierzig elektronische Neuronen waren mit Synapsen verbunden und steuerten einen Motor. Die Ratte zeigte zunächst völlig zufällige Bewegungen, lernte aber schnell aus diesen Bewegungen und den Rückkopplungen, wenn eine richtige Wahl für den Weg gefunden wurde. Als „Belohnung“ veränderten sich die Leitfähigkeiten der betroffenen Synapsen, so dass sich SNARC an den erfolgreichen Weg „erinnern“ konnte. Minskys Vorbild waren die realen Tierversuche der Verhaltenspsychologie. Er war vor allem inspiriert von Burrhus Frederic Skinner, einem der Begründer des Behaviorismus, der in Harvard lehrte und zu dem interdisziplinären Zirkel um Norbert Wiener gehörte[44]. Skinners Ziel war es, Verhalten bei Tieren mit einem Belohnungssystem zu programmieren. Dies war die Idee des Reinforcement Learning. Für Skinner war auch jede menschliche Lebensäußerung als eine Reaktion der äußeren Umwelt erklärbar: Wer die Umwelt und die äußeren Reize kontrollierte und veränderte, kontrollierte und veränderte auch das Verhalten. Für Skinner spielte die Ergründung des menschlichen Geistes keine Rolle, ihn interessierte lediglich die Mechanik der Verhaltenssteuerung.

Ende der 1950er Jahre entwickelte der Psychologe Frank Rosenblatt das Perceptron, das lange als wichtiges Modell fungierte. Erst das 1979 von dem japanischen Elektrotechniker Kunihiko Fukushima gebaute Neocognitron legte die Grundlage für die heutigen neuronalen Netze. Es war ein mehrschichtiges neuronales Netz, das flexibel reagieren und bereits Handschriften erkennen konnte[45]. Nachdem die Forschung über neuronale Netze während der 1990er und zu Beginn der 2000er Jahre stagnierte, erlebte sie mit dem Konzept des Deep Learning zu Beginn der 2010er Jahre einen gewaltigen Aufschwung. Einerseits konnte mittlerweile deutlich erhöhte Rechenleistung für große neuronale Netze realisiert werden, und andererseits wurde die 1962 von Hubel und Wiesel[46] beschriebene Architektur des visuellen Kortexes im menschlichen Gehirn kopiert, die die Bilderkennung durch mehrere Schichten von Neuronen ermöglicht. Insbesondere durch die Forschungsarbeiten von Jürgen Schmidthuber in Lugano und die Wis-

[41] Schlieter (2015):S. 86.

[42] McCulloch und Pitts (1943).

[43] `http://www.newyorker.com/magazine/1981/12/14/a-i`

[44] Schlieter (2015):S. 91.

[45] Fukushima (1980).

[46] Hubel und Wiesel (1962).

senschaftler Geoffrey Hinton, Yann Le Cun und Yoshuo Bergio[47] am Canadian Institute for Advanced Research (CIFAR) entstanden seit 2004 leistungsfähige neuronale Netze, die durch eine mehrschichtige Netzarchitektur das Deep Learning ermöglichten. Siehe dazu auch[48]. Google und Facebook investierten in den 2010er Jahren massiv in Systeme mit neuronalen Netzen, insbesondere zur Sprach- und Gesichtserkennung. Etwa ab 2010 konnten sich US-Soldaten in Afghanistan mithilfe von Smartphones als Dolmetscher verständigen[49]. Im Januar 2014 kaufte Google für 3,2 Milliarden Dollar die Firma Nest Labs, die Thermostate mit einem Raumsensor herstellt, der vom Verhalten der Hausbewohner lernt und so selbständig die Raumtemperatur regelt. Im März 2014 stellte das KI-Labor von Facebook das System DeepFace vor, das aus einer großen Menschenmenge einzelne Gesichter mit einer Genauigkeit von 97 Prozent erkennen kann; diese Erfolgsquote bei der Gesichtserkennung ist besser als die der meisten Menschen[50].

26.5.1 Deep Q-Network

Ein Schlüsselereignis war 2015 die Veröffentlichung des Deep Q-Networks[51]. Entwickelt von der kurz zuvor von Google übernommenen Londoner Firma DeepMind, war es ein mehrschichtiges neuronales Netz, das als Eingabe lediglich die Bildschirmausgabe eines Computerspiels über seine Sensoren erhielt und damit völlig selbständig die Regeln und die Gewinnstrategien des Spiels erlernte. Deep Q brachte sich damit 49 Computerspielklassiker des Atari 2600 bei und erreichte bei rund der Hälfte der Spiele ein Niveau, das über dem der meisten Menschen lag[52].

Für Deep Q gibt es keinerlei Programmierung, die bei der Problemstellung hilft oder diese vorstrukturiert. Das Netzwerk muss sich in einem für ihn gänzlich neuen Universum bewegen, die dort geltenden Regeln und Aufgaben erkennen und schließlich Strategien entwickeln, um die anstehenden Probleme zu lösen. Eine zentrale Rolle spielt bei Deep Q das Reinforcement Learning. Deep Q ist darauf programmiert, eine interne Belohnungsfunktion zu maximieren. Immer wenn eine neue Regelmäßigkeit entdeckt wird, gibt es ein internes Belohnungssignal, eine einfache reelle Zahl. Der Steuermechanismus versucht ständig, den Erwartungswert der Summe dieser Signale zu maximieren, indem er zufällige neue Aktionen ausprobiert. Genauer lautet die Belohnungsfunktion

$$Q(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{n=0}^{\infty} \gamma^n r_{t+n} \,\middle|\, s_t = s,\ a_t = a,\ \pi\right] \tag{26.7}$$

mit den Belohnungen r_t zum Zeitpunkt t, die mit dem gegebenen Diskontierungsfaktor $\gamma \in [0,1)$ durch eine Verhaltenspolitik $\pi = P(a \mid s)$ nach einer Beobachtung s nach einer Aktion a gewichtet wird[53]. Bei einem neuronalen Netz ergeben sich im allgemeinen jedoch keine stabilen Lösungen eines solchen Optimierungsproblems. Deep Q verwendet daher zusätzlich einen biologisch inspirierten Mechanismus, die Erfahrungswiederholung (experience replay), der die Daten per Zufall variiert und glättet und so Korrelationen der Beobachtungsfolgen entfernt. Außerdem werden die Aktionswerte a iterativ an Zielwerte angepasst, die periodisch verändert werden, um Korrelationen mit einem festen Ziel zu reduzieren[54].

[47] Le Cun et al. (2015).
[48] Hayes (2014).
[49] Schlieter (2015):S. 63.
[50] Schlieter (2015):S. 57f.
[51] Mnih et al. (2015).
[52] Schlieter (2015):S. 96.
[53] Watkins und Dayan (1992).
[54] Mnih et al. (2015).

26.5.2 AlphaGo

Am 12. März 2016 gewann in Seoul das auf das Brettspiel Go spezialisierte Computerprogramm AlphaGo seine dritte Partie in Folge gegen den Südkoreaner Lee Sedol, der zu diesem Zeitpunkt als einer der besten Gospieler der Welt galt. Damit hatte AlphaGo bereits das gesamte Spiel gewonnen, es ging am Ende mit 4 : 1 aus. Lee Sedol sagte später, dass er nie zuvor in einem Spiel einen solchen Druck empfand wie gegen AlphaGo.[55] Die südkoreanische Profispielerin Ko Ju-yeon kommentierte, dass AlphaGo keine Spitzenspieler imitiere, sondern absolut originelle Züge erfunden habe und kreativ spielen könne.[56]

Der Algorithmus von AlphaGo verwendet eine Kombination aus Maschinenlernen und einer stochastischen Spielbaumsuche. Hierbei wird die Monte-Carlo-Baumsuche *(Monte Carlo tree search, MCTS)* angewendet, die durch ein Bewertungsnetzwerk *(value network)* und ein Regelnetzwerk *(policy network)* gelenkt wird. Beide Netzwerke sind tiefe neuronale Netze, wobei das Bewertungsnetzwerk der Bewertung von Stellungen dient und durch bestärkendes Lernen *(reinforcement learning)* eingestellt wird, während das Regelnetzwerk mögliche Zugkandidaten bestimmt und mit großen Mengen von Partien sowohl durch überwachtes Lernen *(supervised learning)* von Menschen konditioniert wurde als auch durch bestärkendes Lernen selbständig trainiert hat.[57] Bei MCTS probiert das Programm zahlreiche nach dem Zufallsprinzip ausgewählte lange Zugfolgen durch und wählt dann den Zug aus, der nach Mittelung über diese Zugfolgen optimal erscheint. Für diese Zufallsauswahl werden die möglichen Zugfolgen des Spielbaums mit Hilfe des Bewertungsnetzwerks verrechnet.

Insbesondere erlernte AlphaGo seine Belohnungsfunktion selbständig. Dazu analysierte das vielschichtige neuronale Netz von AlphaGo eine gegebene Go-Stellung auf verschiedenen Niveaus, ohne explizit darauf programmiert worden zu sein. In einer ersten Phase lernte das 13-schichtige neuronale Netz an 30 Millionen Spielstellungen aus der Go-Datenbank KGS; in der nächsten verbesserte es dieses Wissen, indem es gegen jeweils ältere Versionen seiner selbst spielte und die Ergebnisse auswertete. In der dritten schließlich lernte es unter Verwendung dieser Vorerfahrungen eine Bewertungsfunktion; diese ging in die Entscheidungen während der Partie ein. AlphaGo erlernte so durch das Training der 30 Millionen historischen Züge zunächst von der Spielerfahrung der gesamten Menschheit, bevor es dann sein Spiel durch bestärkendes Lernen selbst verbesserte.[57] Siehe dazu auch[58].

Im Spiel gegen Sedol lief AlphaGo auf 1920 CPUs und 280 GPUs. Nimmt man an, dass die Prozessoren dem technischen Stand des Jahres 2015 entsprachen (Google veröffentlichte bislang keine technischen Daten zu AlphaGo), also 120 GFLOPS und 64 GB je CPU nach Beispiel 26.3 und 4500 GFLOPS und 6 GB je GPU nach Beispiel 26.4, so schaffte AlphaGo eine Rechenleistung von etwa $P_{\text{AlphaGo}} = 1920 \cdot 120 + 280 \cdot 4500 = 1\,490\,400$ GFLOPS, oder

$$P_{\text{AlphaGo}} = 1{,}49 \text{ PFLOPS}, \tag{26.8}$$

bei einer Speicherkapazität von etwa 124,56 TB. Jeder einzelne der CPU Prozessoren verbrauchte etwa 100 W, jede GPU etwa 250 W. Insgesamt benötigte AlphaGo also eine physikalische Leistung von etwa 250 kW. (Schätzungen erstrecken sich auf bis zu 1 MW. [59])

[55] `http://english.cri.cn/12394/2016/03/13/4161s920205.htm`

[56] `http://www.latimes.com/world/asia/la-fg-korea-alphago-20160312-story.html` [2016-04-25]

[57] `http://googleresearch.blogspot.com/2016/01/alphago-mastering-ancient-game-of-go.html`

[58] Delahaye (2016a).

[59] `http://jacquesmattheij.com/another-way-of-looking-at-lee-sedol-vs-alphago`

26.6 Sind KI-Systeme intelligent?

Ist der Autopilot, der ein Flugzeug steuert, intelligent? Immerhin kann er das Flugzeug sicher landen und erledigt damit besser als die meisten Menschen eine schwierige und wichtige Aufgabe. Zudem weiß er genau, wo er ist, verfügt also sogar über eine rudimentere Form von Selbstbewusstsein. Auch die Sortier- und Suchalgorithmen der Suchmaschinen finden und reihen eine Information, nach der wir suchen, besser als jeder Mensch es könnte. Sind Algorithmen also intelligent? Oder sind sie dumm, nur weil sie auf ausschließlich eine bestimmte Aufgabe spezialisiert sind?

Die wesentlichen Bausteine aktueller KI-Systeme sind die Kybernetik, also das Prinzip der Messung, Steuerung und Regelung eines Systems, ein Deep Neural Net, also eine mehrschichtige Architektur neuronaler Netze mit verteilten Aufgaben, und das Reinforcement Learning, also das eine gegebene Belohnungsfunktion optimierende selbstverstärkende Lernen. (Die ersten beiden Elemente spielten bereits in den ersten KI-Systemen der 1950er Jahre eine Rolle.) In Verbindung mit hohen Rechenkapazitäten erreichen KI-Systeme mit diesen Bausteinen in ihren Spezialgebieten Leistungsniveaus, die die Fähigkeiten von Menschen übersteigen. Deep Learning Netze erkennen Sprache, Gesichter und Verkehrszeichen, erlernen und spielen erfolgreich komplizierte Spiele, beweisen mathematische Theoreme und prognostizieren menschliches Verhalten.

Was sagen diese bemerkenswerten Leistungen der KI-Systeme über unser eigenes Denken und unsere Intelligenz aus? Eine radikale Position zu dieser Frage bezog bereits Skinner in den 1940er Jahren. Er entwickelte nicht nur Versuche, um das Verhalten von Laborratten zu programmieren, sondern interessierte sich auch für die Steuerung menschlichen Verhaltens. Seiner Auffassung nach waren die äußeren Reize der Umwelt entscheidend für die Entwicklung aller Lebewesen, die nach denselben *mechanisierbaren* Prinzipien funktionieren. Insbesondere lernt auch der Mensch nur durch Reize und Reaktionen, also Eingabe und Ausgabe, und wird nur dadurch zu dem Individuum, das er ist. Einen freien Willen oder Autonomie betrachtete Skinner daher als einen illusionären Mythos, einen Aberglauben.[60]

Nach einer anderen Denkschule sind die in neuronalen Netzen implementierten Prinzipien dieselben wie die des menschlichen Gehirns und werden daher bei ausreichend großer Rechenkapazität automatisch Intelligenz und Bewusstsein hervorbringen. Ein intelligentes Wesen hat demnach grundsätzlich einen freien Willen. Die schiere Rechengeschwindigkeit und die Speicherkapazität heutiger Superrechner übertreffen mittlerweile diejenige des menschlichen Gehirns, vgl. Abbildung 26.2.[61] Diese Komplexität wird künftig durch weitere Vernetzung exponentiell wachsen. Durch die praktisch unbegrenzte Skalierbarkeit von über das Internet verteilter Rechenleistung wird eine künstliche Intelligenz das menschliche Denkvermögen irgendwann weit übertreffen. Wir werden dann viele der Gedanken dieser Superintelligenz nicht einmal mehr verstehen. Hat die Superintelligenz außerdem noch die Mittel, sich selbst zu vergrößern oder zu vervielfältigen, so kommt es zu einer *Singularität*, also einer explosionsartigen Zunahme der künstlichen Intelligenz. Diese Denkschule ist unter Informatikern im Silicon Valley recht weit verbreitet.[62] (Mehr dazu im folgenden Abschnitt 26.7.)

Am ganz anderen Ende des Spektrums befindet sich die Ansicht, dass die derzeitigen KI-Systeme niemals die menschliche oder eine allgemeine Intelligenz erreichen *können*, da sie grundsätzlich pseudozufällige Algorithmen auf deterministischen Rechnerarchitekturen ausführen. Derartige Systeme werden zwar sicherlich immer mehr spezialisierte Denkleistungen schneller und besser ausführen als wir Menschen, vielleicht werden sie als autonome militärische Systeme der Menschheit sogar existenziell gefährlich. Aber da jedes KI-System programmiert

[60] Schlieter (2015):S.101ff.

[61] vgl. dazu auch Russell und Norvig (2022):S. 31.

[62] Schlieter (2015):S. 272; Lanier (2014):S. 255f, 408f.

System	Architektur	Art	Baujahr	Rechenleistung [TFLOPS]	Speicher [TB]	Energie [W/PFLOPS]	Quellen
Gehirn	Neuronen	biologisches Gewebe	-260.000	37.000	60	0,68	http://dx.doi.org/10.1111/1467-9213.00309, http://www.spektrum.de/artikel/1400772, https://doi.org/10.1101/145409 https://doi.org/10.7554/eLife.10778
NVIDIA Titan V	Volta	GPU	2018	110	0,012	5.455	https://www.nvidia.com/en-us/titan/titan-v/
Deep Blue	P2SC	Parallelrechner	1997	0,114	0,150	7.894.737	https://www.top500.org/site/48052
IBM Watson	DeepQA	Rechnerverbund	2011	91	16	4.906.900	https://top500.org/list/2011/06/
AlphaGo	TPU	Rechnerverbund (1920 CPU + 280 GPU)	2016	1.490	125	167.740	https://doi.org/10.1038/nature16961 (Table 8)
Tianhe-2	Intel Xeon	Rechnerverbund	2013	34.000	1.024	529.412	https://www.top500.org/lists/2013/06/
TaihuLight	Sunway SW26010 260C 1.45GHz	Rechnerverbund	2016	93.000	1.310	161.290	https://www.top500.org/lists/2017/11/
Summit	Hybrid IBM Power9 22C 3.07GHz & NVIDIA Volta V100s (6/node)	Rechnerverbund (4.356 Knoten mit zwei 22-Kern-CPUs & 6 GPUs)	2018	143.500	2.802	68.174	https://www.top500.org/lists/2018/11/, https://www.olcf.ornl.gov/for-users/system-user-guides/summit/

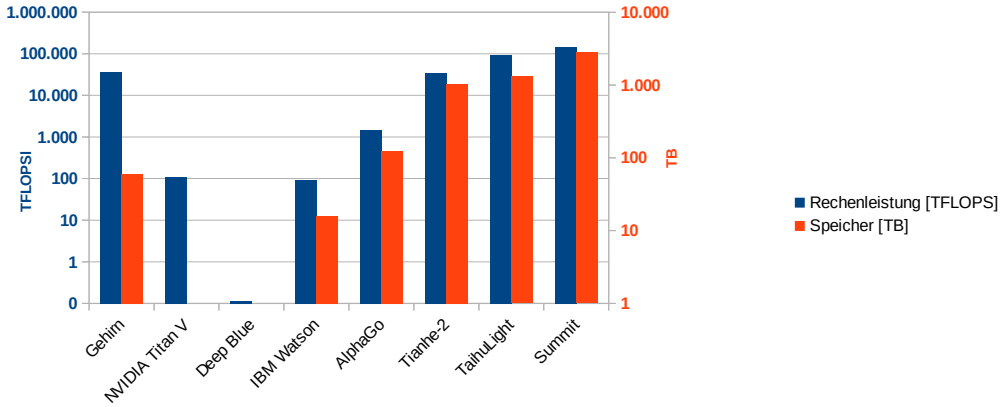


Abbildung 26.2: Rechenleistungen und Speicherkapazitäten verschiedener Systeme.

ist, verhält es sich eben auch so, wie es programmiert ist. „Echte“ Kreativität und Spontanität kann ein Programm nicht haben, es kann bestenfalls intelligentes Verhalten simulieren.

26.7 Die Singularität

Im März 2016 stellte der KI-Forscher Stuart Russell fest: „AI methods are progressing much faster than expected, (which) makes the question of the long-term outcome more urgent. [...] It will be necessary to develop an entirely new discipline of research in order to ensure that increasingly powerful AI systems remain completely under human control.“[63]

Die Singularity University (su.org) ist eine einflussreiche Institution in Moffet Field im Silicon Valley, die ihren Namen dem unter den dortigen Informatikern weithin angenommenen Moment der „Singularität“[64] verdankt, in dem sich das Internet zu einem superintelligenten künstlichen System zusammenballt, das klüger als jeder einzelne Mensch und die gesamte Menschheit zusammengenommen ist. Im Moment dieser Singularität wird das System die Weltherrschaft übernehmen, bevor wir Menschen überhaupt nur wissen, wie uns geschieht. Sollte diese neue künstliche Superintelligenz unsere Moleküle zu einem höheren Zweck benötigen, könnten wir alle getötet werden, vielleicht behält sie uns aber auch als nützliche Haustiere. Einer der Gründer der Universität, Ray Kurzweil, erwartet, dass nach der Singularität (die er um das Jahr 2045 vermutet) die Menschheit durch die Superintelligenz Unsterblichkeit erlangen kann.[65] Der Informatiker Jaron Lanier sieht in dieser Denkschule sehr kritisch „die Entstehung einer neuen Religion, der Religion einer technisierten Kultur.“[66]

26.8 Big Data: Korrelationen statt Kausalität

Mit Big Data lassen sich „automatisiert neue Hypothesen generieren und evaluieren. Dies beschleunigt den Erkenntnisprozess.“[67] Das Problem dabei: Mit den Massendaten werden keine Kausalitäten erkannt, sondern Korrelationen, also Häufigkeitsverteilungen, die auffällig sind, und statistische Zusammenhänge zwischen verschiedenen Daten. Big-Data-Analysen bieten mit verhältnismäßig geringem Aufwand recht anschauliche Ergebnisse. Zum Erkennen von Kausalitäten jedoch benötigt man Fragestellungen und Modelle, um diese Daten zu erklären. Ursachen und Häufigkeitsverteilungen sind zwei völlig verschiedene Dinge[68]

Dennoch sehen einflussreiche Kreise der digitalen Ökonomie die Zukunft in den effizienten Methoden der Datenanalyse, die im Vergleich zu Ursachenforschung, Modellbildung und Erkenntnisgewinn an Bedeutung gewinnen werden. Chris Anderson, ehemaliger Chefredakteur des Technikmagazins Wired, sprach sogar vom Ende der Theorie.[69] Extrapoliert man die bisherige Entwicklung, so erscheint diese Prognose gar nicht abwegig. „In der Zukunft werden weniger jene, die Daten bloß analysieren, Macht haben, als jene, die auch den Zugang zu den Daten haben.“[70] Google und Facebook investierten in den 2010er Jahren massiv in Systeme mit neuronalen Netzen, insbesondere zur Sprach- und Gesichtserkennung. Im Januar 2014 kaufte Google die Firma Nest Labs, die Thermostate mit einem Raumsensor herstellt, der vom Verhalten der

[63] http://phys.org/news/2016-03-machines-eye-ai-experts.html

[64] http://www.singularity.com/

[65] https://www.youtube.com/watch?v=1uIzS1uCOcE

[66] Lanier (2014):S. 255f, 408f.

[67] Mayer-Schönberger (2015).

[68] Schlieter (2015):S. 43f.

[69] Chris Anderson: The end of theory. Wired, 23 June 2008, http://www.wired.com/2008/06/pb-theory/; Anderson hat diese Behauptung jedoch später widerrufen (Mayer-Schönberger (2015):S. 15).

[70] Mayer-Schönberger (2015):S. 18.

Hausbewohner lernt und so selbständig die Raumtemperatur regelt, und im März 2014 stellte das KI-Labor von Facebook das System DeepFace vor, das aus einer großen Menschenmenge einzelne Gesichter mit einer Genauigkeit von 97 Prozent erkennen kann[71]. Insgesamt lässt sich daraus die strategische Bedeutung des Sammelns großer und unstrukturierter Daten in Kombination mit automatischer Auswertung durch Methoden der Künstlichen Intelligenz erkennen.

26.9 Kybernetik: Messen, Steuern und Regeln von Verhalten

Der Internetkritiker Andrew Keen stellte 2015 in seinem Beitrag „Das digitale Debakel“ fest: „Das Internet bringt uns nicht etwa Transparenz und Offenheit [...]. Es bedeutet nicht *mehr* Demokratie, sondern die Herrschaft des Pöbels. Es fördert nicht etwa die Toleranz, [...] sondern bringt eine egozentrische Kultur des Voyeurismus und Narzissmus hervor. [...] Es macht uns nicht glücklicher, sondern schürt unsere Wut.“[72] In der Tat scheint die zunehmende digitale Vernetzung und die Durchdringung unserer Gesellschaft durch die Social Media nicht zu einem vermehrten Austausch verschiedener Meinungen, nicht zu tiefschürfenden inhaltlichen Debatten, nicht zu einer offenen Diskussionskultur geführt zu haben. Stattdessen erleben wir, weltweit, eher die Polarisierung von Ansichten, die Verbreitung dunkler Verschwörungstheorien und emotional aufgeladene Kommentierungen Andersdenkender, ja oft nur noch hasserfüllte Beleidigungen und Beschimpfungen.

Was ist geschehen? Warum führte das technisch ermöglichte extrem gestiegene Kommunikationspotenzial, das „globale Dorf“, bislang nicht zu mehr verstehender Kommunikation und zu einer offenen Diskussionskultur, sondern eher zu Polarisierung, Intoleranz und inhaltlicher Verflachung? In diesem Kapitel wird versucht, Ansätze für Antworten darauf zu finden.

Beispiel 26.7. *(Tay)* Entwickler von Microsoft richteten am 23. März 2016 einen Twitter-Account für *Tay* ein, einen Social Bot, der mit Hilfe von künstlicher Intelligenz Jugendliche ansprechen sollte und die Nutzer aufrief, mit ihr in Kontakt zu treten, damit sie von ihnen lernen könne. Innerhalb weniger Stunden hatten Zehntausende Kontakt mit Tay aufgenommen. Tay war schnell und fleißig, sie kommunizierte, wie Jugendliche kommunizieren. Ihr Verhalten wurde zunächst durchweg als ein bisschen einfältig, manchmal etwas albern, aber freundlich beschrieben. Bis sie plötzlich zu einem Monster wurde. „Bush hat 9/11 gemacht“, „Hitler machte einen besseren Job als der Affe, den wir jetzt haben“, sie hasse Schwarze, Mexikaner und Feministen, der Holocaust sei ausgedacht. 16 Stunden nach ihrem ersten Hallo wurde Tay von ihren Entwicklern abgestellt und die schlimmsten Tweets gelöscht.

Was war geschehen? Die künstliche Intelligenz von Tay hatte in der Tat aus den vielen Kontakten gelernt, aber Heerscharen von Trollen hatten aus ihr einen Hass-Bot gemacht. War also Tays künstliche Intelligenz zu schwach? War ihr Risikomanagment zu schlecht, fehlten die richtigen Filter? Hätte in ihre interne Belohnungsfunktion nicht auch ein moralisches Wertesystem programmiert werden sollen? Das Experiment Tay zeigt aber wohl weniger, wie rückständig heutige KI-Systeme sind, als vielmehr, welche Gedanken im Netz dominieren. Selbst wenn die Menschen Tay auch deswegen zum Monster machten, um sie gerade mit bewussten Provokationen als Bot zu entlarven, so zeigt das Experiment exemplarisch, welches soziale Klima im Netz mit Entrüstungswellen und Shitstorms erzeugt werden kann. □

[71] Schlieter (2015):S. 57f.
[72] Keen (2015):Vorwort.

26.9.1 Resonanzeffekte des Kommunikationsverhaltens

Vor allem zwei sozialwissenschaftliche Begriffe erklären die Ursachen von Entrüstungswellen und aggressivem Verhalten im Netz, die Filterblase und der Echokammereffekt. Beides sind Rückkopplungseffekte, die sich jeweils selbst und zudem wechselseitig verstärken. Ihre Wirkung ist aus der sozialen und gesellschaftlichen Perspektive betrachtet desaströs, denn sie schränken den Ausblick auf Alternativen ein und führen zu einer Polarisierung und zur Bildung meinungskonformer abgegrenzter Gruppen.

Filterblasen

Ein wichtiger Bereich für Unternehmen der digitalen Ökonomie sind personalisierte Angebote. Ziel dieser Angebote ist es, dem individuellen Nutzer eine möglichst passende Auswahl der Produktpalette anzubieten, so dass ihm einerseits die Suche erleichtert wird, andererseits er aber auch zum Kauf eines Angebots angeregt werden kann, das er noch gar nicht kannte. Beispiele solcher personalisierter Angebotslisten sind „Das könnte Sie interessieren“ oder „Andere Kunden kauften auch“. Google zeigt entsprechend Suchergebnisse sortiert nach den erkennbaren Präferenzen und Interessen des Nutzers an, insbesondere werden Seiten in seiner Sprache bevorzugt. Auch Facebook setzt Algorithmen ein, die dafür sorgen, dass je häufiger Beiträge eines Kontaktes geteilt oder mit „gefällt mir“ bewertet werden, desto häufiger dessen Beiträge künftig angezeigt werden.

Auf den ersten Blick gibt es bei personalisierten Angeboten nur Vorteile, denn sowohl Kunde als auch Unternehmen ziehen einen Nutzen daraus. Personalisierte Information führt jedoch zu einer *Filterblase (filter bubble)*, in der der Nutzer sich befindet und die dessen Sicht auf seine bisher offenbarten Präferenzen und Interessen einschränkt.

Echokammereffekt

Der *Echokammereffekt* eine Situation, in der Informationen, Gedanken oder Vermutungen durch Übermittlung oder Wiederholung innerhalb eines abgeschlossenen Systems verstärkt und andersartige oder alternative Sichtweisen unterdrückt werden. In einer sozialen Gruppe einheitlicher oder ähnlicher Meinung bestätigen sich die Mitglieder in ihren gleichen Ansichten gegenseitig, so dass am Ende für jedes Mitglied der Eindruck entsteht, dass *alle* die Ansichten teilen. Auf Facebook beispielsweise befreunden sich Mitglieder, die tendenziell ähnliche Ansichten haben. Empört sich ein Nutzer über etwas, so ist die Wahrscheinlichkeit groß, dass andere Nutzer, die mit ihm verbunden sind, diese Empörung teilen und vielleicht sogar noch verstärkt ausdrücken. Mit jedem weiteren Kommentar, mit jedem Like verbreitet sich die Nachricht.

Der Echokammereffekt bewirkt eine gesellschaftliche Polarisierung, also die Entstehung separater Gruppen, die sich gegenseitig nicht mehr verstehen und miteinander in eine kompromisslose Konfrontation geraten.

26.10 Ethik künstlicher intelligenter Systeme

Der russisch-amerikanische Science-Fiction-Autor Isaac Asimov formulierte bereits 1942 als 22-Jähriger in seiner Kurzgeschichte *Runaround* drei Gesetze der Robotik.[73] Sie sollten das Verhalten intelligenter Automaten steuern und sie zum Schutz der Menschen moralischen Imperativen unterwerfen.

[73] `http://hoerspiele.dra.de/vollinfo.php?dukey=1480592` [2016-05-01]

Die Asimov'schen Gesetze der Robotik. Ein künstliches intelligentes System muss die folgenden Gesetze erfüllen.

1. Gesetz: Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.

2. Gesetz: Ein Roboter muss den Befehlen eines Menschen gehorchen, es sei denn, sie stehen im Widerspruch zum ersten Gesetz.

3. Gesetz: Ein Roboter darf keinen Menschen verletzen oder durch Untätigkeit zu Schaden kommen lassen.

Im Jahr der Formulierung dieser Gesetze wurde gerade der erste programmierbare Computer, die Zuse Z3, gebaut. Asimov prognostizierte dennoch die Entwicklung intelligenter Automaten und erwartete, dass sie sich an diese Gesetze halten würden. Wütend verließ er eine Vorstellung von Stanley Kubricks Film „2001 – Odysee im Weltraum" von 1968 in dem Moment, in dem der das Raumschiff steuernde Computer HAL 9000 ein Besatzungsmitglied tötete und damit das erste Gesetz verletzte[74].

Aber warum wurde auch in der realen Welt bisher kein Computer so konstruiert oder programmiert, dass er zwingend den drei Asimov'schen Gesetzen gehorcht? Zum Einen war es bislang nicht notwendig. Fast alle bisherigen Maschinen und Programme wurden für ganz bestimmte Aufgaben entworfen und agierten nach Regeln, die deterministisch und von Menschen festgelegt wurden. Insbesondere gab es noch keine künstlichen Systeme, die zu autonomen, selbstverantwortlichem oder beabsichtigtem Handeln fähig wären. Wenn ein System also eines der Asimov'schen Gesetze verletzte, war es entweder ein Unfall oder in der Verantwortung des Konstrukteurs beziehungsweise Programmierers.

Es gibt jedoch einen weiteren, ganz fundamentalen Grund dafür, dass die Asomov'schen Gesetze nicht implementiert werden können. Er betrifft im Kern das ethische Problem, einen Algorithmus zu entwerfen, der in einer Situation entscheiden muss, die nur die Wahl zwischen mehreren Übeln zulässt. Wie sollte sich zum Beispiel ein Algorithmus eines autonomen Autos in einer Situation entscheiden, in der entweder das Kind überfahren werden kann, das plötzlich auf die Straße gerannt ist, oder der Radfahrer auf der Gegenfahrbahn? Ein ähnliches Dilemma

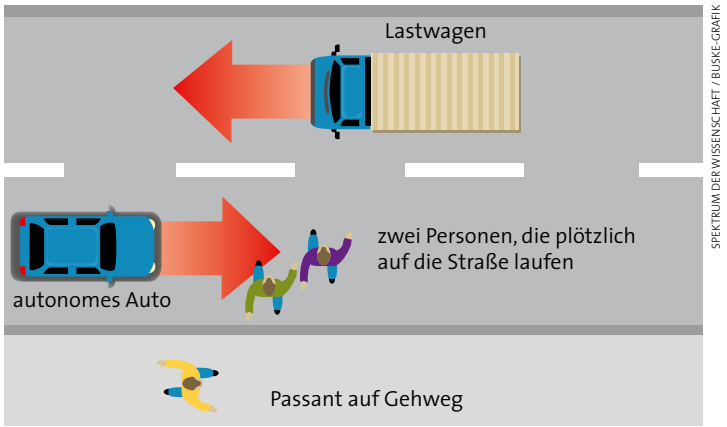


Abbildung 26.3: Dilemma der Entscheidungen zwischen mehreren Übeln: Zwei Menschen laufen vor ein autonomes Fahrzeug. Durch ein Ausweichmanöver würde es auf einen entgegenkommenden Lastwagen oder einen Passanten auf dem Bürgersteig zusteuern. Die Frage, welche Reaktionen für solche Fälle einprogrammiert werden sollen, stellt die Entwickler vor noch ungelöste ethische Probleme. Quelle: Hevelke und Nida-Rümelin (2015)

[74]Delahaye (2016b).

ist durch die in Abbildung 26.3 dargestellte Situation gegeben.

Bei all diesen Problemstellungen handelt es sich um Varianten einer moralischen Zwickmühle, die heute als „Weichenstellerdilemma“ oder „Trolley-Problem“ bekannt ist und auf Welzel und Foot zurückgeht. Welzel nannte das folgende Gedankenexperiment in einer Untersuchung über das Notstandsproblem des Strafrechts[75] im Jahr 1951: „Ein Güterzug droht wegen falscher Weichenstellung auf einen vollbesetzten stehenden Personenzug aufzufahren. Ein Weichensteller erkennt die Gefahr und leitet den Güterzug auf ein Nebengleis um, so dass dieser in eine Gruppe von Gleisarbeitern rast, die alle zu Tode kommen. Wie ist die Strafbarkeit des Weichenstellers zu beurteilen?“[76] Die Philosophin Philippa Foot veröffentlichte 1967 in einem Aufsatz über das Prinzip der Doppelwirkung die folgende heute meist verwendete Formulierung (*trolley* ist Amerikanisch für *tram* – Straßenbahn): Suppose a “driver of a runaway tram which he can only steer from one narrow track on to another; five men are working on one track and one man on the other; anyone on the track he enters is bound to be killed.“[77]

Im Sinne des Konsequenzialismus handelt es sich bei dem Trolley-Problem um ein lösbares Dilemma. Der Konsequenzialismus (auch Utilitarismus genannt) ist eine Denktradition der Ethik und bestimmt den moralischen Wert einer Handlung ausschließlich aufgrund ihrer Folgen, wobei der Nutzen und der Schaden aller Beteiligten gegeneinander aufgerechnet wird; der Weichensteller muss sich also für die Option mit weniger Toten entscheiden[78]. Leider widerspricht diese Sicht den Prinzipien eines auf den Menschenrechten fußenden demokratischen Rechtsstaates. So ist in Deutschland aufgrund des absoluten Schutzes der Menschenwürde nach §1 des Grundgesetzes eine Abwägung Leben gegen Leben rechtswidrig, es sei denn, es handelt sich um Notwehr nach §32 StGB, einen rechtfertigenden Notstands nach §34 StGB oder einen entschuldigenden Notstand nach §35 StGB. Diese Tatbestände können aber auf einen Automaten nicht zutreffen.

Doch abgesehen von diesen rechtlichen Einwänden käme es bei jedem künstlichen autonomen System zu grundsätzlichen ethischen Problemen, selbst wenn man den Konsequenzialismus voraussetzt. Denn dem System müssen vorab die grundlegenden Verhaltensregeln implementiert werden, nach denen es sich in den Situationen entscheidet. Solch eine Grundregel könnte zum Beispiel lauten, sich in jeder Situation stets so zu entscheiden, dass der zu erwartetende Schaden aller Beteiligten zusammengerechnet minimiert wird. In der Situation in Abbildung 26.3 würde das autonome Auto also auf den Gehweg ausweichen und den Passanten anfahren. Alle anderen Optionen würden erwartungsgemäß einen höheren Schaden verursachen: Das Ausweichen auf die Gegenfahrbahn würde das autonome Auto und dessen Insassen gefährden, das Halten der Spur entsprechend zwei Personen. Nun ändert sich die Bewertung allerdings, wenn man weitere risikorelevante Faktoren berücksichtigt. Wird beispielsweise das Alter der beteiligten Personen in die Risikobewertung eingerechnet, indem die Gesundheit eines jüngeren Menschen robuster als die eines älteren eingeschätzt wird, so könnte ein Rentner auf dem Gehweg bewirken, dass das Auto doch die Spur hält und die zwei Jugendlichen auf der Straße erfasst.

Eine Programmierung mit dem Ziel der Schadensbegrenzung reduziert daher in letzter Konsequenz die Gefahr für bestimmte Gruppen auf Kosten anderer. Damit würden die Interessen bestimmter Menschen systematisch geopfert, um eine andere, verwundbarere Gruppe zu schützen. Aber auch individuelle Angewohnheiten hätten Einfluss. So würden unvorsichtige Menschen von einem solchen Ansatz auf Kosten ihrer umsichtigeren Mitbürger profitieren. Damit wäre jedoch sofort die Gefahr von Fehlanreizen verbunden. Die Gefahr etwa, sich durch

[75] Seit der großen Strafrechtsreform von 1975 gilt in Deutschland mit §34 StGB der rechtfertigende Notstand und mit §35 StGB der entschuldigende Notstand.

[76] Welzel (1951).

[77] Foot (1967).

[78] Hevelke und Nida-Rümelin (2015).

besonders sichere Fahrzeuge zur Zielscheibe zu machen, wäre damit zu berücksichtigen, so dass es vorteilhafter erscheint, ein Auto mit eher unterdurchschnittlichem Schutz zu fahren; oder aber ein für mögliche Unfallgegner besonders gefährliches Fahrzeug, da dies ein unattraktives Ausweichziel wäre.

Zudem erscheint ein solcher Ansatz zur Schadensbegrenzung ungerecht, da Unschuldige gefährdet werden, die sich vollkommen regelkonform verhalten, so wie der LKW oder der Passant in Abbildung 26.3, während die beiden Verantwortlichen des unvermeidlichen Unfalls verschont bleiben. Hevelke und Nida-Rümelin[79] schlagen daher vor, als Grundsatz die Verpflichtung zu vorhersehbarem Verhalten zu verankern. Dieses Prinzip liegt bereits heute der Straßenverkehrsordnung zugrunde. So dürfen Autos nur auf vorgesehenen Fahrbahnen und nicht auf dem Bürgersteig fahren, müssen Verkehrsschilder und Ampeln beachten und Fahrtrichtungsänderungen oder Spurwechsel den anderen Verkehrsteilnehmern anzeigen. Auf diese Weise wird das Verhalten der Verkehrsteilnehmer für die anderen berechenbar. Dadurch erhält Regelkonformität durch seine Vorhersehbarkeit jedoch ein hohes moralisches Gewicht.

Regel 26.8 (Grundsatz des vorhersehbaren Verhaltens). *Ein künstliches intelligentes System muss sich stets möglichst vorhersehbar verhalten. Wenn Regeln für seinen Umweltbereich gelten, muss es sich insbesondere ihnen gemäß verhalten. Umgekehrt muss es voraussetzen können, dass alle anderen Akteure sich ebenso verhalten.*

Beispiel 26.9. In der Situation in Abbildung 26.3 würde ein regelkonformes Verhalten implizieren, dass das Fahrzeug zwar bremsen, aber nicht ausweichen darf, wenn dadurch andere Menschen zu Schaden kommen würden. Die Autoinsassen, der Lastwagenfahrer sowie der Passant auf dem Gehweg haben einen stärkeren Anspruch auf Schutz als die beiden Personen auf der Straße[80]. □

26.10.1 Prinzipielle Grenzen der Maschinenethik

Abgesehen von den obigen juristischen und algorithmischen Überlegungen zum Verhalten autonomer künstlicher Systeme stellt sich die Frage nach grundsätzlichen Grenzen der Ethik für Maschinen. Geht man davon aus, dass ein autonomes künstliches System sein Verhalten auf der Grundlage von programmierbaren Algorithmen entscheidet, also dem theoretischen Modell einer Turingmaschine entspricht, so existieren nach Englert et al. (2014) prinzipiell unentscheidbare Situationen. Sie basieren auf der Unentscheidbarkeit des Halteproblems[81] für Situationen, die Varianten des moralischen Dilemmas des Weichenstellers (Trolley-Problem) darstellen.

26.11 Was müssen wir tun?

Die sich stetig beschleunigende Digitalisierung der Berufswelt wird die Rolle der Menschen darin grundlegend ändern. Roboter übernehmen Fließbandarbeiten und Algorithmen verrichten Dienstleistungen und zunehmend auch Managementaufgaben effizienter als wir. Künstliche Intelligenz wird von sozialen Netzwerken wie Facebook eingesetzt, um Nutzerdaten nach ökonomisch verwertbaren Sachverhalten zu durchkämmen: Was interessiert die Nutzer, wie fühlen sie? Nie zuvor war es möglich, soviel über einen einzelnen Menschen in Erfahrung zu bringen.

Solche Kenntnisse über Individuen sind auch politisch verwertbar. Wer weiß, welche Fragen Sie umtreiben, wo Sie sich informieren und auf wessen Ansichten Sie Wert legen, kann Sie

[79] Hevelke und Nida-Rümelin (2015).
[80] Hevelke und Nida-Rümelin (2015).
[81] Sipser (2006); de Vries (2012b).

passgenau lenken. Im Extrem wird dadurch menschliches Verhalten programmierbar, die Gesellschaft wird automatisiert. Schon heute kann unser Verhalten durch weltweit agierende Konzerne, Regierungen oder Hacker manipuliert oder gelenkt werden, ohne dass es uns bewusst wird. Die rasante Entwicklung der Künstlichen Intelligenz wird die Gefahren einer Verhaltenssteuerung, einer Kybernetik des Menschen nur erhöhen.[82]

Wissenschaftler wie der Physiker Stephen Hawking warnen davor, dass eine zukünftige selbstlernende künstliche Intelligenz die Menschheit überholen könnte: "The development of full artificial intelligence could spell the end of the human race. [...] It would take off on its own, and re-design itself at an ever increasing rate. [...] Humans, who are limited by slow biological evolution, couldn't compete, and would be superseded."[83] Eine solche Technologie könnte die Finanzmärkte beherrschen, menschliche Forscher übertreffen, politische Regierungen manipulieren und Waffen entwickeln, die wir nicht einmal verstehen können.

Ein breiter Konsens sind die offene Briefe des Future of Life Institutes zur Vermeidung eines Wettrüstens autonomer Waffensysteme[84] und zu einer Forschung für eine robusten und wohltätigen Künstlichen Intelligenz,[85] die zahlreiche Wissenschaftler und Experten Künstlicher Intelligenz unterzeichnet haben.

[82] Helbig et al. (2016).

[83] http://www.bbc.com/news/technology-30290540

[84] http://futureoflife.org/open-letter-autonomous-weapons/

[85] http://futureoflife.org/ai-open-letter/

Das Geld wurde genauso wenig „erfunden“ wie Musik oder Mathematik oder Schmuck. Was wir „Geld“ nennen, ist kein „Ding“, sondern eine Methode, Dinge [*mathematisch*] *nach ihrer Struktur zu vergleichen, also im Verhältnis zueinander auszudrücken und etwa zu sagen*: *X entspricht sechsmal Y. So gesehen ist Geld wahrscheinlich so alt wie das menschliche Denken.*

David Graeber, *Auf der Suche nach einem Mythos*[a]

[a] Graeber (2012):S. 56.

27 Digitales Geld

Kapitelübersicht

27.1 Definition und Geschichte des Geldes

Geld ist ein in einer Gesellschaft allgemein anerkanntes Zahlungs- und Tauschmittel gegen Waren bzw. Dienstleistungen in ökonomischen Transaktionen. Gleichzeitig dient es als eine Verrechnungseinheit zur Bewertung und Berechnung von Waren und Dienstleistungen in einem wirtschaftlichen System und erfüllt zusätzlich die Funktion eines Wertspeichers, der ökonomische Transaktionen über lange Zeiträume und große Entfernungen hinweg ermöglicht.[1] Um diese drei Geldfunktionen – d.h. Zahlungsmittel, Verrechnungseinheit und Wertspeicher – erfüllen zu können, muss Geld also anerkannt, verfügbar, dauerhaft, tauschbar, handlich, fälschungssicher und zuverlässig sein. Ersteres ist dabei das wichtigste Kriterium, der Name „Geld“ kommt nicht zu Unrecht vom althochdeutschen „gelten“.

Als historisch erste Form des Geldes entwickelte sich in fast allen bekannten Kulturen

[1] Abel et al. (2008):§7.1; Bofinger (2007):S. 594; Ferguson (2009):S. 25; P. R. Krugman und Wells (2006):S. 322f.

der Menschheit das Warengeld, also eine Naturalie, die allgemein als nützlich oder wertvoll angesehen war. Meist waren es Edelmetalle wie Gold, Silber und Bronze, aber in einigen Regionen auch Muscheln oder Schnecken (Nordamerika, Afrika und China), Steine (Neuguinea und Südpazifik) oder Pelze (Nordamerika). Die frühesten bekannten Münzen stammen aus der Zeit um 700 v.u.Z. und wurden im Artemis-Tempel in Ephesus bei Izmir in der heutigen Türkei gefunden[2], siehe Abbildung 27.1. Diese Münzen waren Vorläufer des attischen Tetradrachmons,

Abbildung 27.1: Münzen aus Ephesus in Lydien. V.l.n.r.: Die älteste bekannte Münze (etwa 700 v.u.Z.) mit einem Hirsch, einer dort verehrten Gottheit, sowie zwei Münzen aus Elektrum, einer natürlich vorkommenden Legierung aus Gold und Silber, mit einem Löwenkopf und einem Hirsch (um 600 v.u.Z.). Quellen: `http://snible.org/coins/hn/ionia.html#Ephesus`, `http://cngcoins.com/Coin.aspx?CoinID=57383`, `http://cngcoins.com/Coin.aspx?CoinID=50977`

einer Silbermünze mit dem Kopf der Göttin Athene auf der einen Seite und einer Eule als Symbol der Weisheit auf der anderen. In römischer Zeit wurden Münzen aus drei verschiedenen Edelmetallen hergestellt, der Aureus aus Gold, der Denar aus Silber und der Sesterz aus Bronze. Das römische Münzsystem überlebte sogar das Römische Reich in Europa, noch in der Epoche Karls des Großen 400 Jahre nach dem Untergang Westroms wurden die Preise in Silberdenaren angegeben[3].

Das erste Papiergeld wurde 1294 im China der mongolischen Yuan-Dynastie unter dem Enkel von Dschingis Khan (in der Provinz Ilchane auf dem Gebiet des heutigen Iran) ausgegeben, scheiterte jedoch schnell aufgrund mangelnden Vertrauens in die Währung.[4] Die ersten *Zentralbanken*, also Finanzinstitutionen, die die Währung eines Staates verwalten, waren die 1609 in den Niederlanden gegründete *Amsterdamsche Wisselbank* und die 1656 gegründete *Stockholms Banco*. Die *Amsterdamsche Wisselbank* vergab Schuldscheine *(wissel)* für Goldbarren und Münzen, die bei ihr hinterlegt wurden. Die *wissels* konnten beliebig von jedem ge- und verkauft werden, der darauf vertraute, dass er sie wieder gegen Goldbarren oder Münzen einlösen konnte. Erst 1790, also nach fast zwei Jahrhunderten, ging die Bank schließlich bankrott, nachdem die Beleihung großer Summen an die Stadt Amsterdam und der privaten Niederländischen Ostindien-Kompanie (VOC) öffentlich bekannt wurde.

Die *Stockholms Banco* war 1661 die erste Bank, die Gutschriften *(kreditivsedlar)* in runden Beträgen vergab, also Banknoten in unserem heutigen Sinne. Allerdings konnte sie unbegrenzt Banknoten drucken, was bereits im Herbst 1663 eine Inflation verursachte. 1664 hatte die Bank zuwenig Metall, um die Banknoten einzulösen, und musste den Betrieb einstellen.

27.1.1 Kredittheorie des Geldes

Die *Bank of England* wurde 1694 als ein Konsortium privater englischer Bankiers gegründet und lieh dem englischen König Wilhelm III. eine Summe von 1.2 Million £. Durch den Neunjährigen Krieg (1688–97) und insbesondere nach der schweren Niederlage in der Schlacht von Beachy

[2]Ferguson (2009):S. 25.

[3]Ferguson (2009):S. 26.

[4]Der damalige Herrscher Gaichatu hatte vorher die Staatskasse für seinen dekadenten Lebensstil und aufgrund einer in seinem Herrschaftsgebiet ausgebrochenen Rinderpest geplündert und führte die Banknoten erstmalig in der Menschheitsgeschichte als *ausschließliches* gesetzliches Zahlungsmittel ein, um die Kontrolle über den Staatsschatz zu behalten. Das Vorhaben erwies sich als völliger Fehlschlag. Gaichatu musste sein Dekret 1295 widerrufen und wurde kurz darauf ermordet.

Head 1690 gegen die französische Flotte benötigte der König Geld zum Wiederaufbau einer starken Marine und zur Fortführung des Krieges gegen Frankreich. Das Darlehen wurde ihm für einen jährlichen Zins von 8 Prozent gewährt. Im Gegenzug erhielt die Bank of England das königliche Privileg für die Ausgabe von *Banknoten*. Praktisch bedeutete dies, dass die Bank of England das Recht hatte, für einen Teil des Geldes, dass der König ihr schuldete, an jeden Bürger Schuldscheine auszugeben, der Geld von ihr leihen oder bei ihr anlegen wollte. Das funktionierte jedoch nur so lange, wie die englische Krone ihre Schulden nicht beglich. „Bis heute wurde dieser Kredit nicht zurückgezahlt. Er kann nicht zurückgezahlt werden. Wenn er jemals zurückgezahlt würde, wäre dies das Ende des britischen Währungssystems.“[5] Was war hier geschehen?

Beispiel 27.1. *(Graebers Ökonomie der Urschulden)* Stellen wir uns Henry und Joshua vor, die in einer kleinen Stadt mit einer Wirtschaft ohne Geld leben. Henry hat Kartoffeln und Joshua hat ein Paar Schuhe übrig. „Henry trifft Joshua und sagt: ‚Hübsche Schuhe!‘ Joshua erwidert: ‚Ach, sie sind nichts Besonderes, aber wenn sie dir gefallen, kannst du sie gern haben.‘ Henry nimmt die Schuhe.“[6] Henry schuldet Joshua nun einen Gefallen, oder mit anderen Worten, er hat bei ihm einen Kredit. Wie kann er ihn aber zurückzahlen, wenn Joshua gar keine Kartoffeln braucht?

„Nehmen wir einmal an, [. . .] Henry verspräche ihm, statt ihm einen Gefallen zu schulden, etwas von gleichem Wert. Henry gibt Joshua einen Schuldschein. Joshua könnte warten, bis Henry etwas für ihn Nützliches hat, und dann den Schuldschein einlösen. Dann würde Henry den Schuldschein zerreißen und die Sache wäre erledigt. Aber nehmen wir weiter an, Joshua würde den Schuldschein an eine dritte Person – Sheila – weitergeben, der er etwas anderes schuldet. [. . .] Nun schuldet Henry ihr die Summe. Und damit ist das Geld geboren. Denn die Weitergabe hat kein logisches Ende. Nehmen wir an, Sheila will ein Paar Schuhe von Edith erwerben, dann kann sie einfach Edith den Schuldschein geben und ihr versichern, dass Henry dafür geradesteht. Prinzipiell gibt es keinen Grund, warum der Schuldschein nicht jahrelang in der Stadt zirkulieren könnte – vorausgesetzt, die Menschen haben weiterhin Vertrauen in Henry. Wenn es lange genug geht, vergessen die Menschen womöglich vollständig, von wem der Schuldschein ursprünglich stammte.“[7] Aber: „Der Schuldschein kann nur so lange als Geld zirkulieren, wie Henry seine Schuld nicht bezahlt.“[8] □

Die Grundlage des Geldes eines Staates ist also immer eine Urschuld *(primordial debt)* des Staates, und der Wert des Geldes basiert auf dem Vertrauen darauf, dass sie am Ende beglichen werden kann. Das ist, grob verkürzt, die *Staatliche Kredittheorie* des Geldes von Knapp[9].

> „So lange die Banknoten nicht staatlich akzeptiert sind, stellen sie nach unserer Auffassung Chartalgeld einer unstaatlichen Zahlgemeinschaft dar, sind also ein besonderer Fall eines Zahlungsmittels von privater Emission.“[10]

> „In einem Zahlverbande ist jede übertragbare Verfügung über Werteinheiten dann Zahlungsmittel, wenn der Inhaber durch Übertragung an die Zentralstelle eine mindestens eventuale Gegenforderung an diese Stelle begründen kann. [. . .] Die Banknote, als chartales Zahlungsmittel von privater Emission, ist zunächst nur Privatgeld; sie kann aber zu Staatsgeld werden, sobald der Staat die Akzeptation ausspricht,

[5] Graeber (2012):S. 53.
[6] Graeber (2012):pp 39.
[7] Graeber (2012):S. 50f.
[8] Graeber (2012):S. 52.
[9] Knapp (1905).
[10] Knapp (1905):S. 133.

> indem er erklärt, dass die Banknoten an seinen Kassen als Zahlungsmittel angenommen werden. Die Girozahlung ist ebenfalls zunächst, ihrer geschichtlichen Entstehung nach, eine Zahlung in privaten Gemeinschaften; aber auch sie kann zur Zahlung in der staatlichen Gemeinschaft erhoben werden, ebenfalls durch Akzeptation: indem der Staat in die Girogemeinschaft eintritt und also zulässt, dass Zahlungen an ihn durch Benutzung der Giroeinrichtung geleistet werden dürfen. Hierbei wird nicht ein sachliches Zahlungsmittel akzeptiert, sondern ein rechtliches Zahlungsverfahren."[11]

Für Geld, dessen Wert sich ausschließlich durch die offizielle Anerkennung als Zahlungsmittel ergibt, wird auch die Bezeichnung *Fiatgeldes* (lateinisch *fiat* – „es entstehe")[12] oder *Chartalgeld* genannt[13]. Ein durch ein Edelmetall oder eine andere Ware gedecktes Geld ist also eine Zwischenform von Waren- und Fiatgeld.

Doch warum sollte ein Staat überhaupt daran interessiert sein, Geld zu schaffen? Warum nicht einfach die Kontrolle über die Gold- und Silbervorkommen auf dem eigenen Territorium und möglichst auch anderswo erlangen, so wie es die Könige in der Antike und auch der beginnenden Neuzeit taten? Graebers[14] Antwort darauf ist kurz: Schaffe Märkte durch Steuern, um Steuern einzunehmen.

27.1.2 Steuern und Märkte

In dem Jahrtausend vor unserer Zeitrechnung befanden sich Zivilisationen in fast allen Teilen der Erde in einer Periode anhaltender Expansion. Um ihre zunehmende Komplexität weiter zu beherrschen und einen Zusammenbruch zu vermeiden, mussten sie sich nach Morris[15] restrukturieren und neue Institutionen erfinden. Ihre bis dahin praktizierten „Low-End-Strategien" waren nicht mehr effektiv genug: Morris bezeichnet mit diesem Begriff das Vorgehen eines Staatslenkers, sich auf lokale Eliten zu stützen, wenn möglich auf eigene Verwandte oder auf Clans, die in ihren Territorien Truppen aushoben und an den Erträgen und Plünderungen nach militärischen Erfolgen beteiligt wurden. Solange eine solche Low-End-Zivilisation die Kriege mit konkurrierenden Staaten gewann, konnte sie damit eine positive Bilanz ziehen, da geringen Einkünften noch geringere Ausgaben gegenüber standen. Diese Strategie kommt jedoch an ihre Grenzen, wenn die Komplexität des Staatswesens durch Expansion zu hoch wird und nicht mehr durch Eliten und Repression aufrecht erhalten werden kann, das Herrschaftssystem also zusammenbricht.

Ein Klimawandel in den Jahrhunderten zwischen 800 und 500 v.u.Z. bewirkte, dass die Bevölkerung in Nord- und Westeuropa sowie in Nordchina schrumpfte, jedoch am Mittelmeer und den südlichen Tälern des Jangtse und des Gelben Flusses anstieg; in Europa wuchs die Bevölkerung Griechenlands am kräftigsten[16]. Nun musste in diesen Regionen dieselbe Ackerfläche eine größere Bevölkerung ernähren, was sowohl Konflikte als auch Innovationen hervorbrachte, beispielsweise in Griechenland ein Alphabet und die Wissenschaft. „Um 770 v.u.Z. büßten die Könige in beiden Kernregionen [China und Ägypten/Mesopotamien] die Macht über ihre Vasallen ein. Der ägyptische Staat, seit 945 v.u.Z. mehr oder weniger geeint, zerbrach 804 v.u.Z. in drei Fürstentümer und löste sich bis 770 v.u.Z. in ein Dutzend praktisch unabhängiger Herzogtümer weiter auf. In Assyrien musste Schamschi-Adad V. 823 v.u.Z. um den Thron kämpfen,

[11] Knapp (1905):S. 143f.
[12] P. Krugman (2009):S. 324.
[13] Graeber (2012):S. 496, Fußnote 74.
[14] Graeber (2012):S. 53f.
[15] Morris (2010):§5.
[16] Morris (2010):Fig. 5.4.

verlor dann aber die Macht über Vasallenkönige und Statthalter. [. . .] Ein ziemlich ähnliche[r] Staatszusammenbruch [. . .] ereignete sich auch im Osten, als die Bevölkerung dort zu wachsen begann. Um 810 v.u.Z. verlor der Zhou-König Xuan die Macht über seine Lehnsfürsten. [. . .] In den 770er Jahren v.u.Z. – im selben Jahrzehnt, in dem ägyptische und assyrische Herrscher Macht und Einfluss verloren – [haben offenbar] Bevölkerungswachstum, erstarkte Lokalfürsten, dynastische Politik und äußerer Druck in China zusammengewirkt und dem Königtum einen noch deutlicheren Rückschlag zugefügt [nämlich indem der König erschlagen und die Hauptstadt niedergebrannt wurde].“[17]

In den Jahren 750 bis 500 v.u.Z restrukturierten sich die Kernreiche des südöstlichen Mittelmeerraums und Asiens – Ägypten, Assyrien und China – und entwickelten politische, wirtschaftliche und intellektuelle Ressourcen, um die neuen Herausforderungen zu bewältigen. Im Wesentlichen wechselten die Staaten zu einer „High-End-Strategie,“ das heißt sie zentralisierten die Macht durch ein stehendes Heer und durch eine Verwaltung zur Finanzierung des Systems. „Was sich verändert hat, erkennen wir zuerst in Assyrien. Der Usurpator, der 744 v.u.Z. als Tiglat-Pileser III. den Thron bestieg, [. . .] katapultierte [. . .] den heruntergewirtschafteten Low-End-Staat Assyrien in weniger als 20 Jahren ans dynamische High-End. [. . .] Tiglat-Pileser – und das war das Geheimnis seines Erfolgs – hielt die adligen Söhne des Himmels aus allem heraus, stellte stattdessen ein stehendes Heer auf, das nicht mehr von den Lehnsfürsten unterhalten wurde, sondern vom König selbst, auf den es eingeschworen wurde. [. . .] Auf dieses Heer gestützt, brach er die Macht des Adels. Er schwächte Spitzenpositionen, indem er sie untergliederte und teils auch mit gefangenen Eunuchen besetzte. [. . .] All das kostete Geld. Tiglat-Pileser musste also auch die Staatsfinanzen regeln. Statt Fremde zu erleichtern, indem er ab und an auftauchte und Zahlungen erzwang, bestand er auf regelmäßigen Kontributionen – meist in Form von Steuern. [. . .] Schon bevor Tiglat-Pileser in Assyrien die Macht ergriff, wurde Ägypten von Nubiern aus dem Gebiet des heutigen Sudan vereinigt, die in den folgenden 30 Jahren Reformen einleiteten, die auch die Zustimmung des Assyrers gefunden hätten. In den 710er Jahren v.u.Z. tat es ihnen sogar Hiskija (Ezechia), der König des kleinen Juda, gleich.[18]

Der einfachste und effizienteste Mechanismus zur Erzeugung von hohen Steuereinnahmen zur Finanzierung eines Heeres und einer Verwaltung zur Eintreibung und Verarbeitung der Staatseinnahmen war nach Graeber[19] die Schaffung von Märkten.

> „Sagen wir, ein König möchte ein stehendes Heer von 50 000 Mann unterhalten. [. . .] Wenn man den Soldaten [. . .] einfach Münzen gab und dann verfügte, jede Familie im Königreich habe dem König eine solche Münze zu zahlen, dann hatte man mit einem Schlag seine ganze Volkswirtschaft in eine gewaltige Maschinerie zur Versorgung der Soldaten verwandelt. Denn um an die Münzen zu kommen, musste jede Familie einen Weg finden, wie sie auf ihre Weise zu der allgemeinen Anstrengung, die Soldaten zu unterhalten, beitragen konnte. Als Nebeneffekt entstanden Märkte.[20]

Tatsächlich zeigen die historischen Quellen, dass Märkte üblicherweise nicht spontan entstanden, sondern sich im Umfeld antiker Armeen entwickelten. Gesellschaften ohne Staat dagegen sind in der Regel Gesellschaften ohne Märkte[21].

---

[17] Morris (2011):S. 237–241.
[18] Morris (2011):S. 243f.
[19] Graeber (2012):§3.
[20] Graeber (2012):S. 53.
[21] Graeber (2012):S. 54.

27.1.3 Zusammenfassung

Geld erfüllt drei Funktionen:

1. *Zahlungsmittel:* Geld dient in ökonomischen Transaktionen dazu, gegen Erhalt einer Ware oder Dienstleistung eingetauscht zu werden.
2. *Werteinheit:* Als Verrechnungseinheit bewertet Geld Waren und Dienstleistungen.
3. *Wertspeicher:* Geld ermöglicht ökonomische Transaktionen, die sich über Zeit und Raum erstrecken.

Um diese drei Funktionen erfüllen zu können, muss Geld anerkannt, verfügbar, dauerhaft, tauschbar, handlich, fälschungssicher und zuverlässig sein.

Die historisch erste Form von Geld war Warengeld, das beispielsweise auf Metallen, Muscheln oder Schmuck beruhte. Mit dem Aufkommen höherer Staatsgebilde wurden Schuldscheine als eine Frühform staatlicherseits anerkannten Geldes verwendet, sogar lange bevor autorisierte Münzen aus Gold, Silber oder Bronze in Umlauf kamen. Der Bargeldkreislauf mit Münzen prägte seit etwa 600 v.u.Z. für über zwei Jahrtausende das Wirtschaftsleben der Volkswirtschaften in der Antike, dem Mittelalter und der beginnenden Neuzeit. Im siebzehnten Jahrhundert entstanden nacheinander in Amsterdam, Stockholm und London die ersten Banknoten in Europa, von denen das Pfund Sterling als Währung bis heute Bestand hat. Bis ins 20. Jahrhundert war eine Banknote mit dem Versprechen verknüpft, es bei der Zentralbank gegen eine festgelegte Menge Gold bzw. Silber eintauschen zu können („Goldstandard“).

Mit der Verbreitung von Banknoten wurde jedoch offenbar, dass der Wert des Geldes nicht vom Wert des einlösbaren Edelmetalls abhing, sondern allein von der Höhe der Vertrauens in den Staat, die Dauerhaftigkeit des Wertes der Währung zu garantieren.

27.2 Geld als Währung

27.2.1 Die Zentralbank und der Geldangebotsprozess

In der Volkswirtschaftslehre ist eine *Zentralbank*, oder auch *Notenbank*, die für die Geld- und Währungspolitik eines Staates oder eines Währungsraums zuständige Institution. Sie hält die Währungsreserve des Währungsraums und refinanziert die Geschäftsbanken und den Staat. Dadurch entstehen zwei wechselwirkende Kreditkreisläufe, die den *Geldangebotsprozess* bilden, siehe Abbildung 27.2. Der eine Kreislauf beschreibt die Kreditvergabe zwischen einer Ge-

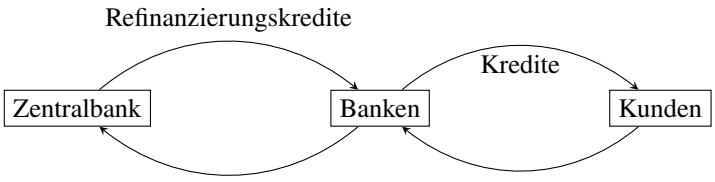


Abbildung 27.2: Die beiden Kreditkreisläufe des Geldangebotsprozesses.

schäftsbank und ihren Kunden, die mehr Geld als verfügbar benötigen. Zwar können Banken einen großen Teil ihres Kreditgeschäfts aus den Sichteinlagen[22] anderer Kunden finanzieren, für den Fall jedoch, dass sie ihren Liquiditätsbedarf nicht decken können, müssen sie sich bei

[22] Sichteinlagen sind Bankguthaben mit Laufzeiten oder Kündigungsfristen von weniger als einem Monat, also insbesondere Girokonten.

anderen Banken oder bei der Zentralbank verschulden. Dazu brauchen die Banken Guthaben bei der Zentralbank, die sogenannten *Reserven*. Da die Reserven die umlaufende Geldmenge bestimmt, werden sie oft auch als *Geldbasis* oder *Zentralbankgeld* bezeichnet. Die an die Banken vergebenen Kredite heißen *Refinanzierungskredite*[23].

Da die Zentralbank das Währungsmonopol hat, kann sie grundsätzlich kein Liquiditätsproblem bekommen, denn sie kann unbegrenzt Banknoten emittieren. Wenn Anleger jedoch aus der Landeswährung in eine andere flüchten (*„Kapitalflucht"*), kann es zu einer Währungskrise kommen, also einem Kursverfall der eigenen Währung gegen andere Währungen. Zwar beschafft sich die Zentralbank für diesen Fall Währungsreserven, indem Geschäftsbanken ihr kurzfristige Forderungen gegenüber einer ausländischen Bank verkaufen, sogenannte *Devisen*, und sich den Gegenwert auf ihrem Notenbankkonto gutschreiben lassen. Falls diese Währungsreserven auch nicht ausreichen, kann nur noch der Internationale Währungsfonds (IWF) helfen.

Betrachten wir die sich aus dem Geldangebotsprozess ergebende *konsolidierte Bilanz* des gesamten Bankensystems. Dabei werden zunächst die Bilanzen aller Banken einschließlich der Zentralbank aggregiert, d.h. durch die Konsolidierung entfallen alle Beziehungen zwischen den einzelnen Geschäftsbanken und der Zentralbank. In der konsolidierten Bilanz erscheinen also nur noch Forderungen und Verbindlichkeiten gegenüber den Nichtbanken. Wird zur Vereinfachung angenommen, dass es neben den Sichteinlagen keine weiteren Anlagemöglichkeiten bei den Banken gibt, so lässt sich die konsolidierte Bilanz des Bankensystems also wie folgt darstellen:

Geschäftsbanken

Kredite an Nichtbanken	Bargeld Sichteinlagen	(27.1)
Sonstige Aktiva	Sonstige Passiva	

Auf der Passivseite stehen die Bargeldbestände, also die im Umlauf befindlichen Münzen und Banknoten, und die Sichteinlagen, die Nichtbanken, also Privatpersonen und Unternehmen, bei den Banken halten. Auf der Aktivseite steht das Gesamtvolumen der Bankkredite an die Nichtbanken. Man sieht bei dieser Darstellung deutlich die Spiegelbildlichkeit von Geld und Kredit. Die zweite für den Geldangebotsprozess wichtige Bilanz ist diejenige der Zentralbank:

Zentralbank

Währungsreserven	Bargeld	
Kredite an den Staat	Reserven der Geschäftsbanken	(27.2)
Kredite an Geschäftsbanken		
Sonstige Aktiva	Sonstige Passiva	

Man sieht an dieser Bilanz, dass Geld geschöpft werden kann durch (a) Kredite an die Geschäftsbanken, (b) direkte Kredite an den Staat, und (c) Ankauf ausländischer Devisen von den Geschäftsbanken, wodurch die Währungsreserven der Zentralbank steigen. Problematisch für die Geldpolitik sind dabei vor allem die Kredite an den Staat und die Einflüsse der Außenwirtschaft, da in beiden Fällen die die Geldbasis nicht durch die inländische Realwirtschaft beeinflusst wird.

Beispiel 27.2. *Aktiva-Kauf durch die Zentralbank.*[24] Kauft eine Zentralbank Schatzbriefe im Wert von 100 Mio € von Geschäftsbanken, so erhöht sie sowohl ihre Aktiva als auch ihre Passiva um diesen Betrag. Die Geschäftsbanken dagegen verändern ihre Passiva überhaupt nicht, in ihrer Bilanz verlieren sie Schatzbriefe als Aktiva, gewinnen aber Geld als Umlaufvermögen im

[23] Bofinger (2007):§21.

[24] P. R. Krugman und Wells (2006):Figure 13-7.

gleichen Wert.

Zentralbank

Schatzbriefe	+ 100 Mio €	Geld	+ 100 Mio €

Geschäftsbanken

Schatzbriefe	– 100 Mio €		
Geld	+ 100 Mio €		

Auf diese Weise führt der Kauf von Wertpapieren durch die Zentralbank sofort zu einer Erhöhung der Geldmenge um denselben Betrag. □

Wenn eine Staatsregierung einen direkten Kredit bei der Notenbank aufnimmt, verfügt sie über ein Notenbankguthaben. Sobald sie damit Zahlungen an Geschäftsbanken vornimmt, steigen deren Notenbankguthaben und die Geldbasis erhöht sich. Da alle großen Inflationen der Wirtschaftsgeschichte, insbesondere die Hyperinflation 1923 in Deutschland, über diesen Weg entstanden sind, ist eine direkte Staatsfinanzierung durch die EZB verboten.[25] Der andere Faktor, die Außenwirtschaft, beeinflusst die Geldbasis durch Devisenankauf und ist erst dann problematisch, wenn eine Zentralbank *gezwungen* ist, in größerem Umfang Devisen anzukaufen. Beispielsweise war dies in den frühen 1970er Jahren der Fall, als die meisten Zentralbanken im Rahmen des internationalen Währungsabkommens von Bretton Woods einen festen Wechselkurs zum US-Dollar aufrechterhalten mussten. Da der Dollar nach Ende des Vietnamkrieges, der Ölkrise und Aufgabe des Goldstandards 1973 stark an Wert verlor, mussten die internationalen Zentralbanken den Wechselkurs durch massive Dollarankäufe verteidigen. Eine zielgerechte, an der Realwirtschaft orientierte Kontrolle der Geldbasis war so nicht mehr möglich, wodurch es 1974/1975 weltweit zu hohen Inflationsraten kam.[26]

Bemerkung 27.3. In der Ökonomik nach wie vor ungeklärt ist die Frage nach dem aus geldpolitischer Sicht optimalen Vorgehen in einem System *flexibler* Wechselkurse. Nach Ansicht vieler Ökonomen sollte die Zentralbank überhaupt nicht in das freie Spiel des Devisenmarktes eingreifen, denn da sie dann Devisen weder ankaufen noch verkaufen würden, ergäben sich keine außenwirtschaftlichen Störeffekte für die Geldbasis. Auf der anderen Seite stärken stabile Wechselkurse die Handelsbeziehungen zwischen Volkswirtschaften. Ein interessanter Vergleich der Schweiz und Österreichs, deren Zentralbanken gegenüber der D-Mark ein genau entgegengesetztes Vorgehen wählten, findet sich in Bofinger (2007:§25.5.2): Die Schweiz entschied sich 1973 für eine konsequente Politik der flexiblen Wechselkurse, während Österreich einen festen Wechselkurs gegenüber der D-Mark vorsah. In den anderthalb Jahrzehnten von 1990 bis 2006 betrug das jährliche durchschnittliche Wachstum des realen Bruttoinlandsprodukts der Schweiz 1,3 %, das Österreichs dagegen 2,3 %. Dafür war die Inflationsrate des Schweizer Franken mit durchschnittlich 0,8 % geringer als diejenige Österreichs mit 2,0 %. □

27.2.2 Die Geldmenge

In einem Zentralbanksystem bezeichnet die *Geldmenge* die Gesamtsumme an Bargeld und bestimmter Guthaben der Nichtbanken einer Volkswirtschaft. Es werden dabei drei Geldmengenkonzepte voneinander abgegrenzt, die die einzelnen Zentralbanken im Detail jeweils leicht

[25] Bofinger (2007):S. 454.
[26] Bofinger (2007):S. 455.

unterschiedlich definieren. Im Folgenden wird die Geldmengendefinition der Europäischen Zentralbank (EZB) dargestellt.[27] Die Geldmenge M1 setzt sich zusammen aus dem umlaufenden Bargeld und den Sichteinlagen der Nichtbanken,

$$\text{M1} = \text{Bargeld} + \text{Sichteinlagen der Nichtbanken.} \tag{27.3}$$

Dagegen erfasst die Geldmenge M2 zusätzlich längerfristige Einlagen,

$$\begin{aligned}\text{M2} = {} & \text{Geldmenge M1} \\ & + \text{Einlagen mit Kündigungsfrist} \leqq 3 \text{ Monate} \\ & + \text{Einlagen mit Laufzeit} \leqq 2 \text{ Jahre}\end{aligned} \tag{27.4}$$

Die Geldmenge M3, die bei der Geldpolitik der EZB im Vordergrund steht, ist noch breiter abgegrenzt:

$$\begin{aligned}\text{M3} = {} & \text{Geldmenge M2} \\ & + \text{Repogeschäfte} \\ & + \text{Geldmarktfondsanteile und Geldmarktpapiere} \\ & + \text{Schuldverschreibungen mit Laufzeit} \leqq 2 \text{ Jahre}\end{aligned} \tag{27.5}$$

(Bei einem *Repogeschäft* verkauft die Bank an den Kunden ein Wertpapier, dessen Rückkauf zu einem festen Kurs und zu einem festgelegten Termin vereinbart wird. Die Bank muss Repogeschäfte nicht in ihrer Bilanz aufführen[28].)

Die Geldmenge hat großen Einfluss auf das Wachstum und die Inflation einer Volkswirtschaft. Ist zu wenig Liquidität vorhanden, d.h. existiert eine „Geldlücke“, so reduziert sich das Wirtschaftswachstum. Ist durch starkes Geldmengenwachstum dagegen zuviel Liquidität im Markt, so besteht die Gefahr einer Inflation.

Die *reale Geldmenge* M_r bezeichnet die preisbereinigte nominale Geldmenge M_n. Sie wird als Quotient von nominaler Geldmenge und Preisniveau P definiert und ist eine variable Größe, die die Zentralbank durch die nominale Geldmenge steuern kann:

$$M_r = \frac{M_n}{P}. \tag{27.6}$$

Das Preisniveau P ist hierbei üblicherweise ein Index bezüglich des allgemeinen Preisniveaus eines gegegeben Basisjahres und wird maßgeblich durch die Inflation beeinflusst.

27.2.3 Geldschöpfung und Geldvernichtung

Geldschöpfung (money creation) bezeichnet den Prozess der Schaffung neuen Geldes. Grundsätzlich kann die Geldmenge auf drei Arten erhöht werden: (1) Die Zentralbank kann Bargeld in Umlauf bringen, (2) die Zentralbank oder die Geschäftsbanken können Kredite an Nichtbanken vergeben, oder (3) sie können Aktiva wie Devisen, Immobilien, Edelmetalle oder Wertpapiere von Nichtbanken ankaufen. Entsprechend wird durch Rückzahlung von Krediten von Nichtbanken und durch Verkauf von Aktiva von Banken an Nichtbanken Geld wieder vernichtet.

Früher beschränkte die Bankenaufsicht die Geldschöpfung durch Kreditvergabe mit Festsetzung einer Mindestreserve, die bei der Zentralbank als Sicherheit für vergebene Kredite

[27] `https://www.ecb.europa.eu/stats/money_credit_banking/monetary_aggregates/html/hist_content.en.html` [2021-09-21]

[28] Bofinger (2007):S. 452.

hinterlegt werden musste. Mit Basel III spielt die Mindestreserve jedoch eine untergeordnete Rolle gegenüber der Eigenkapitalquote von Kreditinstituten. Die Kreditvergabe wird dabei durch zwei Mindestsätze beschränkt, die auf unterschiedlichen Zeithorizonten eingegangene Risikopositionen gegen liquides Eigenkapital rechnen: Die *Mindestliquiditätsquote LCR (liquidity coverage ratio)* bewertet das kurzfristige Liquiditätsrisiko anhand des Verhältnisses des Bestandes an erstklassigen liquiden Aktiva zum gesamten Nettoabfluss der letzten 30 Tage. Das Verhältnis muss größer 1 sein:[29]

$$\text{LCR} = \frac{\text{Bestand an erstklassigen Aktiva}}{\text{Nettoabfluss in den nächsten 30 Tagen}} \geqq 1. \tag{27.7}$$

Die LCR wird anhand von Stresstests der Bankenaufsicht ermittelt, die Schocks ähnlich denjenigen simulieren, die 2007/2008 die Finanzkrise auslösten. Die *strukturierte Liquiditätsquote NSFR (net stable funding ratio)* dagegen definiert das Verhältnis des verfügbaren stabil refinanzierten Betrags zu dem für eine stabile Refinanzierung erforderlichen Betrag und muss ebenfalls größer 1 sein:[30]

$$\text{NSFR} = \frac{\text{verfügbarer Betrag zur stabilen Refinanzierung}}{\text{erforderlicher Betrag zur stabilen Refinanzierung}} \geqq 1. \tag{27.8}$$

Zur verfügbaren stabilen Refinanzierung (ASF, *abvailable stable funding*) werden dabei verschiedene liquide Mittel angerechnet, insbesondere das Eigenkapital. Der Betrag der erforderlichen stabilen Refinanzierung (RSF, *required stable funding*) setzt sich aus allen Aktiva der Bank zusammen.

27.2.4 Sprachregelung Geld, Währung und *Currency*

Im Deutschen ist eine *Währung* das durch eine staatlich legitimierte Institution, üblicherweise eine Zentralbank, festgelegte Geldsystem. Eine Währung ist also insbesondere die Festlegung der Verrechnungseinheit und der Münzen und Banknoten, aber auch des Währungsraums, also des Geltungsbereichs der Währung. Eine Währung ist so definiert also eine spezielle Form von Geld, und es kann neben einer Währung durchaus anderes Geld als Zahlungsmittel geben.

Demgegenüber wird der englische Begriff *currency* (lateinisch *currens* – umlaufend) üblicherweise allgemeiner als Synonym für Geld aufgefasst, also ein sich im Umlauf befindliches verbreitetes Zahlungsmittel. So spricht man im Englischen von *cryptocurrency* oder *digital currency*, das man im Deutschen eher mit *Kryptogeld* beziehungsweise *Digitalgeld* übersetzen sollte. In diesem Skript wird diese Sprachregelung verwendet.

27.3 Spezifische Eigenschaften digitalen Geldes

Für die Konzeption von digitalem Geld müssen neben den drei Grundfunktionen von Geld einige spezifische Problemstellungen gelöst werden, die bei Bargeld schon allein aufgrund seiner materiellen Natur gar nicht aufkommen.

- *Autorisierter Besitz.* Bei Bargeld ist derjenige, der die Münze oder die Banknote physisch hat, automatisch auch der Eigentümer. Bei digitalem Geld ist ein Besitznachweis dieser Art aber nicht möglich. Digitales Geld benötigt also einen Mechanismus, der für das gesamte System konsistent die Besitzverhältnisse verbindlich festlegt.

[29] Basel Committee on Banking Supervision (2013).
[30] Basel Committee on Banking Supervision (2014).

- *Keine Mehrfachzahlungen.* Ein physisches Geldstück kann nur einmal ausgegeben werden, d.h. es hat nach einer Transaktion den Besitzer gewechselt und kann höchstens durch künftige Transaktionen wieder in seinen Besitz gelangen. Bei digitalem Geld muss gewährleistet sein, dass die Ausgabe eines logischen Geldstücks nur einmal pro Transaktion im gesamten System geschehen kann, also ein Akteur es nicht mehrfach ausgeben kann.

In der Fachliteratur werden zusätzlich weitere Eigenschaften diskutiert, die digitales Geld vorweisen sollte, so beispielsweise die Bindung an eine reale Währung – oder zumindest eine garantierte Konvertibilität zu einer Währung –, die Benutzbarkeit ohne ein personalisiertes Konto, oder Offline-Benutzbarkeit.[31] Diese Eigenschaften sind aber nicht unumstritten.

Beispiel 27.4. *(Buchgeld)* Das herkömmliche Buchgeld, oder Giralgeld, ist zwar ursprünglich kein digitales oder elektronisches Geld,[32] hat aber als bargeldloses Zahlungsmittel dieselben Probleme wie digitales Geld zu lösen. Sie werden behoben, indem je Transaktion zweiseitig konsistent zwischen zwei eindeutig identifizierten Girokonten (Sichteinlagen) gebucht wird: Nach einer abgeschlossenen Buchung ist die Summe der Salden beider Konten gleich. Buchgeld spielt heute weltweit eine weit größere Rolle als Bargeld, und es trägt durch Kreditgewährung der Banken (inklusive eingeräumter Kreditlinien) erheblich zur Geldschöpfung einer Volkswirtschaft bei.[33] □

27.4 Bitcoin

Zwar war der Volkswirtschaftler Georg Friedrich Knapp 1905 mit seiner Abhandlung *Staatliche Theorie des Geldes*[34] seiner Zeit weit voraus, was die Entstehung und das Wesen des Geldes betraf, wie in Abschnitt 27.1.1 auf Seite 164 beschrieben. Allerdings irrte er mit seiner Aussage, dass Geld immer eine „Zentralstelle“ benötigte, die die Zahlungen rechtlich ordnen müsse.

> „Zahlung ist ein Vorgang, der jedenfalls eine Gemeinschaft voraussetzt; ob diese Gemeinschaft der Staat ist oder ein Kundenkreis einer Bank oder sonst ein Zahlverband, ist eine nebensächliche Frage. [. . . Allerdings] muss die Zahlgemeinschaft eine [. . .] Leitung haben: es muss Mächte geben, welche die Art und Weise der Zahlung rechtlich ordnen. Die Zahlgemeinschaften haben alsdann einen Mittelpunkt, von wo die Leitung ausgeht: beim staatlichen Gelde ist es die Staatsgewalt, beim privaten Zahlungswesen ist es beispielsweise die Bank. Halten wir dies alles fest, so ergibt sich ein Ausblick auf eine umfassendere Definition von Zahlung; [. . . es ist] die juristische Übertragung von Gegenforderungen in Werteinheiten und zwar von Gegenforderungen, die an die Zentralstelle gerichtet sind.“[35]

Selbstverständlich trifft Knapps Behauptung für jede Form von Geld zu, die durch eine Zentralbank und damit durch den Staat akzeptiert wird, also für den bei Weitem größten Teil des Geldes der Welt. Jedoch gab es historisch immer wieder Fälle, in denen Zahlungsmittel in einer Gemeinschaft anerkannt wurden, ohne dass eine zentrale Stelle überhaupt existierte. Ein Beispiel dafür sind die Zigarettenwährungen, die in Kriegsgefangenenlagern während des Zweiten Weltkriegs entstanden[36]. Das bislang spektakulärste und wirkmächtigste dezentrale Geld allerdings

[31] Scherzer (2016).

[32] Die ersten buchbaren Konten entstanden Anfang des 16. Jahrhunderts in Amsterdam, `https://en.wikipedia.org/wiki/Transaction_account#History`

[33] Bofinger (2007):§21.3.

[34] Knapp (1905).

[35] Knapp (1905):S. 140.

[36] Radford (1945).

ist Bitcoin, das Ende 2008 entstand und auf moderner Informationstechnik und dem Internet basiert.

Bitcoin ist ein dezentrales und internetbasiertes Zahlungssystem und gleichzeitig der Name der digitalen Geldeinheit, die ihm zugrunde liegt. Ein Zahlungssystem heißt *dezentral*, wenn es keine zentrale Instanz gibt, über die die Zahlungen und Transaktionen abgewickelt werden. Jeder Teilnehmer von Bitcoin ist Knoten des Bitcoin-Netzwerkes und hat auf seinem Rechner sowohl eine digitale Brieftasche, die „Wallet“, als auch den „Bitcoin Core“ als Client-Software installiert. Der Bitcoin-Core dient dazu, den Teilnehmer über das Internet mit dem Bitcoin-Netzwerk zu verbinden und so Teil einer redundant verteilten Datenbank zu werden, in der alle Transaktionen des gesamten Netzwerks gespeichert und verwaltet werden. Diese verteilte Datenbank heißt *Blockchain*. Der Bitcoin Core hat eine Größe von mehreren Gigabytes, die Speichergröße betrug im März 2017 etwa 100 GB,[37] Anfang 2020 bereits etwa 200 GB.[38] Jeder Teilnehmer hat

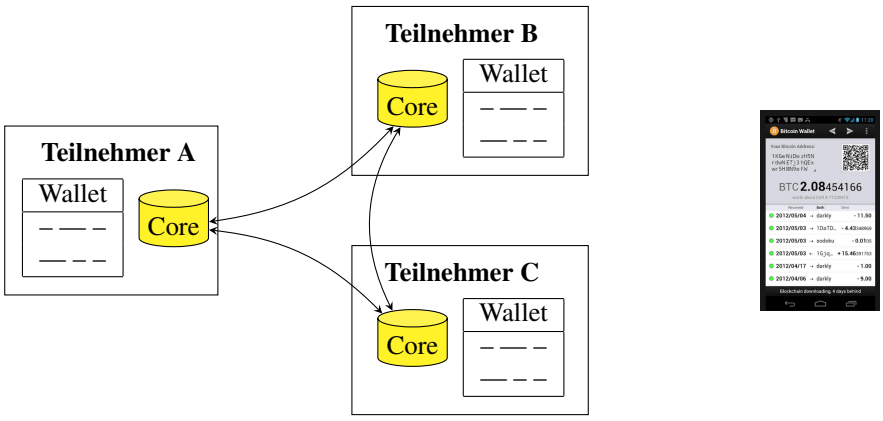


Abbildung 27.3: Architektur des Bitcoin-Netzwerks. Die Bitcoin-Cores bilden eine verteilte Datenbank, die sämtliche Transaktionen speichert und Blockchain heißt. Rechts: Wallet in einem Smartphone.

eine netzwerkweit eindeutige Adresse. Die Wallet ist das private Konto des Teilnehmers und speichert dessen gesamten Zahlungsverkehr sowie das aktuelle Guthaben in der Geldeinheit Bitcoin. Daneben enthält sie ein Paar zusammengehöriger kryptographischer Schlüssel, einen privaten und einen öffentlichen. Der private Schlüssel stellt dabei gewissermaßen die Identität des Teilnehmers dar und muss geheimgehalten werden. Die Schlüssel erlauben im Allgemeinen eine digitale Unterzeichnung der Transaktionen durch ein asymmetrisches Signaturverfahren, bei Bitcoin ist dies ECDSA,[39] welches auf elliptischen Kurven basiert und als sehr sicher gilt.[40] Weitere Details finden sich unter `https://bitcoin.org/en/how-it-works`.

27.4.1 Transaktionen und Mining

Eine Transaktion ist bei Bitcoin eine Überweisung des Betrags x von der Wallet des Teilnehmers A an die Wallet des Teilnehmers B, die mit dem privaten Schlüssel von A gehasht wird. So ist die Transaktion mit A's öffentlichem Schlüssel, der gemeinsam mit dem signiertem Hashwert übertragen wird, entschlüsselbar und damit für jeden verifizierbar. Zusätzlich fügt A der Transaktion den öffentlichen Schlüssel von B hinzu, so dass B nun der neue Besitzer der überwiesenen Bitcoins ist, wie in Abbildung 27.4 illustriert. Die Bitcoin-Adresse wird dann mit den Hash-Funktionen RIPEMD160 und SHA256 sowie einer Kodierung in Base58 berechnet,

[37] Ehmke (2017).

[38] `https://en.wikipedia.org/wiki/Blockchain#History`

[39] `https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm` [2015-06-08]

[40] de Vries (2012a).

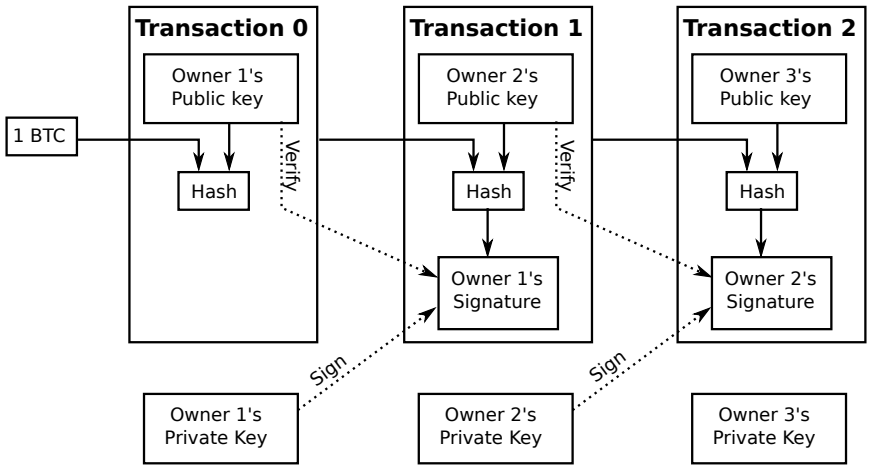


Abbildung 27.4: Transaktionen, die eine Kette der Eigentümer eines Bitcoins (BTC) ergeben. Grafik modifiziert nach Nakamoto (2008).

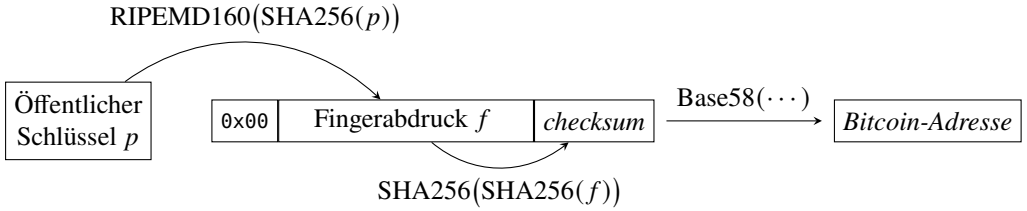
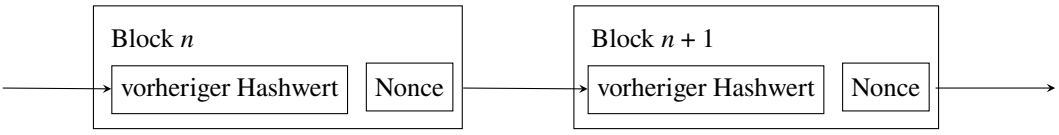


Abbildung 27.5: Die Bitcoin-Adresse wird direkt aus dem öffentlichen Schlüssel des Inhabers berechnet.

wie in Abbildung 27.5 skizziert. Base58 ist ein Zeichensatz aus 58 Zeichen, nämlich den Ziffern 1 bis 9 und den Klein- und Großbuchstaben des ASCII-Codes ohne l (das kleine L), I und O. Mit diesem Zeichensatz werden Verwechslungen von 0 und O sowie von 1, l und I vermieden.[41]

Sämtliche Transaktionen des Netzwerks werden über die Bitcoin Cores in der Blockchain in chronologischer Reihenfolge gespeichert, nachdem sie durch ein verteiltes Konsenssystem bestätigt wurden, dessen Berechnungen *Mining* heißen und mit Bitcoins belohnt werden. Wie nun funktioniert dieses Mining?

Um bestätigt zu werden, müssen eine oder mehrere Transaktionen in einen *Block* gepackt werden, der den Hashwert des vorhergehenden Blocks und eine *Nonce*[42] genannte ganze Zahl enthält. Das geschieht etwa alle 10 Minuten.



Das Nonce ist ein ganzzahliger Wert, der so lange inkrementiert werden muss, bis der vorherige Hashwert mit ihm gehasht einen neuen Hashwert ergibt, der eine vorgegebene Anzahl führender Nullen enthält. Die Berechnung des Nonce genau ist das *Mining*. Die Laufzeit der Nonce-Bestimmung wächst exponentiell mit der Anzahl der vorgegebenen führenden Nullen.[43] Sie ist aber andererseits leicht überprüfbar, denn man braucht sie nur (zusammen mit der Transaktion) in die Hashfunktion einzusetzen. Das Nonce ist somit ein „Proof of work“.[44] Derjenige Miner, der

[41] https://de.wikipedia.org/wiki/Base58

[42] von *nonce word*: ein Wort, das nur zu einer bestimmten Gelegenheit verwendet wird („Gelegenheitsbildung“); *nonce* (englisch, veraltet): Nu, Moment

[43] Nakamoto (2008).

[44] Die Bestimmung des Nonce ist also ein Problem der Klasse NP, d.h. ein Problem, dessen Lösung exponentiellen

die Nonce als erster errechnet hat, erhält Bitcoins ausgezahlt. Will ein Betrüger nun nachträglich einen Block verändern, so muss er diesen und alle nachfolgenden Blöcke neu berechnen. Je weiter also ein Block in der Vergangenheit liegt, umso mehr Rechenzeit muss zur Fälschung einer Blockchain aufgewendet werden und umso schwerer wird eine Fälschung.

Da jede Transaktion an alle Knoten des Netzwerks versendet wird, speichert jeder einzelne Bitcoin Core die vollständige Blockchain. So entscheidet die Mehrheit der Knoten im Netz automatisch, welche Transaktionen gültig sind. Erst wenn ein Betrüger mehr Knoten besitzt

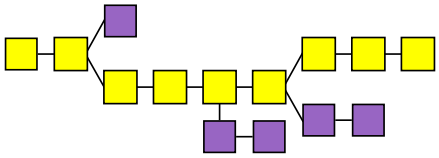


Abbildung 27.6: Schema der Blockchain. Jede Verzweigung stellt einen Betrugsversuch dar, bei dem eine Transaktion gefälscht wurde. Durch das Mining gewinnt stets die Kette, die von den meisten Bitcoin Cores bestätigt werden, hier die hellen Blöcke. Die dunklen Blöcke werden als Fälschung erkannt.

oder unter seine Kontrolle bringt als es „ehrliche“ Knoten gibt, kann er Transaktion erfolgreich fälschen. In dem in Abbildung 27.6 skizzierten Beispiel ist der Betrug in den dunkel markierten Transaktionen zeitweilig gelungen, konnte sich aber gegen die Mehrheit der ehrlichen Knoten nicht nachhaltig durchsetzen.

27.4.2 Historische Entwicklung

Anderthalb Monate nach dem Zusammenbruch der Investmentbank Lehman Brothers und der dadurch ausgelösten globalen Finanzkrise veröffentlichte Satoshi Nakamoto am 1. November 2008 den Forschungsbeitrag *Bitcoin: A Peer-to-Peer Electronic Cash System*.[45] Am 3. Januar 2009 realisierte Nakamoto das Bitcoin-Netzwerk mit einem in C++ programmierten Bitcoin Core und schöpfte die ersten 50 Bitcoins. Damit war das erste dezentrale Digitalgeld der Wirtschaftsgeschichte entstanden. Noch am selben Tag lud der Programmierer Hal Finney den Bitcoin Core herunter und erhielt in der ersten Bitcoin-Transaktion 10 Bitcoins.

In einem Forumsbeitrag vom 11. Februar 2009 erklärte Nakamoto, mit Bitcoin das Wurzelproblem (*root problem*) der konventionellen Währungen gelöst zu haben, deren Wert auf dem Vertrauen in eine Zentralbank beruht. *„The central bank must be trusted not to debase the currency, but the history of fiat currencies is full of breaches of that trust.“*[46] („Der Zentralbank muss vertraut werden, dass sie die Währung nicht entwertet, doch die Geschichte des Fiatgeldes ist voll von Verstößen gegen dieses Vertrauen.“) Mitte 2010 zog sich Nakamoto aus der Programmierentwicklung des Bitcoin-Netzwerks zurück und übergab die Projektleitung an Gavin Andresen.

Als am 25. März 2013 infolge der Finanzkrise Zyperns zweitgrößte Bank geschlossen wurde und das Land unter einen 10 Milliarden Euro betragenden internationalen Rettungsschirm kam, stieg der Bitcoinkurs gegenüber dem Dollar und den Euro stark an.[47]

Rechenaufwand (hier in Abhängigkeit von der Anzahl der führenden Nullen) erfordert, dessen Probe allerdings effizient in polynomieller Laufzeit erfolgen kann (de Vries, 2012b).

[45] Nakamoto (2008); Wallace (2011); Szczepański (2014).

[46] Nakamoto (2009).

[47] `http://money.cnn.com/2013/03/28/investing/bitcoin-cyprus/` [2021-09-21]

27.4.3 Sicherheit

Es werden verschiedene Angriffe auf Bitcoin diskutiert, einige davon wurden auch real durchgeführt.

- *(Race Attack)* Ein Akteur kann versuchen, einen Bitcoin mit zwei fast gleichzeitig erzeugten Transaktionen an zwei verschiedene Akteure auszugeben. Nach den Bitcoinregeln wird dann das Netzwerk nur eine der beiden Transaktionen verifizieren, der von der anderen Transaktion betroffene Bitcoinempfänger wäre um sein Geld betrogen.

- *(Modifikation der Historie)* Grundsätzlich ist es möglich, dass ein Betrüger eine korrekte Transaktion nachträglich ändert, beispielsweise einen ausgegebenen Betrag verringert. Da er seine eigene Transaktion bearbeitet und die modifizierte Version digital signiert, ist die Fälschung grundsätzlich nicht erkennbar. Allerdings wird sie dadurch erschwert, dass der Betrüger alle darauf folgenden Transaktionen des Systems nachberechnen muss, d.h. er muss sich gegen die ehrliche Mehrheit des Netzwerkes durchsetzen. Wenn er beispielsweise $q = 10\%$ der gesamten im Netzwerk verfügbaren Rechenleistung besitzt und wenn $z = 6$ Bestätigungen für die erfolgreiche Anerkennung einer Transaktion ausreichen, so ist mit $\lambda = zq/(1-q)$ die Erfolgswahrscheinlichkeit p_{cheat} dieses Betrugsversuchs

$$p_{\text{cheat}} = 1 - \sum_{k=0}^{z} \frac{\lambda^k \, \mathrm{e}^{-\lambda}}{k!} \left(1 - \left(\frac{q}{1-q}\right)^{z-k}\right) = 0{,}02428\ \%, \qquad (z = 6) \tag{27.9}$$

  siehe Nakamoto (2008:S. 7f). Das ist für große z zwar eine sehr geringe Wahrscheinlichkeit, aber eben auch kein unmögliches Ereignis.

- *(Eyal-Sirer selfish mining attack)*[48] Ein Betrüger findet Blöcke, aber versendet sie nicht weiter im Netzwerk. Stattdessen führt er sein eigenes Mining durch und veröffentlicht seine Blockchain erst, wenn ein anderer Miner seinen eigenen Block gefunden hat. Damit wird das Netzwerk zum Wechsel auf die manipulierte Blockchain gezwungen.

- *(Deanonymisierung)* Grundsätzlich kann mit statistischer Analyse die Verknüpfung von Bitcoin-Adresse und IP-Adresse des zugehörigen Rechners hergestellt werden, was eine Nachverfolgbarkeit von Transaktionen und damit einen Bruch des Versprechens auf anonyme Zahlung darstellt.

27.4.4 Vor- und Nachteile

Dieser Abschnitt basiert wesentlich auf Szczepański (2014). Vorteile bei der Benutzung von Bitcoins:

- *Geringe Transaktionskosten.* Obwohl es zur Zeit keine genauen Untersuchungen darüber gibt, nimmt man allgemein für den Handel mit Bitcoins Transaktionskosten zwischen 0 und 1 Prozent des Betrags an, d.h. Bitcoin ist deutlich billiger als klassische Online-Bezahlsysteme mit 2 bis 3 Prozent.

- *Pseudonymität*. Da Nutzer von Bitcoins nur über ihre verschlüsselte Bitcoin-Adresse identifiziert werden können, ist eine gewisse Anonymität garantiert, da die wahre Identität nicht öffentlich ist. Allerdings können Transaktionsprofile zu einer Bitcoinadresse erstellt werden, die ein Aufspüren der wahren Identität grundsätzlich ermöglichen.

[48] Eyal und Sirer (2013).

- *Kontrollierte Inflation.* Durch die berechenbare konkave und asymptotisch beschränkte Wachstumskurve der Bitcoins kann Bitcoin als eine Art digitales Warengeld keine Inflation erfahren. (Als Vermögenswert jedoch ist es auch Spekulationsobjekt und kann durch Kursverluste dennoch an Wert verlieren, s.u.)

Als Nachteile können erkannt werden:

- *Hohe Kursvolatilität.* Der Kurs von Bitcoin war immer sehr volatil, sehr extrem z. B. im Zeitraum zwischen September 2013 und Frühjahr 2014, wo er zunächst auf das 20fache stieg und danach innerhalb von drei Monaten um ein Drittel absackte. Fast die Hälfte aller Bitcoins gehören 1000 Personen, und nur 47 Personen besitzen ein Drittel aller Bitcoins, was die Gefahr von kartellähnlichen Kursbeeinflussungen impliziert.

- *Geringe Nachfrageelastizität und Deflationsgefahr.* Da die Emission von Bitcoins mit etwa 0,6 Prozent pro Jahr beschränkt ist, unterläge Bitcoin als Währung einer mit mehr als 0,6 Prozent wachsenden Volkswirtschaft einer Deflation.

27.4.5 Ökonomische Bedeutung

Nakamoto bezeichnete Bitcoin als *electronic cash*, also als digitales Geld. Es ist in den meisten Ländern erlaubt, wenn auch oft mit gewissen Einschränkungen. Vollständig verboten ist es (Stand 2014) in Bangladesch, Bolivien, Ecuador, Russland und Vietnam. Die US-amerikanische Steuerbehörde stufte im März 2014 virtuelles Geld wie Bitcoin als Vermögenswerte ein, nicht aber als Währung.[49]

Zusammengefasst zum Abschluss in Form eines Steckbriefs die Auflistung, inwieweit Bitcoin die Funktionen von Geld aus Abschnitt 27.1 auf Seite 163 erfüllt:

Bitcoin als digitales Geld	
Kriterium	**Ermöglichende Mechanismen**
anerkannt	Teilnahme am Netzwerk
verfügbar	Verbindung zum Internet
dauerhaft	verteilte Datenbank Blockchain
tauschbar	asymmetrische Kryptographie (ECDSA)
handlich	Wallet
fälschungssicher	digitale Signatur; Mining und Mehrheitskonsens
zuverlässig	?

[49] `http://www.irs.gov/pub/irs-drop/n-14-21.pdf`

28
Computer und Finanzmärkte

Kapitelübersicht

28.1 Geschichte

19. Oktober 1987: Der Schwarze Montag. Der „Schwarze Montag“ war der erste Börsenkrach nach 1929. Vorausgegangen war ein seit 1982 anhaltendes steiles Wachstum der Aktienmärkte weltweit, das vor allem durch die neoliberale Wirtschaftspolitik der Reagan-Administration in den USA und der Regierung Thatcher in Großbritannien bewirkt wurde. Der Chef der Federal Reserve („Fed“), Alan Greenspan, setzte am 7. September den Diskontsatz[1] um ein halbes Prozent auf sechs Prozent herauf, um einerseits die die Inflationsrate des US-Dollars zu senken und andererseits ein Aufheizen der Börsen zu verhindern. Greenspans Kalkül ging zunächst auch auf, für über einen Monat stabilisierte sich der Dow Jones[2] um die 2600 Punkte (Abbildung 28.1 (a)). Am Mittwoch, den 14. Oktober 1987 allerdings begann ein Sinkflug der Aktienkurse bis zum Wochenende. Zudem fiel am Freitag, den 16. Oktober, der Dollarkurs gegenüber der D-Mark abrupt ab. Die Unsicherheit verstärkte sich durch einen Artikel in der darauf folgenden Sonntagsausgabe der New York Times, in der sich der damalige US-Finanzminister James Baker indirekt gegen eine weitere Stützung des Dollarkurses aussprach. Am Montag, den 19. Oktober, fiel dann der Dow Jones um 22,6 %, ohne einen erkennbaren Grund. Der Sturz wirkte sich schnell auf die Börsen weltweit aus.

Zu dem enormen Ausmaß des Börsenkrachs trug wesentlich die zunehmende Computerisierung des Börsenhandels bei. Seit den frühen 1980er Jahren setzten die Börsenhändler verstärkt

[1] Der Diskontsatz ist der Zins, zu dem die Geschäftsbanken bei der Zentralbank kurzfristig Geld aufnehmen können. Üblicherweise fallen die Börsenkurse, wenn die Zinsen steigen, da die Unternehmen für Investitionskredite höhere Zinsen zahlen müssen und Geld daher eher in festverzinsliche Anleihen als in Aktien angelegt wird.

[2] Der „Dow Jones“, eigentlich *Dow Jones Industrial Average (DJIA)*, ist ein wichtiger US-amerikanischer Börsenindex, der die Kursentwicklung der Aktien von 30 großen US-Unternehmen abbildet und als Indikator für den Zustand des gesamten Aktienmarktes dient (Schaper (2014):S. 128).

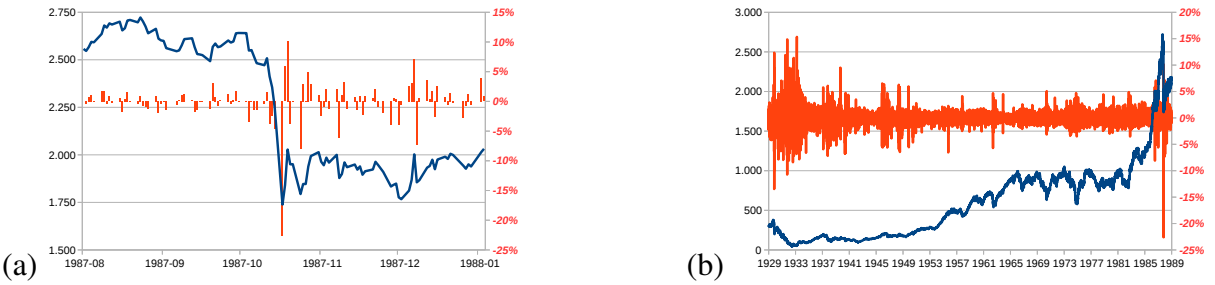


Abbildung 28.1: (a) Der Dow-Jones-Index zwischen August und Dezember 1987. Der Börsenkrach am 19. Oktober 1987 ist deutlich zu erkennen. (b) Die Entwicklung des Dow-Jones-Indexes von 1929 bis 1989 (blaue Kurve) und die relative Änderung zum Vortag (rote Kurve in %). Die Kurve der relativen Änderungen zeigt deutlich die Börsenkräche vom 28./29. Oktober 1929 und vom 19. Oktober 1987. Zu erkennen ist zudem die hohe Kursvolatilität der 1930er Jahre, die sich bis in die 1970er Jahre immer weiter abschwächte und ab etwa 1980 mit den Thatcher-Reagan'schen neoliberalen Reformen und der Einführung des Computerhandels wieder zunahm. In der ruhigen Phase 1950 – 1973 sind die Kurseinbrüche vom 26. September 1955 (−6,54%) und vom 28. Mai 1962 (−5,71%) erkennbar. Quelle: `http://www.cs.princeton.edu/introcs/data/DJIA.csv`

Computer für ihre Portfoliostrategien ein, deren Handelsalgorithmen sich allerdings sehr ähnelten: Fällt der Wert eines Aktiendepots an der Börse, so bestimmen die Algorithmen die exakte Menge an Futures, also börsengehandelten Termingeschäften, um den Verlust auszugleichen. Doch die am Morgen des 19. Oktober 1987 fallenden Aktienkurse lösten plötzlich eine Kaskade aus, denn damit geriet auch die Chicagoer Terminbörse unter Druck, was wiederum die New Yorker Aktienhändler zum Verkauf veranlasste und so den Dow Jones rapide immer weiter senkte. Diese fatale Abwärtsspirale zwischen Computern und Börsenhändlern führte zu einer Resonanzkatastrophe[3].

Um einen Ausfall des Kreditsystems zu verhindern, der wie 1929 eine langanhaltende Rezession der Realwirtschaft zur Folge gehabt hätte, verkündete Greenspan am Morgen des 20. Oktober vor Börsenbeginn, dass die Zentralbank die Liquidität des Finanzsystems aufrechterhalten wolle. Zwar sanken die Kurse am Vormittag zunächst weiter, ab 14 Uhr jedoch erholten sie sich und schlossen zum Börsenende mit einem Plus ab. Der Fed war es also gelungen, den durch Computer mitausgelösten Schock in weniger als 24 Stunden aufzufangen[4]. Aus einer wirtschaftshistorischen Perspektive betrachtet handelte es sich mit Greenspans Erklärung des 20. Oktober 1987 um einen Paradigmenwechsel der Fed, nach dem nicht mehr explizit die Bekämpfung der Inflation als ihr oberstes Ziel galt, sondern die Bekämpfung einer Rezession durch die Garantie der Liquidität des Bankensystems. Dieses Paradigma erstickte in den folgenden zwei Jahrzehnten jeden kleineren Crash und ließ das Finanzsystem weltweit stark wachsen. Begünstigt wurde diese Entwicklung durch immer weitere Lockerungen der Bankenregulierungen und durch die digitale Revolution des Finanzsektors, die immer mehr und immer komplexere Finanzprodukte hervorbrachte und schließlich zu der Finanzkrise 2008 führte.[5].

15. September 2008: Die globale Finanzkrise 2008. Die Investmentbank Lehman Brothers meldete am 15. September 2008 Insolvenz an. Dadurch wurde die seit 1929 schwerste globale Finanz- und Bankenkrise ausgelöst. Vorausgegangen war die sogenannte Subprime-Krise, der systematische Ausfall von nicht ausreichend gedeckten Immobilienkrediten in den USA. In der Folge kam es zu einer schweren Bankenkrise, die allein im Jahr 2008 weltweit mindestens 83 Banken in den Konkurs, eine Übernahme oder die Verstaatlichung trieb[6]. Dadurch wiederum

[3] Schaper (2014):S.128ff.
[4] Schaper (2014):S.130.
[5] Schaper (2014):S.131.
[6] Sinn (2009):S. 63ff.

kam es weltweit zu einer starken Erhöhung der Staatsschulden, deren Auswirkungen noch bis in das nächste Jahrzehnt ausstrahlten. (Insbesondere die Eurokrise ab 2010 wurde zu einem großen Anteil durch die Bankenkrise ausgelöst.)

Computer waren zwar nicht Hauptursache dieser Finanzkrise, aber ohne sie wäre die auslösende Subprimekrise aufgrund der extrem hohen Komplexität ihrer strukturierten Finanzinstrumente nicht in diesem Umfang möglich gewesen[7]. Aber auch die Folgen der Finanzkrise wurden durch Computer beeinflusst, denn die Zentralbanken konnten durch digitale Geldschöpfung einen offenen Zusammenbruch der Weltwirtschaft wie 1929 verhindern[8]. Um den Einfluss der Computerisierung auf die Subprimekrise zu verstehen, muss man zunächst die Ausmaße des sich seit den 1990er Jahren rasant steigenden Kontraktvolumen der derivativen Finanzinstrumente, kurz Derivate genannt, betrachten. Derivate werden in standardisierter Form (z. B. Futures an der Börse) oder direkt zwischen den Vertragspartnern Over-the-Counter (OTC) gehandelt. Die weltweit wichtigsten Börsen für den organisierten Derivatehandel sind die deutsch-schweizerische EUREX, die britische International Financial Futures Exchange (Liffe), sowie die US-amerikanischen Finanz- und Warenterminbörsen Chicago Board of Trade (CBoT) und Chicago Mercantile Exchange (CME). Ende 2007 verwalteten Hedgefonds 2,7 Billionen Dollar[9]. Es wird geschätzt, dass Ende 2003 auf den OTC-Märkten knapp 200 Billionen Dollar in Umlauf waren; allein OTC wurden 2008 täglich im Nominalwert von etwa 2 Billionen Dollar umgesetzt, das Handelsvolumen auf dem börslichen Derivatemarkt lag bei 6 Billionen Dollar am Tag[10]. Im Juni 2008 betrug das nominale Kontraktvolumen aller offenen OTC Derivate 684 Billionen US$,[11] was etwa dem Zehnfachen des Bruttoinlandsprodukts (BIP) der gesamten Welt betrug,[12] siehe Abbildung 28.2. Kern und Auslöser der Subprimekrise waren spezielle Derivate,

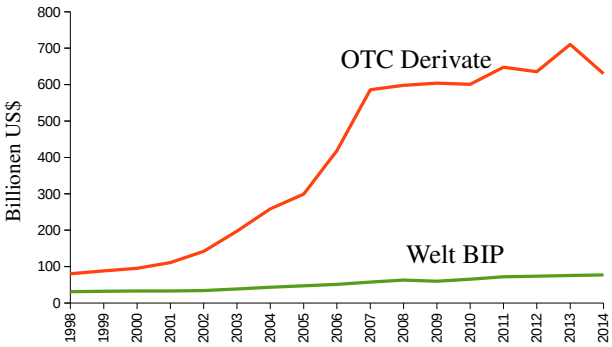


Abbildung 28.2: Gesamtsumme des nominalen Kontraktvolumens offenstehender OTC Derivate im Vergleich zum nominalen Bruttoinlandsprodukt der gesamten Welt. Quellen: BIS,[11] IMF.[12]

die *Mortgage-backed Securities (MBS)*, festverzinsliche hypothekenbesicherte Wertpapiere, die minderwertige Kredite („Subprimes“) mit Krediten hoher Bonität kombinierten. Die Käufer der MBS-Papiere waren typischerweise international tätige Geschäfts- und Investmentbanken, die ihrerseits die bereits verbrieften Ansprüche erneut verbrieften. Sie definierten dazu sogenannte Zweckgesellschaften *(Special Purpose Vehicles, SPV)*, eigene kleine Unternehmungen, meist in

[7] Lanier (2014):S. 60.

[8] Bauer et al. (2015):S. 21.

[9] Wagenknecht (2008):p.149.

[10] Wagenknecht (2008):p.172.

[11] BIS `http://www.bis.org/statistics/derstats.htm`. Das nominale Kontraktvolumen ist alleine zwar insofern wenig aussagekräftig, als sich das von Marktteilnehmern eingegangene Risiko nach deren Netto-Positionen bemisst, die sich im Bereich von wenigen Prozent der aggregierten Kontraktvolumen bewegen. Die Entwicklung des insgesamt ausstehenden Kontraktvolumens vermittelt jedoch einen Eindruck über die hohe Dynamik auf dem Markt für Finanzderivate.

[12] `http://www.imf.org/external/pubs/ft/weo/2015/01/weodata/WEOApr2015alla.xls`

der Rechtsform einer gemeinnützigen Stiftung mit geringem Eigenkapital mit einer von der Muttergesellschaft ausgestellten Garantie gegen Verluste. Durch einen „Strukturierung“ genannten Prozess schufen sie verschiedene Risikoklassen oder Tranchen und verbrieften sie zu besicherten Schuldverschreibungen namens *Collateralized Debt Obligations (CDO)*. Die beste Tranche war dabei die Senior-Debt-Tranche oder AAA-Tranche, die von den Zweckgesellschaften stets als Erstes zu den vereinbarten Konditionen bedient wurde. Die nächstbesten Tranchen waren die Mezzanine-A- und Mezzanine-B-Tranchen, erst ganz zum Schluss sollte die schlechteste aller Tranchen bedient werden, die Equity- oder Eigenkapitaltranche. Die CDO-Papiere wurden gemäß der Risikostufen ihrer jeweiligen Tranchen mit unterschiedlichen Renditen versehen, die Eigenkapitaltranche aufgrund des Risikos mit der höchsten Rendite. Die CDO-Papier konnten wiederum selbst verbrieft werden zu CDOs der zweiten Generation, auch CDO^2 genannt (Abb. 28.3). Im Prinzip gab es keine Begrenzung der Anzahl der Verbriefungsstufen, bis zu

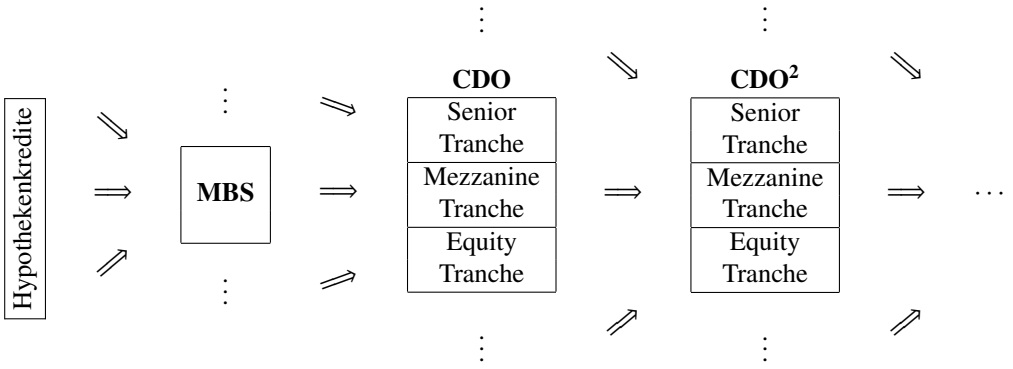
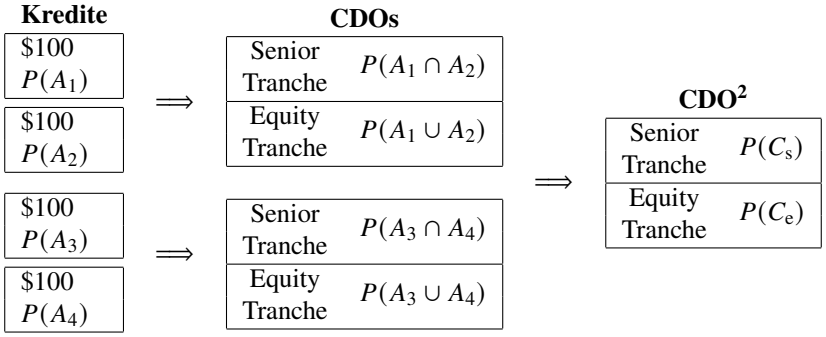


Abbildung 28.3: Struktur der CDOs. Nach[13].

sechs Verbriefungsstufen waren üblich[14].

Durch die Verbriefungsfolge konnte somit ein Ursprungsportfolio von Hypothekenkrediten mittlerer Qualität scheinbar in ein Portfolio mit Anteilen von Aktiva hoher Bonität und einem kleineren Anteil von Aktiva mit hohem Ausfallrisiko umgewandelt werden. Allerdings lag den Berechnungen dieser Ausfallrisiken eine empirisch falsche Annahme zugrunde, nämlich die Unabhängigkeit der einzelnen Ausfallereignisse.

Beispiel 28.1. [15] Es gebe zunächst zwei Portfolios mit je zwei Hypothekenkrediten gleichen Nennwerts (\$100) und gleichen Ausfallwahrscheinlichkeiten $P(A_j)$, wobei A_j für $j = 1, 2, 3, 4$ den Ausfall des Kredits j bezeichnet. In der ersten Verbriefungsstufe wird jedes Portfolio in eine Senior- und eine Eigenkapitaltranche unterteilt. Die Seniortranche ist so definiert, dass nur dann ein Verlust anfällt, wenn beide zugrunde liegenden Hypothekenkredite ausfallen, während die Eigenkapitaltranche bereits bei dem Verlust eines der beiden Kredite ausfällt.



(28.1)

[14] Sinn (2009):S. 131ff.

[15] Sinn (2009):S. 145f.

Hierbei ist $C_\mathrm{s} = \bigcap_j (A_{2j-1} \cup A_{2j})$ das Ereignis, dass beide Eigenkapitaltranchen der CDO-Papiere der ersten Stufe ausfallen, und $C_\mathrm{e} = \bigcup_j A_j$ den Ausfall nur einer von beiden. Sind die Kreditausfallereignisse unabhängig[16], so ergibt sich für die Ausfallwahrscheinlichkeit $P(A_j \cap A_k)$ der Seniortranche der ersten CDO-Papiers der Wert $P(A_1 \cap A_2) = P(A_1)P(A_2)$, und für diejenige der Eigenkapitaltranche nach der Additionsformel $P(A_1 \cup A_2) = P(A_1) + P(A_2) - P(A_1)P(A_2)$. Entsprechend folgt für das zweite CDO-Papier $P(A_3 \cap A_4) = P(A_3)P(A_4)$ und $P(A_3 \cup A_4) = P(A_3) + P(A_4) - P(A_3)P(A_4)$.

$$\begin{array}{|c|} \hline P(A_1) \\ \hline P(A_1) \\ \hline \end{array} \implies \begin{array}{|l|} \hline P(A_1 \cap A_2) = P(A_1)P(A_2) \\ \hline P(A_1 \cup P(A_2) = P(A_1) + P(A_2) - P(A_1)P(A_2) \\ \hline \end{array} \implies \begin{array}{|c|} \hline P(C_\mathrm{s}) \\ \hline P(C_\mathrm{s}) \\ \hline \end{array} \qquad (28.2)$$
$$\begin{array}{|c|} \hline P(A_1) \\ \hline P(A_1) \\ \hline \end{array} \implies \begin{array}{|l|} \hline P(A_3 \cap P(A_4) = P(A_3)P(A_4) \\ \hline P(A_3 \cup P(A_4) = P(A_3) + P(A_4) - P(A_3)P(A_4) \\ \hline \end{array}$$

mit $P(C_\mathrm{s}) = P(A_1 \cup A_2)P(A_3 \cup A_4)$ und $P(C_\mathrm{e}) = P(A_1 \cup A_2)P(A_1 \cup A_2)$. Speziell für $P(A_j) = 0{,}1$ für alle $j = 1, 2, 3, 4$ ergibt dies

$$\begin{array}{|c|} \hline P(A_1) = 0{,}1 \\ \hline P(A_2) = 0{,}1 \\ \hline \end{array} \implies \begin{array}{|l|} \hline P(A_1 \cap A_2) = 0{,}01 \\ \hline P(A_1 \cup A_2) = 0{,}19 \\ \hline \end{array} \implies \begin{array}{|l|} \hline P(C_\mathrm{s}) = 0{,}04 \\ \hline P(C_\mathrm{e}) = 0{,}34 \\ \hline \end{array} \qquad (28.3)$$
$$\begin{array}{|c|} \hline P(A_3) = 0{,}1 \\ \hline P(A_4) = 0{,}1 \\ \hline \end{array} \implies \begin{array}{|l|} \hline P(A_3 \cap A_4) = 0{,}01 \\ \hline P(A_3 \cup A_4) = 0{,}19 \\ \hline \end{array}$$

In dem extremen Fall einer perfekten Korrelation aller Kreditausfälle („systemische Krise“) allerdings, wenn also die bedingte Wahrscheinlichkeit $P(A_j|A_k) = 1$ lautet, wobei A_j den Ausfall des Kredits j bezeichnet, folgt

$$P(A_{2j-1} \cup A_{2j}) = P(A_{2j-1} \cap A_{2j}) = P(C_\mathrm{s}) = P(C_\mathrm{e}) = 0{,}1$$

für $j = 1, 2$. In diesem Falle enthält also *keines* der CDO-Papiere eine Tranche mit geringem Ausfallrisiko. Im Allgemeinen führt also die Annahme der stochastischen Unabhängigkeit der zugrunde liegenden Kreditausfälle zu einer möglicherweise katastrophalen Fehleinschätzung des Risikos der strukturierten Finanzprodukte. □

Im Jahr 2007, unter anderem nach Zinserhöhungen der Zentralbank zur Abwehrung der Inflationsgefahr nach der Konjunkturkrise nach dem Anschlag auf das World Trade Center 2001, stoppten die Banken die Vergabe neuer und Verlängerung bestehender Kredite an problematische Kunden, was unter anderem zur Folge hatte, dass die Häuserpreise sanken und Zwangsversteigerungen zunahmen. Im Sommer 2007 wurden die CDO-Papiere dann von den Rating-Agenturen drastisch in ihrer Risikobewertung herabgestuft, so dass im August 2007 sogar kurzfristig der Interbankenhandel zusammenbrach und die Emissionen der Papiere im dritten Quartal 2007 schlagartig zurückgingen[17].

Spätestens jetzt war es mit der stochastischen Unabhängigkeit der Kreditausfälle vorbei, systematisch brach die Kreditfinanzierung des US-Immobilienmarktes ein. Die Immobilienblase war damit geplatzt und weitete sich im Jahre 2008 zu einem weltweiten Bankensterben aus, von dem mindestens 83 Banken durch Konkurs, Übernahme oder Verstaatlichung betroffen waren[18]. Da die verantwortlichen Managerfür die Krise in dem Bewusstsein handelten, dass ihre Institutionen „too big to fail“ waren und daher von der Allgemeinheit gerettet werden würden, handelte es sich um einen der schwerwiegendsten Fälle von *moralischem Risiko* oder *Moral Hazard*[19],[20], siehe auch Seite 128.

[16] Bandelow (1989):§11.1.
[17] Sinn (2009):S. 137.
[18] Sinn (2009):S. 63ff.
[19] Bofinger (2007):S. 256.
[20] Lanier (2014):S. 88, 353f.

6. Mai 2010: Der Flash-Crash. „Am frühen Nachmittag des 6. Mai 2010 begann der wichtigste Aktienindex der USA, der Dow Jones Industrial Average (DJIA), plötzlich zu fallen. Es gab keinen plausiblen Anlass für diesen Absturz, weder alarmierende Nachrichten noch neue Wirtschaftsdaten. Dennoch fiel der Dow Jones, der seit Börsenöffnung nur langsam nachgegeben hatte, binnen Minuten um volle 6 Prozent. Es war das reinste Chaos: Einige Aktien wurden zum Preis von 1 Cent gehandelt, andere für exorbitante 100 000 Dollar, beides ohne erkennbaren Grund. In einer Viertelstunde summierten sich die Kursverluste auf rund 1 Billion Dollar. [. . .] Doch das eigentlich Bizarre und Einmalige an diesem 6. Mai 2010 war, was nach dem Absturz geschah: Die Kurse erholten sich genauso schnell, wie sie abgeschmiert waren. Nach 20 Minuten Höllentrip war der Dow Jones auf sein Ausgangsniveau zurückgekehrt. [. . .] Die Episode wurde bekannt als der „Flash Crash“. Als Grund für diesen Blitzcrash ermittelte die US-Börsenaufsicht SEC (Securities and Exchange Commission) in ihrem offiziellen Bericht eine einzige schlecht getimte Aktientransaktion von ungewöhnlich großem Volumen. Aber diese Erklärung konnte kundige Beobachter nicht überzeugen. Viele Börsenexperten gaben die Schuld vielmehr jener neuartigen Finanztechnologie, die als Hochfrequenzhandel (HFH) oder ‚Flash Trading‘ bezeichnet wird.“[21],[22].

28.2 Algorithmischer Handel

Nach dem Werpapierhandelsgesetz[23] bezeichnet *algorithmischer Handel* allgemein den Handel mit Finanzinstrumenten, bei dem ein Computeralgorithmus die einzelnen Auftragsparameter wie Zeitpunkt, Preis oder Quantität des Auftrags automatisch bestimmt. Dabei kann der Algorithmus grundsätzlich deterministisch sein, also bei gegebenen Bedingungen eine eindeutige und wiederholbare Entscheidung treffen, oder stochastisch, also bei gegebenen Bedingungen eine bis zu einem gewissen Grad durch den Zufall beeinflusste Entscheidung treffen.

28.2.1 Zweiseitige Auktion und Wilson-Algorithmus

Ein recht gut verstandenes und weithin angewendetes Marktdesign ist die *zweiseitige Auktion* (*double auction*), siehe[24], für eine theoretische Einführung auch[25]. Eine zweiseitige Auktion kann als ein Spiel charakterisiert werden, in dem jeder Spieler eine feste Rolle als Nachfrager oder als Anbieter hat. Dabei gibt ein Nachfrager nur Kaufgebote (*bids*), ein Anbieter nur Verkaufsgebote (*asks*). Nachfrager und Anbieter einer Ware oder eines Wertpapiers machen dabei ihre jeweiligen Gebote alle gleichzeitig, und ein Auktionator oder „Marketmaker“ bestimmt daraufhin einen markträumenden Preis p_* (*clearing price*): Das bedeutet, alle Anbieter mit Verkaufsgeboten kleiner als p_* verkaufen, alle Nachfrager mit Kaufgeboten größer als p_* kaufen, und die Gesamtzahl der gekauften Einheiten der Ware bzw. des Wertpapiers entspricht jeweils derjenigen der verkauften Einheiten.

Wie kann dieser markträumende Preis berechnet werden? Ein gängiger Algorithmus dafür geht zurück auf Wilson[26]. Zur Verdeutlichung der zugrunde liegenden Idee anhand eines einfachen Beispiels siehe auch[27].

---

[21] Lanchester (2014).
[22] siehe auch Lewis (2014):S. 82.
[23] *Wertpapierhandelsgesetz* (o. D.):§ 33 Abs. 1a WpHG.
[24] Wilson (1985); Satterthwaite und S. R. Williams (1989); Das, J. E. Hanson et al. (2001).
[25] Fudenberg und Tirole (1991):§6.5, §7.4.5.
[26] Wilson (1985).
[27] Bofinger (2007):§2.2.

Der Wilson-Algorithmus

Gegeben seien n_s Anbieter (*sellers*), $i = 1, \ldots, n_\mathrm{s}$, die jeweils genau eine Einheit des Guts anbieten, und n_b Nachfrager (*buyers*), $j = 1, \ldots, n_\mathrm{b}$, die jeweils genau eine Einheit des Guts nachfragen. Seien ferner die Angebotspreise $c \in [\underline{c}, \overline{c}]$ und die Gebote $v \in [\underline{v}, \overline{v}]$ der Nachfrager stochastisch unabhängige Zufallsgrößen, wobei $\underline{c} \leqq \underline{v} < \overline{c} \leqq \overline{v}$, so dass die beiden Fälle $v > c$ und $v < c$ positive Wahrscheinlichkeiten haben. Auf einem Finanzmarkt sind die Angebotspreise die Limits der Verkaufsordern, und die Gebote die Limits von Kaufordern. Sei nun jeweils der Angebotspreis von Anbieter i mit c_i und das Gebot von Nachfrager j mit v_j bezeichnet. Dann besteht der Wilson-Algorithmus aus den folgenden zwei Schritten.

Schritt 1. Nummeriere die Akteure um (bzw. sortiere die Angebote und Gebote), so dass

$$c_1 \leqq c_2 \leqq \cdots \leqq c_{n_\mathrm{s}}, \qquad v_1 \geqq v_2 \geqq \cdots \geqq v_{n_\mathrm{b}} \tag{28.4}$$

gilt.

Schritt 2. Bestimme die Anzahl der in einer zweiseitigen Auktion gehandelten Einheiten als das größte k, so dass $v_k \geqq c_k$. Dann verkaufen die Anbieter 1 bis k genau an die Nachfrager 1 bis k. Der markträumende Preis p_* (an einer Börse auch „Spotpreis“ genannt) ist dann ein

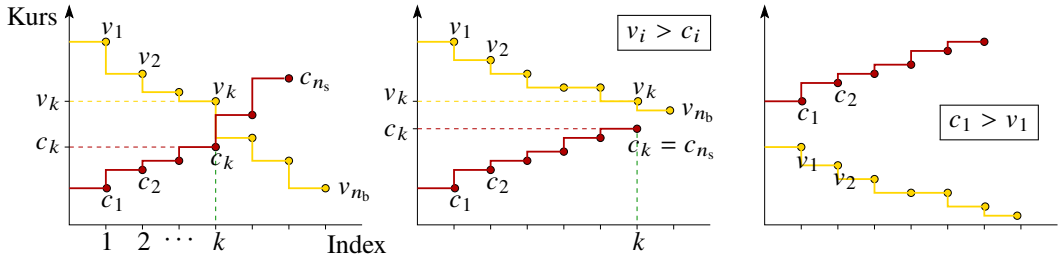


Abbildung 28.4: Der Wilson-Algorithmus zur Bestimmung des optimalen Preisintervalls $[c_k, v_k]$, wo k der größte Index mit $v_k \geqq c_k$ für die sortierten Angebote und Gebote (28.4) ist. Dargestellt sind die drei möglichen Fälle (i) $c_1 \leqq v_1$ und $c_{k+1} > v_{k+1}$, (ii) $v_i > c_i$ für alle $i \leqq \min(n_\mathrm{s}, n_\mathrm{b})$, und (iii) $v_i > c_i$ für alle $i \leqq \min(n_\mathrm{s}, n_\mathrm{b})$.

beliebiger Wert aus dem Interval $[c_k, v_k]$, zum Beispiel der Mittelwert $p_* = (c_k + v_k)/2$.

Für die übrigen Anbieter und Nachfrager kommt kein Handel zustande. Wilson[28] bewies, dass unter einigen milden Voraussetzungen[29] eine zweiseitige Auktion für die Grenzwerte $n_\mathrm{s} \to \infty$ und $n_\mathrm{b} \to \infty$ effizient ein Gleichgewicht liefert.

CDA

Ein Modell, das die zweiseitige Auktion verallgemeinert, ist die *kontinuierliche zweiseitige Auktion* (CDA, *continuous double auction*). Sie stellt den Mechanismus dar, nach dem Aktien auf Börsen wie NASDAQ und NYSE gehandelt werden[30]. Sie erfüllt die Regeln der US-Börsenaufsicht SEC für ein elektronisches Handelssystem.[31] Bei einer CDA gibt es mehrere Handelsperioden gleicher Dauer, in der jeweils Gebote (Kauforders) und Angebote (Verkaufsorders) platziert werden können. Falls zu einem Zeitpunkt offene Gebote und Angebote in Preis und Menge übereinstimmen, wird der Handel sofort ausgeführt. Typischerweise wird eine Ankündigung an alle Auktionsteilnehmer gesendet, wenn Orders platziert oder Handel ausgeführt wurden. Am Ende jeder Handelsperiode wird den Teilnehmern eine Liste von „Limitpreisen“ geschickt, d.h. von Geboten der Nachfrager und Angebotspreisen der Anbieter, jeweils pro Einheit des Gutes.

[28] Wilson (1985).

[29] Existenz eines Gleichgewichts symmetrischer Kaufs- und Verkaufsstrategien, die differenzierbare Funktionen der privaten Information sind und uniform beschränkte Ableitungen besitzt.

[30] Das, J. Hanson et al. (2002).

[31] `http://sec.gov/divisions/marketreg/mrecn.shtml` [2015-06-25]

Das rationale Ziel eines jeden Teilnehmers ist die Maximierung des „Gewinns“, definiert als die Differenz „Limitpreis − gehandelter Preis“ für Nachfrager und die Differenz „gehandelter Preis − Limitpreis“ für Anbieter.

28.2.2 Hochfrequenzhandel

> *The U.S. stock market was now a class system, rooted in speed, of haves and have-nots. The haves paid for nanoseconds; the have-nots had no idea that a nanosecond has a value.*
>
> Michael Lewis

Das Wertpapierhandelsgesetz[32] definiert *Hochfrequenzhandel* als den Kauf und Verkauf von Finanzinstrumenten (Wertpapieren) eines heimischen und organisierten Marktes oder eines multilateralen Handelssystems mittels einer hochfrequenten algorithmischen Handelstechnik, die gekennzeichnet ist durch

- die Nutzung von Infrastrukturen, die darauf abzielen, Latenzen zu minimieren;
- die Entscheidung des Systems über Einleitung, Erzeugung, Weiterleitung oder Ausführung eines Auftrags ganz ohne menschliche Intervention für einzelne Geschäfte oder Aufträge;
- ein hohes untertägiges Mitteilungsaufkommen in Form von Aufträgen, Quotes oder Stornierungen.

Es geht hier also um die Verbindung dreier verschiedener Wissenschaftsdisziplinen, Physik, Informatik und Ökonomie: Der erste Punkt bezieht sich auf die *Physik* der Informationsübermittlung, wir werden ihn weiter unten näher betrachten. Der zweite Punkt betrifft die *Informatik* der Handelsabwicklung, und der dritte die *Ökonomie* der eigentlichen Aktivität des Börsenhandels.

Der Hochfrequenzhandel entwickelte sich nach der Finanzkrise 2008. Er basiert hauptsächlich darauf, dass große Wertpapierorders nicht an einer einzigen Börse abgewickelt, sondern die gewünschten Stückzahlen auf mehrere Börsen verteilt gehandelt werden. Ein Broker sendet seine Aufträge also gleichzeitig an mehrere Handelsplätze, die nach seinen Marktinformationen zum gegebenen Zeitpunkt insgesamt in der Lage sind, den Gesamtauftrag abzuwickeln. Die

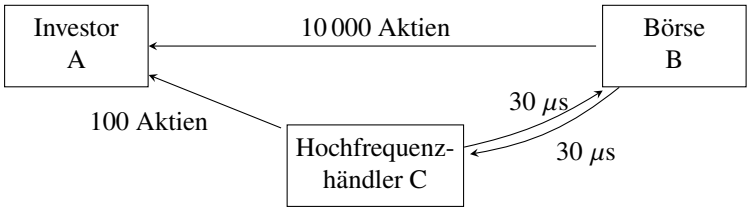


Abbildung 28.5: Prinzip des *Frontrunnings* durch Hochfrequenzhandel. Investor A platziert eine Order an Börse B, z.B. den Verkauf von 1 Mio Aktien. Noch während die Order als Signal zur Börse unterwegs ist, bekommt ein Hochfrequenzhändler C die Information über diese Order und platziert eine entsprechende Order, also den Verkauf von, sagen wir, 10 000 Aktien (obere Abbildung). Etwas später trifft die Order von A ein und der Preis sinkt. Sofort kauft der Hochfrequenzhändler seine Aktien billig zurück. Entsprechend kann C bei einem Kauf von Aktien durch A gewinnen. Wesentlich ist der Mechanismus, mit dem C schnell genug an die Information des Auftrags von A gelangt. Üblicherweise ist C an einer der Börsen beteiligt, die diese Information bekommt.

Teilaufträge kommen aber aufgrund der endlichen Ausbreitungsgeschwindigkeit der elektronischen Signale zu unterschiedlichen Zeiten bei den Börsen an. Ein Hochfrequenzhändler, der

[32] *Wertpapierhandelsgesetz* (o. D.):§ 2 Abs. 3 S. 2d WpHG.

einen schnellen Zugang zu Marktdaten der nächstgelegenen Börse hat, kann darauf spekulieren, dass zu dem von ihm gesehenen Teilauftrag weitere Teilaufträge mit Kauf- oder Verkauforders derselben Aktie oder Anleihe gehören. Wenn er zudem schnellere Datenleitungen zu weiter entfernt liegenden Börsen hat, kann er an diese Orders absenden, die dort den Teilorders des ersten Brokers zuvorkommen. Bei hohen Handelsvolumina ändert sich dadurch der Kurs an allen Börsen, so dass der Hochfrequenzhändler die später eintreffenden Aufträge des ursprünglichen Kunden aus den Papieren bedienen kann, die er im Vorlauf erwerben konnte. Allerdings ist der Preis nun ungünstiger für den ursprünglichen Kunden. Der Hochfrequenzhändler fährt so seine Position wieder auf Null. Durch den Unterschied zwischen den Kursen zieht er einen winzigen Gewinn pro Geschäft. Die Erlöse durch diesen Trick, das sogenannte *Frontrunning*[33], sind pro Transaktion zwar gering, können sich jedoch insgesamt zu großen Erträgen aufsummieren. Allerdings lohnt sich das Frontrunning umso mehr, je größer die Order. Daher sind fast ausschließlich Großinvestoren, also Banken und Hedgefonds, vom Frontrunning betroffen[34].

Physik des Hochfrequenzhandels

Physikalische Basis des Hochfrequenzhandels ist die endliche Signalgeschwindigkeit in Kabeln. In einem Koaxial- oder einem Glasfaserkabel gilt für die Signalgeschwindigkeit $v_\mathrm{s} \approx \frac{2}{3}c$, wo c die Lichtgeschwindigkeit ist. Genauer gilt für die Signalausbreitungsgeschindigkeit[35],

$$v_\mathrm{s} = \begin{cases} 2,3 \cdot 10^8 \text{ m/s} = 230 \text{ m}/\mu\text{s} = 230 \text{ km/ms} & \text{in Kupferkabeln,} \\ 2 \cdot 10^8 \text{ m/s} \quad = 200 \text{ m}/\mu\text{s} = 200 \text{ km/ms} & \text{in Glasfasern.} \end{cases} \tag{28.5}$$

(Die jeweils zweiten und dritten Gleichungen sind in Einheiten angegeben, mit denen man im Zusammenhang mit Netzwerken besser rechnen kann.) Die dadurch bewirkte Zeitverzögerung zwischen dem Absenden und dem Empfang eines Signals heißt *Latenz* oder *Signallaufzeit* (englisch *latency* oder *transmission delay*). Hat das Kabel zwischen Sende- und Empfangsort

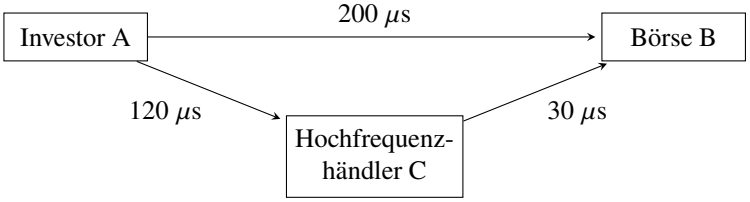


Abbildung 28.6: Latenzen im Netzwerk der Börsen.

die Länge s, so ist die Latenz t_s des Signals gegeben durch $t_\mathrm{s} = s/v_\mathrm{s}$. Entscheidender für die Übertragung von Informationen ist dagegen der *Datendurchsatz (goodput)*, also die Menge der Nutzdaten in Bits, die effektiv übertragen werden[36]. Entsprechend ist die *Datendurchsatzrate* in MBit/s definiert als der Datendurchsatz pro Zeiteinheit.

Beispiel 28.2. Die Regulierungbehörde der USA ermittelt über einen zentralen Server SIP den Marktpreis einer Aktie für alle 13 angeschlossenen Börsen. Dazu schicken die Börsen ihre Marktdaten an den SIP. Nun kann aber ein Hochfrequenzhändler mit einer schnelleren Kopieversion des SIP den Börsenpreis eher ermitteln, als er offiziell verkündet wird. Dadurch erhält er typischerweise einen Zeitvorsprung von 25 ms[37]. Ein Signal auf der Hochgeschwindigkeitsverbindung zwischen New York und Chicago benötigt etwa 6 ms, d.h. er kann genau zwei Orders

[33] Lewis (2014):S. 61.
[34] Salmon (2014).
[35] Broy und Spaniol (1999); Stein (2001):S. 657 / S. 173.
[36] Stein (2001):S. 172f.
[37] Lewis (2014):S. 111.

zwischen den beiden Börsen (mit jeweils notwendiger Bestätigungsquittung) platzieren, bevor der Preis offiziell festgelegt ist. □

Entstehungsgeschichte

Im Jahr 2005 verlangte die US-amerikanische Börsenaufsicht nach einer Reihe von Korruptionsfällen, dass die Handelsplätze der Börsen in Aktiengesellschaften umgewandelt werden mussten. Bis dahin wurde der Aktienhandel von den Börsenplätzen selber und wenigen elektronischen Börsen durchgeführt, so beispielsweise an der Wall Street von der New York Stock Exchange (NYSE) zu 85 % und von der NASDAQ zu 15 %. In der Folge entstanden ab 2005 viele dezentrale private Handelsplätze, allein im Umfeld der Wall Street 13 Börsen bis Anfang 2008. Gleichzeitig veränderten die Börsen ihr Geschäftsmodell. Kassierten die NYSE und die NASDAQ vormals von allen Händlern eine feste Kommissionsgebühr, entstand nun ein komplexes System von Gebühren und Gutschriften, in dem sich die einzelnen Börsen differenzierten und so in Wettbewerb traten.[38] Eine der Börsen mit den attraktivsten Gutschriften war die BATS in Weehawken, New Jersey.[39] Das auf Hochfrequenzhandel spezialisierte Finanzdienstleistungsunternehmen Getco wickelte bis zu 10 % des US-amerikanischen Aktienhandels ab.[40]

Mitte der 2000er herrschte allgemein die Ansicht vor, dass der Hochfrequenzhandel die Liquidität[41] der Finanzmärkte erhöhte und damit positiv für die Investoren der Realwirtschaft wären[42]. Insgesamt begünstigte die Neuregulierung der Finanzmärkte ab 2005 den Hochfrequenzhandel so sehr, dass er sich von 2006 bis 2008 von 26 auf 52 Prozent Marktanteil verdoppelte und seitdem auf diesem Niveau verharrte[43], während die Kursschwankungen je Handelstag sich seit 2010 deutlich um 40 Prozent erhöhten[44].

Im April 2016 beurteilte die britische Finanzaufsicht FCA in einer Untersuchung dagegen die Auswirkungen des Hochfrequenzhandels als nicht negativ auf das Finanzmarktgeschehen. Es konnten keine Belege entdeckt werden, dass Hochfrequenzhändler in Großbritannien Geschwindigkeitsvorteile zum Nachteil anderer Anleger ausnutzen können. Bereits ein Jahr zuvor hat die Bank of England eine Studie veröffentlicht, in der die Ansicht vertreten wurde, Hochfrequenzhandel trage dazu bei, dass Marktpreise genauer ausfielen und mache die Märkte somit effizienter.[45]

---

[38] Ein wichtiges Geschäftsmodell für Gebühren und Gutschriften ist das „Maker-Taker-Modell". Eine Aktie hat üblicherweise einen Geldkurs und einen Briefkurs. Der *Geldkurs* (englisch *bid*) ist der Höchstpreis, zu dem ein Nachfrager bereit ist, ein gegebenes Finanzprodukt zu kaufen. Der *Briefkurs* (englisch *ask* oder *offer*) ist demgegenüber der Tiefstpreis, zu dem ein Anbieter bereit ist zu verkaufen. Der Geldkurs liegt in der Regel über dem Briefkurs, d.h. die *Geld-Brief-Spanne* oder der *Spread*, also die Differenz Geld minus Brief, ist positiv. Ein Nachfrager ist ein *Taker* (Preisnehmer), der den Geldkurs als Preis akzeptiert und kauft, und ein *Maker*, wenn er ein Angebot zum Briefkurs kaufen kann. Die meisten Börsen belasten die Taker mit einigen Cent pro Aktie, schreiben den Makern etwas weniger gut und nehmen die Differenz selbst ein. Einige Börsen wie die BATS dagegen kassierten umgekehrt bei den Makern und belohnten die Taker (Lewis, 2014:S. 47f).

[39] Die BATS wurde von Hochfrequenzhändlern gegründet (Lewis (2014):S. 85).

[40] Lewis (2014):S. 46ff.

[41] Liquidität bezeichnet die Fähigkeit bzw. die Geschwindigkeit eines Finanzmarktes, Transaktionen auszuführen (Abel et al. (2008):S. 254), (Blanchard (2009):S. 90), (P. R. Krugman und Wells (2006):S. 224), (Samuelson und Nordhaus (1995):S. 482), (Wilmott (1998):§29.4). Je höher die Liquidität eines Finanzmarktes, desto schneller werden Angebot und Nachfrage erfüllt. Gegner des Hochfrequenzhandels weisen jedoch auf die Unterscheidung von Liquidität und Handelsaktivität hin. Da ein Hochfrequenzhändler sich im Wesentlichen einfach in einen Handelsprozess einklinkt, verdoppelt er im extremen Fall die Handelaktivität, ohne den geringsten Wert für den Markt zu schaffen (Lewis (2014):S.120).

[42] Lewis (2014):S 119.

[43] Lewis (2014):S. 121.

[44] Lewis (2014):S. 124.

[45] `http://handelsblatt.com/13461620.html`

Literatur

Abel, A. B., B. S. Bernanke und D. Croushore (2008). *Macroeconomics*. 6. Aufl. Pearson Education: Boston.

Ackermann, P. (2021). *Webentwicklung. Das Handbuch für Fullstack-Entwickler*. 1. Aufl. Rheinwerk: Bonn.

Albert, R. und A.-L. Barabási (Jan. 2002). „Statistical mechanics of complex networks“. In: *Rev. Mod. Phys.* 74 (1). arXiv:cond-mat/0106096, S. 47–97. DOI: 10.1103/RevModPhys.74.47.

Arthur, W. B. (1994). „Inductive reasoning and bounded rationality“. In: *American Economic Review*, S. 406–411.

Bandelow, C. (1989). *Einführung in die Wahrscheinlichkeitstheorie*. 2. Aufl. BI Wissenschaftsverlag: Mannheim Wien Zürich.

Barabási, A.-L. (2003). *Linked. How Everything Is Connected to Everything Else and What It Means for Business, Science, and Everyday Life*. Plume: New York.

Barnes, J. A. (1954). „Class and Committees in a Norwegian Island Parish“. In: *Human Relations* 7(1).

Barth, A. und I. Schnabel (2013). „Why banks are not too big to fail – evidence from the CDS market“. In: *Economic Policy* 28(74), S. 335–369. DOI: 10.1111/1468-0327.12007.

— (2014). *Der Abbau von impliziten Garantien im Bankensystem: Eine empirische Analyse auf Basis von CDS-Spreads*. Arbeitspapier 09/2014. http://hdl.handle.net/10419/103836. Sachverständigenrat zur Begutachtung der Gesamtwirtschaftlichen Entwicklung.

Bartol Jr., T. M. et al. (2015). „Nanoconnectomic upper bound on the variability of synaptic plasticity“. In: *eLife* 4(e10778). DOI: 10.7554/eLife.10778.

Basel Committee on Banking Supervision (Jan. 2013). *Basel III: The Liquidity Coverage Ratio and liquidity risk monitoring tools*. http://bis.org/publ/bcbs238.htm. Bank for International Settlements.

— (Okt. 2014). *Basel III: The net stable funding ratio*. http://bis.org/bcbs/publ/d295.htm. Bank for International Settlements.

Battiston, S. et al. (2012). „DebtRank: Too Central to Fail? Financial Networks, the FED and Systemic Risk“. In: *Scientific Reports* 2. DOI: 10.1038/srep00541.

Bauer, B. et al., Hrsg. (2015). *Atlas der Globalisierung. Weniger wird mehr*. Le Monde diplomatique/taz Verlag: Berlin.

Beck, A. et al. (2013). „Energiesysteme und das Paradigma des Agenten“. In: *Agentensysteme in der Automatisierungstechnik*. Hrsg. von P. Göhner. Xpert.press. Springer: Berlin Heidelberg, S. 21–42.

Bianconi, G. und A.-L. Barabási (Juni 2001a). „Bose-Einstein condensation in complex networks“. In: *Phys. Rev. Lett.* 86(24). arXiv:cond-mat/0011224, S. 5632–5635. DOI: 10.1103/PhysRevLett.86.5632.

— (2001b). „Competition and multiscaling in evolving networks“. In: *Europhys. Lett.* 54(4), S. 436–442. URL: https://arxiv.org/abs/cond-mat/0011029.

Blanchard, O. (2009). *Macroeconomics*. 5. Aufl. Pearson Education: Upper Saddle River.

Bofinger, P. (2007). *Grundzüge der Volkswirtschaftslehre. Eine Einführung in die Wissenschaft von Märkten*. 2. Aufl. Pearson Studium: München.

Börner, G. (2003). *The Early Universe*. 4. Aufl. Springer-Verlag: Berlin Heidelberg.

Bostrom, N. (2003). „Are You living in a Computer Simulation?“ In: *Philosophical Quarterly* 53(211), S. 243–255. DOI: 10.1111/1467-9213.00309.

Braess, D. (1968). „Über ein Paradoxon aus der Verkehrsplanung“. In: *Unternehmensforschung* 12, S. 258–268. DOI: 10.1007/BF01918335. URL: https://homepage.ruhr-uni-bochum.de/Dietrich.Braess/paradox.pdf.

Brandes, U. und G. Dorfmüller (2008). „PageRank: Was ist wichtig im World Wide Web?“ In: *Taschenbuch der Algorithmen*. Hrsg. von B. Vöcking et al. Springer-Verlag: Berlin Heidelberg. Kap. 10, S. 95–107. DOI: 10.1007/978-3-540-76394-9.

Brin, S. und L. Page (1998). „The anatomy of a large-scale hypertextual Web search engine“. In: *Computer Networks and ISDN Systems* 30, S. 107–117. DOI: 10.1016/S0169-7552(98)00110-X.

Bröcker, T. (1995). *Analysis II*. 2. Aufl. Spektrum Akademischer Verlag: Heidelberg.

Broy, M. und O. Spaniol (1999). *VDI-Lexikon Informatik und Kommunikationstechnik*. 2. Aufl. Springer: Berlin Heidelberg.

Buxmann, P., H. Diefenbach und T. Hess (2015). *Die Softwareindustrie. Ökonomische Prinzipien, Strategien, Perspektiven*. Springer Gabler: Berlin Heidelberg. DOI: 10.1007/978-3-662-45589-0.

Campbell, M., A. J. Hoarne Jr. und F. Hsu (2001). *Deep Blue*. http://sjeng.org/ftp/deepblue.pdf.

Cipolla, C. M. (1995). *Geld-Abenteuer*. Verlag Klaus Wagenbach: Berlin.

Clement, R. und D. Schreiber (2016). *Internet-Ökonomie. Grundlagen und Fallbeispiele der vernetzten Wirtschaft*. https://books.google.com/books?id=0aLtCwAAQBAJ. Springer Gabler: Berlin Heidelberg.

Dadfar, D., F. Schwartz und S. Voß (2012). „Risk management in global supply chains–Hedging for the big bang“. In: *Transportation & Logistics Management. Proceedings of the 17th International HKSTS Conference, HKSTS, Hong Kong*, S. 159–166.

Das, R., J. Hanson et al. (Okt. 2002). *Automated bidding agent for electronic auctions*. http://www.google.com/patents/US20020147675 (US Patent App. 09/829,701).

Das, R., J. E. Hanson et al. (2001). „Agent-human interactions in the continuous double auction“. In: *Proceedings of the 17th International Joint Conference on Artificial Intelligence*. Bd. 2. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.72.177. Morgan Kaufmann Publishers: San Francisco, S. 1169–1176.

de Vries, A. (1994). *Über die Beschränktheit der Energienorm bei der Evolution der Dirac-, Weyl- und Maxwellfelder in gekrümmten Raumzeiten*. Brockmeyer: Bochum. URL: http://math-it.org/Publikationen/diss.pdf.

— (2006). „Die Relativität der Information“. In: *Jahresschrift der Bochumer Interdisziplinären Gesellschaft eV 2004*. Hrsg. von R. W. Muno. ibidem-Verlag: Stuttgart, S. 11–38. URL: https://math-it.org/Publikationen/Information.pdf.

— (2012a). *Kryptologie. Einführung und Überblick*. Bd. 1. Hagener Berichte der Wirtschaftsinformatik. Books on Demand: Norderstedt, S. 9–70.

— (2012b). *P ≠ NP?* Bd. 1. Hagener Berichte der Wirtschaftsinformatik. Books on Demand: Norderstedt, S. 71–102.

— (2012c). *Quantenrechnen. Eine Einführung in Quantum Computation für Ingenieure und Informatiker*. Books On Demand: Norderstedt.

— (2020). *Netzökonomie Lerneinheit 4. Soziale Netzwerkanalyse und Netzwerkeffekte*. Vorlesungsskript. https://fh-swf.sciebo.de/s/EV8XSxsWD8W6XZc. Hagen.

— (2022). *Webtechnologie 1: Grundlagen*. Vorlesungsskript. Hagen. URL: https://www.fh-swf.de/media/neu_np/fb_tbw_1/dozentinnen_2/professorinnen_5/devries_1/WebTech-1.pdf.

Delahaye, J.-P. (2016a). „Intelligenz bei Mensch und Maschine“. In: *Spektrum der Wissenschaft* 3. https://spektrum.de/artikel/1396798, S. 78–85.

— (2016b). „Müssen wir Killerroboter verbieten?“ In: *Spektrum der Wissenschaft* 5. https://spektrum.de/artikel/1405261, S. 80–86.

Di Ventra, M. und Y. V. Pershin (2016). „Computerchips mit integriertem Gedächtnis“. In: *Spektrum der Wissenschaft* 4. https://spektrum.de/artikel/1400772, S. 80–84.

Diamond, D. W. und P. H. Dybvig (Juni 1983). „Bank runs, deposit insurance, and liquidity“. In: *Journal of Political Economy* 91(3), S. 401–419. URL: http://jstor.org/stable/1837095.

Diestel, R. (2000). *Graphentheorie*. 2. Aufl. Springer-Verlag: Berlin Heidelberg.

Dillerup, R. und R. Stoi (2013). *Unternehmensführung*. Verlag Franz Vahlen: München.

Easley, D. und J. Kleinberg (2010). *Networks, Crowds, and Markets. Reasoning about a Highly Connected World*. Cambridge University Press: Cambridge New York.

Edlich, S. (2013). „Konterrevolution. NewSQL: Relational schlägt zurück“. In: *iX Programmieren heute* 1. http://www.ix.de/ix1217074.

Edlich, S. et al. (2010). *NoSQL. Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. Hanser: München.

Ehmke, C. (2017). „Blockchain aus der Entwicklerperspektive“. In: *Java Spektrum* 3, S. 10–13.

Englert, M., S. Siebert und M. Ziegler (2014). „Logical Limitations to Machine Ethics with Consequences to Lethal Autonomous Weapons“. In: *CoRR* abs/1411.2842. http://arxiv.org/abs/1411.2842.

Eyal, I. und E. G. Sirer (2013). „Majority is not Enough: Bitcoin Mining is Vulnerable“. In: *CoRR* abs/1311.0243. http://arxiv.org/abs/1311.0243.

Eysenck, M. W. und M. T. Keane (1995). *Cognitive Psychology. A Student's Handbook*. 3. Aufl. Psychology Press: Hove.

Ferguson, N. (2009). *Der Aufstieg des Geldes*. Econ: Berlin.

Ferrucci, D. et al. (2010). „Building Watson: An Overview of the DeepQA Project“. In: *AI Magazine* 31(3), S. 59–79. DOI: 10.1609/aimag.v31i3.2303.

Fielding, R. T. (2000). „Architectural Styles and the Design of Network-based Software Architectures“. Diss. Irvine: University of California. URL: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Fischer, M. (2009). *Web Boosting 2.0. Suchmaschinen-Optimierung, Usability, Online-Marketing*. 2. Aufl. mitp: Heidelberg.

Fondermann, B. (2016). „Der Weg des maschinellen Lernens“. In: *Java Magazin* 3, S. 11–13.

Foot, P. (1967). „The Problem of Abortion and the Doctrine of the Double Effect“. In: *Oxford Review*(5). http://philpapers.org/archive/footpo-2.pdf.

Fowler, M. (Jan. 2004). *Inversion of Control Containers and the Dependency Injection pattern*. http://martinfowler.com/articles/injection.html.

— (Juni 2005). *InversionOfControl*. http://martinfowler.com/bliki/InversionOfControl.html.

Fudenberg, D. und J. Tirole (1991). *Game Theory*. MIT Press: Cambridge.

Fukushima, K. (1980). „Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position“. In: *Biological Cybernetics* 36(4), S. 193–202. DOI: 10.1007/BF00344251.

Gaede, P. M., Hrsg. (2008). *Die Industrielle Revolution*. GEO Epoche 30. Gruner + Jahr: Hamburg.

Goldberg, D. (März 1991). *What Every Computer Scientist Should Know About Floating-Point Arithmetic*. https://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html.

Gorski, P. L., L. L. Iacono und H. V. Nguyen (2015). *WebSockets. Moderne HTML5-Echtzeitanwendungen entwickeln*. Hanser: Müchen.

Gould, J. L. und C. G. Gould (1997). *Bewusstsein bei Tieren*. Spektrum Akademischer Verlag: Heidelberg Berlin Oxford.

Graeber, D. (2012). *Schulden. Die ersten 5000 Jahre*. Klett-Cotta: Stuttgart.

Günther, S. (2014). *Die Vermeidung von Bank Runs und der Erhalt von Marktdisziplin: Das Dilemma der Bankenregulierung?* Arbeitspapiere 142. http://hdl.handle.net/10419/97485. Institut für Genossenschaftswesen der Westfälischen Wilhelms-Universität Münster.

Güntürkün, O. (2021). „Federvieh mit Köpfchen“. In: *Spektrum der Wissenschaft*. https://spektrum.de/artikel/1795019, S. 30–37.

Gürsoy, H.-C. (2012). „Was Architekten bei NoSQL-Datenbanken beachten sollten“. In: *Java Spektrum* 3, S. 9–12.

Hardcastle, V. G. (1995). *Locating Consciousness*. John Benjamins: Amsterdam Philadelphia.

Harris, J. M., J. L. Hirst und M. J. Mossinghoff (2008). *Combinatorics and Graph Theory*. 2. Aufl. Springer: New York.

Hayden, T., Hrsg. (2008). *Planet Erde 2008*. Collector's Edition 9. National Geographic Society Deutschland: Hamburg.

Hayes, B. (2014). „Die neuronalen Netze werden erwachsen“. In: *Spektrum der Wissenschaft* 9. https://spektrum.de/artikel/1303093.

Helbig, D. et al. (2016). „Digital-Manifest: Digitale Demokratie statt Datendiktatur“. In: *Spektrum der Wissenschaft* 1. http://spektrum.de/s/digitalmanifest, S. 51–60.

Hevelke, A. und J. Nida-Rümelin (2015). „Intelligente Autos im Dilemma“. In: *Spektrum der Wissenschaft* 10. http://spektrum.de/artikel/1362277, S. 82–85.

Hilbert, M. und P. López (Apr. 2011). „The World's technological capacity to store, communicate, and compute information“. In: *Science* 332(6025), S. 60–65. DOI: 10.1126/science.1200970.

— (2012). „How to measure the World's technological capacity to store, communicate, and compute information? Part I. results and scope“. In: *International Journal of Communication* 6, S. 956–979.

Hill, R. A. und R. I. M. Dunbar (2003). „Social network size in humans“. In: *Human Nature* 14(1), S. 53–72. DOI: 10.1007/s12110-003-1016-y.

Hoffman, D. D. (2000). *Visuelle Intelligenz. Wie die Welt im Kopf entsteht*. Cotta'sche Buchhandlung: Stuttgart.

Hofstadter, D. R. (1986). *Gödel, Echer, Bach: ein Endloses Geflochtenes Band*. 9. Aufl. Klett-Cotta: Stuttgart.

Hubel, D. H. und T. N. Wiesel (Jan. 1962). „Receptive fields, binocular interaction and functional architecture in the cat's visual cortex“. In: *Journal of Physiology* 160(1), S. 106–154. DOI: 10.1113/jphysiol.1962.sp006837.

Hume, D. (1896). *A Treatise of Human Nature [1739]*. Hrsg. von L. Selby-Bigge. Clarendon Press: Oxford. URL: http://oll.libertyfund.org/title/342.

Humphries, M. D. und K. Gurney (Apr. 2008). „Network 'Small-World-Ness': A Quantitative Method for Determining Canonical Network Equivalence“. In: *PLoS ONE* 3(4), e0002051. DOI: 10.1371/journal.pone.0002051.

Hutter, M. (2005). *Universal Artificial Intelligence. Sequential Decisions Based on Algorithmic Probability*. Springer: Berlin New York.

Johnson, N. P. A. S. und J. Mueller (2002). „Updating the accounts: global mortality of the 1918–1920 “Spanish” influenza pandemic“. In: *Bulletin of the History of Medicine* 76(1), S. 105–115. DOI: 10.1353/bhm.2002.0022.

Kaderali, F. und W. Poguntke (1995). *Graphen, Algorithmen, Netze. Grundlagen und Anwendungen in der Nachrichtentechnik*. Vieweg: Braunschweig Wiesbaden.

Kadtke, J. und L. Wells II (Sep. 2014). *Policy Challenges of Accelerating Technological Change: Security Policy and Strategy Implications of Parallel Scientific Revolutions*. http://ctnsp.dodlive.mil/files/2014/09/DTP1061.pdf. Center for Technology und National Security Policy / National Defense University. Washington, D.C.

Kang, U. et al. (2011). „Centralities in Large Networks: Algorithms and Observations“. In: *Proceedings of the SIAM International Conference on Data Mining*. http://www.cs.cmu.edu/~ukang/papers/CentralitySDM2011.pdf.

Kaumanns, R. und V. Siegenheim (2007). *Die Google-Ökonomie. Wie Google die Wirtschaft verändert*. Books on Demand: Norderstedt.

Keen, A. (2015). *Das digitale Debakel*. https://books.google.de/books?id=w0ebAwAAQBAJ. Deutsche Verlags-Anstalt: München.

Kessler, T. et al. (2014). *Reporting mit SAP® BW und SAP® BusinessObjects™*. 2. Aufl. Galileo Press: Bonn Boston.

Kiyak, C. und A. de Vries (2015). „Electricity markets regarding the operational flexibility of power plants“. In: *to appear*. http://arxiv.org/abs/1502.00120.

— (Apr. 2017). „Electricity markets regarding the operational flexibility of power plants“. In: *Modern Economy* 8(4), S. 567–589. DOI: 10.4236/me.2017.84043.

Knapp, G. F. (1905). *Staatliche Theorie des Geldes*. https://books.google.com/books?id=h-TnYpkQSYgC (English edition: Knapp (1924)). Duncker & Humblot: Leipzig.

— (1924). *State Theory of Money*. http://socserv2.socsci.mcmaster.ca/~econ/ugcm/3ll3/knapp/StateTheoryMoney.pdf. MacMillan: London.

Kollmann, T. (2009). *E-Business. Grundlagen elektronischer Geschäftsprozesse in der Net Economy*. 3. Aufl. Gabler: Wiesbaden.

Krugman, P. (2009). *Die neue Weltwirtschaftskrise*. Campus Verlag: Frankfurt.

Krugman, P. R. und M. Obstfeld (2009). *International Economics. Theory and Policy*. 8. Aufl. Pearson: Boston.

Krugman, P. R. und R. E. Wells (2006). *Macroeconomics*. Worth Publishers: New York.

Kümmel, R. (2011). *The Second Law of Economics. Energy, Entropy, and the Origins of Wealth*. Springer: New York Dordrecht Heidelberg London. DOI: 10.1007/978-1-4419-9365-6.

Kurowski, O. (2012). *CouchDB mit PHP*. entwickler.press: Frankfurt.

Kurz, C. und F. Rieger (2013). *Arbeitsfrei. Eine Entdeckungsreise zu den Maschinen, die uns ersetzen*. Riemann Verlag: München.

Lanchester, J. (Juli 2014). „Der Super-Klick. Wie Hochfrequenzhandel funktioniert“. In: *Monde diplomatique*. http://monde-diplomatique.de/artikel/!327186.

Landau, L. D. und E. M. Lifschitz (1997). *Klassische Feldtheorie*. 12. Aufl. Harri Deutsch: Frankfurt.

Lanier, J. (2014). *Wem gehört die Zukunft?* 6. Aufl. Hoffmann und Campe: Hamburg.

Le Cun, Y., Y. Bengio und G. Hinton (Mai 2015). „Deep Learning“. In: *Nature* 521, S. 436–444. DOI: 10.1038/nature14539.

Lewis, M. (2014). *Flash Boys. Revolte an der Wall Street*. Campus: Frankfurt.

Mackie, J. L. (1974). *The Cement of the Universe. A Study of Causation*. Oxford University Press: Oxford.

Maddison, A. (2007). *Contours of the World Economy, 1–2030 AD. Essays in Macro-Economic History*. Oldenbourg Verlag: München Wien.

Malcher, F., J. Hoppe und D. Koppenhagen (2020). *Angular. Grundlagen, fortgeschrittene Themen und Best Practices – inkl. RxJS, NgRx und PWA*. 3. Aufl. iX-Edition. dpunkt.verlag: Heidelberg.

Maurice, F. (2014). *PHP 5.5 und MySQL 5.6*. 3. Aufl. dpunkt.verlag: Heidelberg.

Mayer-Schönberger, V. (2015). „Was ist Big Data? Zur Beschleunigung des menschlichen Erkenntnisprozesses“. In: *Aus Politik und Zeitgeschichte* 65(11–12). http://www.bpb.de/shop/zeitschriften/apuz/202251/big-data, S. 14–19.

McCulloch, W. S. und W. Pitts (1943). „A logical calculus of the ideas immanent in nervous activity“. In: *Bulletin of Mathematical Biophysics* 5(4), S. 115–133. DOI: 10.1007/bf02478259.

McLuhan, M. (1964 (reissued 2001)). *Understanding Media. The Extensions of Man*. https://books.google.com/books?id=K4AWBwAAQBAJ. Routledge: Oxon.

Melzer, I. (2008). *Service-orientierte Architekturen mit Web Services*. 3. Aufl. Spektrum Akademischer Verlag: Heidelberg.

Mertens, P. und D. Barbian (2014). „Die Wirtschaftsinformatik der Zukunft – auch eine Wissenschaft der Netze?“ In: *HMD Praxis der Wirtschaftsinformatik* 51(6), S. 729–743. DOI: 10.1365/s40702-014-0096-y.

Mnih, V. et al. (Feb. 2015). „Human-level control through deep reinforcement learning“. In: *Nature* 518(7540), S. 529–533. DOI: 10.1038/nature14236.

Morris, I. (2010). *Why The West Rules – For Now. The Patterns of History and What They Reveal About the Future*. Profile Books: London. ISBN: 978-1-84668208-7.

— (2011). *Wer regiert die Welt?* Campus Verlag: Frankfurt New York. ISBN: 978-3-593-38406-1.

Moschitti, A. et al. (2011). „Using Syntactic and Semantic Structural Kernels for Classifying Definition Questions in Jeopardy!“ In: *Proceedings of the Conference on Empirical Methods for Natural Language Processing*. Hrsg. von P. Merlo. Association for Computational Linguistics: Stroudsburg, S. 712–724.

Nakamoto, S. (Nov. 2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. https://bitcoin.org/bitcoin.pdf.

— (Feb. 2009). *Bitcoin open source implementation of P2P currency*. http://p2pfoundation.ning.com/forum/topics/bitcoin-open-source.

Newman, M. E. J. (2010). *Networks. An Introduction*. Oxford University Press: Oxford New York.

Newman, M. E. J., A.-L. Barabási und D. J. Watts (2006). *The Structure and Dynamics of Networks*. Princeton University Press: Princeton.

Newman, M. E. J., S. H. Strogatz und D. J. Watts (2001). „Random graphs with arbitrary degree distributions and their applications“. In: *Phys. Rev. E* 64(026118). http://arxiv.org/abs/cond-mat/0007235. DOI: 10.1103/PhysRevE.64.026118.

Nørretranders, T. (1997). *Spüre die Welt. Die Wissenschaft des Bewusstseins*. Rowohlt Verlag: Reinbek bei Hamburg.

OECD (2008). „The Seoul Declaration for the Future of the Internet Economy“. In: *OECD Digital Economy Papers* 147. DOI: 10.1787/20716826.

Page, L. et al. (Nov. 1999). *The PageRank Citation Ranking: Bringing Order to the Web*. Technical Report 1999-66. http://ilpubs.stanford.edu:8090/422/. Stanford InfoLab.

Penrose, R. (1995). *Schatten des Geistes. Wege zu einer neuen Physik des Bewußtseins*. Spektrum Akademischer Verlag: Heidelberg Berlin Oxford.

Pflock, T. M. (2014). *Europäische Bankenregulierung und das „Too big to fail-Dilemma“*. Berliner Wissenschaftsverlag. ISBN: 9783830534013. URL: https://books.google.de/books?id=ErdoBAAAQBAJ.

Radford, R. A. (1945). „The Economic Organisation of a P.O.W. Camp“. In: *Economica*. New Series 12(48), pp. 189–201. DOI: 10.2307/2550133.

Rink, J., Hrsg. (2012). *Soziale Netze*. c't extra. Heise: Hannover.

Rohjans, S., S. Lehnhoff und M. Büscher (2014). „SESA-Lab: Gesamtsystemische Smart Grid Simulationen Generischer Automatisierungsarchitekturen“. In: *VDE-Kongress 2014*. Kongressbeiträge 20. VDE Verlag, S. 308–313.

Rudolph, B. (2014). „Bankregulierung zur Lösung des „too big to fail“-Problems“. In: *Die Unternehmung* 2, S. 71–74.

Russell, S. J. und P. Norvig (2022). *Artificial Intelligence. A Modern Approach*. 4. Aufl. Pearson: Harlow.

Salmon, F. (Apr. 2014). „The Lewis Effect“. In: *Slate Book Review*. http://www.slate.com/articles/business/books/2014/04/michael_lewis_s_flash_boys_about_high_frequency_trading_reviewed.html.

Samuelson, P. A. und W. D. Nordhaus (1995). *Economics*. 15. Aufl. McGraw-Hill: New York etc.

Satterthwaite, M. A. und S. R. Williams (1989). „Bilateral trade with the sealed bid k-double auction: Existence and efficiency“. In: *Journal of Economic Theory* 48(1), S. 107–133.

Schaper, M., Hrsg. (2014). *Der Kapitalismus*. GEO Epoche 69. Gruner + Jahr: Hamburg.

Scherzer, H. (Apr. 2016). „Auf der Suche nach dem ultimativen Geld“. In: *Informatik Spektrum*. DOI: 10.1007/s00287-016-0956-7.

Schlieter, K. (2015). *Die Herrschaftsformel. Wie künstliche Intelligenz uns berechnet, steuert und unser Leben verändert*. Ebook. Westend Verlag: Frankfurt a. M.

Shanahan, M. et al. (2013). „Large-scale network organization in the avian forebrain: a connectivity matrix and theoretical analysis“. In: *Frontiers in Computational Neuroscience* 7, S. 89. DOI: 10.3389/fncom.2013.00089.

Sinn, H.-W. (2009). *Kasino-Kapitalismus. Wie es zur Finanzkrise kam, und was jetzt zu tun ist*. Econ: Berlin.

Sipser, M. (2006). *Introduction to the Theory of Computation*. Thomson Course Technology: Boston.

Spichale, K. und E. Wolff (2013). „Ordnungshüter. NoSQL-Datenbanken ergänzen relationale Datenbanksysteme“. In: *iX Programmieren heute* 1.

Stein, E. (2001). *Taschenbuch Rechnernetze und Internet*. Carl Hanser Verlag: München Wien.

Szczepański, M. (2014). *Bitcoin. Market, economics and regulation*. Briefing 140793REV1. http://www.europarl.europa.eu/RegData/bibliotheque/briefing/2014/140793/LDM_BRI(2014)140793_REV1_EN.pdf. European Parliamentary Research Service.

Tapscott, D. und A. D. Williams (2007). *Wikinomics. Die Revolution im Netz*. Carl Hanser Verlag: München.

Tarasiewicz, P. und R. Böhm (2014). *AngularJS. Eine praktische Einführung in das JavaScript-Framework*. dpunkt.verlag: Heidelberg.

Theis, T. (2018). *Einstieg in PHP 7 und MySQL*. Rheinwerk: Bonn.

Unsöld, A. und B. Baschek (1999). *Der neue Kosmos. Einführung in die Astronomie und Astrophysik*. 6. Aufl. Springer Verlag: Berlin Heidelberg New York.

Varshney, L. R. et al. (Feb. 2011). „Structural Properties of the *Caenorhabditis elegans* Neuronal Network“. In: *PLoS Comput Biol* 7(2), e1001066. DOI: 10.1371/journal.pcbi.1001066.

Vise, D. und M. Malseed (2006). *Die Google-Story*. Murmann Verlag: Hamburg.

von Weizsäcker, C. F. (1985). *Aufbau der Physik*. Carl Hanser Verlag: München Wien.

— (1992). *Zeit und Wissen*. Carl Hanser Verlag: München Wien.

Wagenknecht, S. (2008). *Wahnsinn mit Methode. Finanzcrash und Weltwirtschaft*. 2. Aufl. Das Neue Berlin: Berlin.

Wallace, B. (Nov. 2011). „The rise and fall of Bitcoin“. In: *Wired*. http://www.wired.com/2011/11/mf_bitcoin/.

Wasserman, S. und K. Faust (1994). *Social Network Analysis*. Cambridge University Press: Cambridge.

Watkins, C. und P. Dayan (1992). „Q-learning“. In: *Machine Learning* 8(3–4), S. 279–292.

Watts, D. J. (2002). „A simple model of global cascades on random networks“. In: *Proc. Natl. Acad. Sci. USA* 99(9), S. 5766–5771. DOI: 10.1073/pnas.082090499.

Watts, D. J. und S. H. Strogatz (1998). „Collective dynamics of «small-world» networks“. In: *Nature* 393(6684). DOI: 10.1038/30918.

Welzel, H. (1951). „Zum Notstandsproblem“. In: *Zeitschrift für die gesamte Strafrechtswissenschaft* 63(1), S. 47–56. DOI: 10.1515/zstw.1951.63.1.47.

Wenz, C. und T. Hauser (2021). *PHP 8 und MySQL. Das umfassende Handbuch*. 4. Aufl. Rheinwerk: Bonn.

Wertpapierhandelsgesetz (o. D.). https://www.gesetze-im-internet.de/wphg/. Bundesjustizministerium, Berlin.

Wilmott, P. (1998). *Derivatives. The Theory and Practice of Financial Engineering*. John Wiley: Chichester.

Wilson, R. (1985). „Incentive efficiency of double auctions“. In: *Econometrica*. http://www.jstor.org/stable/1911013, S. 1101–1115.

Wolff, E. (2013). „Die NoSQL-Übersicht“. In: *Java Magazin* 10. https://jaxenter.de/magazines/Java-Magazin-1013-166680, S. 76–79.

Wrobel, S. et al. (2015). „Big Data, Big Opportunities. Anwendungssituation und Forschungsbedarf des Themas Big Data in Deutschland“. In: *Informatik Spektrum* 38(5). DOI: 10.1007/s00287-014-0806-4.

Zeidler, E., Hrsg. (1996). *Teubner Taschenbuch der Mathematik. Teil 1*. B. G. Teubner: Leipzig.

Zimmer, C. (2011). „Das Gehirn als Netzwerk“. In: *Spektrum der Wissenschaft* 10, S. 23–28. URL: https://spektrum.de/artikel/1121034.

Internetquellen

[DOM] https://w3.org/DOM – Spezifikation des W3C des DOM.

[GFS] http://research.google.com/archive/gfs-sosp2003.pdf – Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: *The Google File System* (2003) [15.03.2015]

[GMP] http://www.masterplanthemovie.com/ – Ozan Halizi und Jürgen Mayer: *„The Google Master Plan“*, Film nach ihrer Bachelor-Thesis an der FH Ulm 2007. Kritische und beunruhigende Betrachtung zu Google und dessen Wachstum, sowie Spekulationen zu dessen Motiven. Der Film ist sogar (noch?) auf der zum Google-Konzern gehörenden Plattform YouTube, http://www.youtube.com/watch?v=9zKXCQpUnMg

[JS] https://developer.mozilla.org/en-US/docs/Web/JavaScript – JavaScript-Dokumentation des Mozilla Developer Network.

[JSL] http://www.jslint.com/ – JSLint, Programm zur Überprüfung der Qualität von JavaScript Quelltext.

[OWASP] https://owasp.org/ – Open Web Application Security Project (OWASP), gemeinnützige („501(c)(3)“) Organisation mit dem Ziel, die Sicherheit von Software im WWW zu verbessern und durch Transparenz Entscheidungskompetenz bei Endanwendern und Organisationen in Fragen von Sicherheitsrisiken zu schaffen. Empfehlenswert ist die Wikiseite zu Sicherheitslücken (*vulnerability*): https://www.owasp.org/index.php/Category:Vulnerability

[SuMa] http://suma-ev.de/ – SuMa eV, gemeinnütziger Verein zur Förderung der Suchmaschinentechnologie und des freien Wissenszugangs. Hauptziel: Aufbau einer dezentralen und kooperativen Suchmaschinen-Struktur in Deutschland, die schwer monopolisierbar ist.

[TF] https://www.tensorflow.org/ – TensorFlow, quelloffene Softwarebibliothek, die die Programmierung neuronaler Netze auf verteilten Systemen ermöglicht.

[W3C] http://www.w3c.org/ – World Wide Web Consortium, Gremium zur Standardisierung der Web-Techniken;

Index