März 2021
9. Jahrg.
84364
Seite 1–56

InTeR

Zeitschrift zum Innovations- und Technikrecht

1

dfv Mediengruppe · Frankfurt am Main

begriffs erfüllen, handelt es sich bei ihnen – mangels Einschlägigkeit eines Erlaubnistatbestandes – um eine nicht erlaubnisfähige und damit unzulässige Rechtsdienstleistung. Andernfalls unterliegen solche Generatoren schon nicht dem Anwendungs- bzw. Verbotsbereich des RDG.

Online-Rechtsdokumentengeneratoren weisen im Vergleich zu „klassischen" (Rechts-)Dienstleistungen markante Besonderheiten digitaler Prägung auf, die der Gesetzgeber bei der Konzeption des RDG noch nicht vor Augen hatte. Namentlich die den Generatoren wesensimmanente Nutzung einer programmierten Software, welche nach ihrer Bereitstellung im Internet völlig automatisiert und ohne eine menschliche Interaktion Rechtsdokumente erzeugt, führt zu einigen Schwierigkeiten beim Versuch der Subsumtion solcher Angebote unter den „tradierten" Rechtsdienstleistungsbegriff des § 2 Abs. 1 RDG. Bereits über das Vorliegen des Tatbestandsmerkmals einer „Tätigkeit" lässt sich angesichts der Vielgestaltigkeit der im Zusammenhang mit einem Online-Rechtsdokumentengenerator entfalteten Aktivitäten trefflich streiten. Nach dem hier zugrunde gelegten Verständnis übt der Anbieter eines Online-Rechtsdokumentengenerators mit der Programmierung und internetbasierten Bereitstellung der entsprechenden Software zwar eine „Tätigkeit" in „fremden Angelegenheiten" aus; diese Tätigkeit wird allerdings nicht – wie es § 2 Abs. 1 RDG weiterhin verlangt – in „konkreten Angelegenheiten" entfaltet. Ungeachtet dessen erfolgt mit der Programmierung der Software auch keine „rechtliche Prüfung des Einzelfalls". Online-Rechtsdokumentengeneratoren sind daher nach dem hier gefundenen Ergebnis nicht als Rechtsdienstleistung i. S. v. § 2 Abs. 1 RDG zu qualifizieren.

Wie allerdings die vorstehenden Ausführungen deutlich gemacht haben dürften, gleicht die Ergebnisfindung einer „auslegerischen Gratwanderung"[48]. Je nachdem auf welche (ggf. variierenden) Anknüpfungspunkte man zur Beurteilung für das Vorliegen der einzelnen Tatbestandsmerkmale des § 2 Abs. 1 RDG abstellt, sowie unter Berücksichtigung der individuellen Merkmale des jeweils in Betracht genommenen Online-Rechtsdokumentengenerators erscheint die Einordnung einzelner Generatoren als Rechtsdienstleistung sicherlich ebenso gut vertretbar. Dieser Befund verweist unterdessen auf einen legislativen Handlungsbedarf.[49] Insoweit könnte vor allem mit der Verankerung einer klarstellenden und den Spezifika von automatisierten (Rechts-)Dienstleistungen gerecht werdenden Regelung im RDG ein solides Maß an Rechtsklarheit in den Rechtsdienstleistungssektor hineingetragen werden.

48 *Wessels,* MMR 2020, 56, 59.

49 Siehe hierzu den von der FDP-Bundestagsfraktion unternommenen Vorstoß in Gestalt des Entwurfs eines Gesetzes zur Modernisierung des Rechtsdienstleistungsrechts v. 18.4.2019, BT-Drs. 19/9527.

Dipl.-Jur. Marvin Gülker, Passau*

Der urheberrechtliche Schutz von Schnittstellen, insbesondere von APIs – Teil I

Der Aufsatz geht anlässlich des Streits um die Programmiersprache Java zwischen Google und Oracle in den USA der Frage nach, wie es um den Schutz von Schnittstellen hierzulande bestellt ist. Der erste Teil erläutert technische wie rechtliche Unterschiede zwischen sonst oft zusammen behandelten Arten sogenannter APIs und kommt zu dem Ergebnis, dass allgemeine Aussagen über deren Schutz nicht möglich sind. Dem zweiten Teil vorbehalten ist ein neuer Ansatz zur Prüfung der Schutzfähigkeit jenseits der oft diskutierten „Merger Doctrine".

I. Einleitung[1]

1991 trat die Richtlinie über den Rechtsschutz von Computerprogrammen[2] in Kraft, die 2009 mit nur unbedeutenden Änderungen betreffs der Schutzdauer neu kodifiziert wurde[3] (nachfolgend: CPRL). Ihre Bestimmungen sind nahezu wortgleich in den §§ 69a ff. UrhG in das deutsche Recht umgesetzt worden[4] und lassen bis heute Fragen offen, welche die bisher nur vereinzelte Rechtsprechung des EuGH nicht zu klären vermag. Zu diesen offenen Fragen gehört auch der urheberrechtliche Schutz der in § 69a Abs. 2 S. 2 UrhG erwähnten Schnittstellen, der den Gegenstand dieses Aufsatzes bildet. Wie der seit Jahren schwelende und kürzlich vor dem Supreme Court verhandelte[5] milliardenschwere Rechtsstreit zwischen Google und Oracle in den USA um die Übernahme von Schnittstellen der Programmiersprache Java zeigt, handelt es sich dabei nicht etwa nur um eine akademische Randnotiz.

Dieser Aufsatz zeigt zunächst die technischen Grundlagen auf (unten II.), bevor sich deren rechtliche Würdigung anschließt, die mit der Untersuchung der Computerprogramm-Definition beginnt (unten III.) und danach die Schnittstellen betrachtet (unten IV. und V.).

II. Technischer Hintergrund

1. Allgemeines zu Schnittstellen

In der Informatik versteht man unter einer Schnittstelle (engl. Interface) die gemeinsame Grenze zwischen zwei

* Mehr über den Autor erfahren Sie auf Seite III.

1 Der Aufsatz basiert auf einer Seminararbeit des Verf., die er am Lehrstuhl für Bürgerliches Recht, Internationales Privatrecht und Rechtsvergleichung, Handels- und Wirtschaftsrecht (Prof. Dr. *Renate Schaub,* LL.M.) an der Ruhr-Universität Bochum geschrieben hat.

2 Richtlinie 91/259/EWG des Rates über den Rechtsschutz von Computerprogrammen (ABl. L Nr. 122v. 17.5.1991, S. 42).

3 Richtlinie 2009/24/EG des europäischen Parlaments und des Rates über den Rechtsschutz von Computerprogrammen (ABl. L Nr. 111v. 5.5.2009, S. 16).

4 Zweites Gesetz zur Änderung des UrhG, BGBl. I 1993, 910.

5 *Lindner,* F.A.Z. v. 7.10.2020, S. 18.

Komponenten mithilfe von Funktionen, physischen Verbindungen, Signalaustausch oder anderen Charakteristika.[6] Es handelt sich um einen Oberbegriff für eine Vielzahl verschiedener, im Laufe der Jahre immer zahlreicher werdender Schnittstellenarten, darunter Hardware- und Betriebssystemschnittstellen oder Benutzeroberflächen.[7] Schnittstellen dienen stets dem Austausch von Informationen zwischen den beiden Komponenten.[8] Gelingt dieser, sind die beiden Komponenten „interoperabel".[9] Anders ausgedrückt: In technischer Hinsicht ist Schnittstelle die Gesamtheit derjenigen Dinge, die zur Herstellung solcher Interoperabilität zwischen zwei Komponenten nötig sind.[10]

Damit dieser Datenaustausch funktioniert, müssen die beteiligten Computerprogramme den Computer jeweils in einer dem Regelwerk der Schnittstelle entsprechenden Weise ansteuern. Den Code, der diese Regeln umsetzt, bezeichnet man als „Implementation" einer Schnittstelle.[11] Plastisch am Beispiel von Hardware-Schnittstellen ausgedrückt bedeutet dies, dass etwa die USB-Buchse mit dem Regelwerk für den Datenaustausch über USB die Schnittstelle darstellt, der USB-Treiber des Betriebssystems und die Firmware der Computermaus dagegen Implementationen der Schnittstelle sind. Für Software-Schnittstellen verhält es sich ähnlich, nur existiert hierbei keine für Endnutzer erkennbare physische Manifestation der Schnittstelle, wie es bei der USB-Buchse der Fall ist. Aus den Gesetzesmaterialien zur CPRL stammt das Beispiel, man solle sich die Schnittstelle vorstellen wie eine Steckdose, die das Stromnetz auf der einen Seite mit dem Verbraucher auf der anderen Seite verbindet[12].

Schnittstellen ermöglichen es dem Programmierer, komplexe Aufgaben in handhabbare Teile zu zerlegen. Sie ermöglichen es auch, für Unteraufgaben Fremdprogramme zu benutzen und so eigenen Aufwand einzusparen, was insbesondere beim Einsatz von Open-Source-Software deutlich wird.[13] Die Aufteilung gewährleistet darüber hinaus Austauschbarkeit, Wiederverwertbarkeit und unabhängige Verbesserungen der einzelnen Komponenten.[14]

2. Über APIs

Gegenstand dieses Aufsatzes ist eine besondere Form der Schnittstelle, das „Application Programming Interface" (API). Hierbei zeigt sich, dass aufgrund jüngerer technischer Entwicklung mit Bezug auf das World Wide Web der Begriff des API zwei ganz unterschiedliche Schnittstellenarten meinen kann: „klassische" APIs und die neu aufkommenden Web-APIs.[15] Es wird hiermit vorgeschlagen, in der rechtswissenschaftlichen Diskussion den Begriff des API deshalb nicht isoliert zu verwenden, sondern stets klarzustellen, ob die Rede von klassischen APIs oder von Web-APIs ist.

a) Klassische APIs

Bei klassischen APIs handelt es sich um Software-Software-Schnittstellen. Ein Computerprogramm soll Funktionen eines anderen Computerprogramms nutzen, das sich auf demselben Computer befindet.[16] Hierbei ruft das eine Computerprogramm Funktionen des anderen Computerprogramms direkt auf, nachdem zuvor beide Computerprogramme in den Arbeitsspeicher geladen wurden.[17] Das funktioniert nur, wenn schon im Quellcode-Stadium beide Computerprogramme dieselben Namen, Parameter und Rückgabewerte der aufzurufenden Funktionen in identischer Weise verwenden. Man bezeichnet die entsprechenden Code-Teile als Deklarationen[18]; ihre Gesamtheit bildet das (klassische) API eines Computerprogramms[19]. Die Implementation (zu dem Begriff schon oben) ist in diesem Kontext der konkrete Code, der die Aufgabe der Funktion löst, und kann je nach Programmierer bei gleichbleibendem API bzw. gleichbleibender Funktionsdeklaration sehr unterschiedlich ausfallen[20].

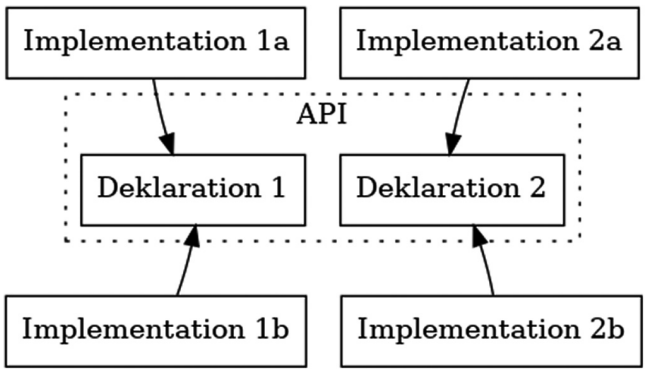
Ein Beispiel: Die Standardbibliothek[21] der Programmiersprache C enthält eine Funktion namens fopen(). Diese besitzt folgende Deklaration bzw. folgendes API:[22]

```
FILE* fopen(const char* filename,
            const char* mode);
```

Das dargestellte Quelltext-Stück besagt, dass fopen() zwei unveränderliche (const) Zeichenketten (Typenbezeichnung: char*) annimmt: den Dateinamen (filename) und einen hier nicht näher zu erläuternden Zugriffsmodus (mode), der ebenfalls als Zeichenkette zu übergeben ist. Zurückgegeben wird dann ein Verweis auf die geöffnete Datei (Typenbezeichnung: FILE*), welchen man dann wiederum für andere Operationen auf der Datei (z. B. Schreiben in die Datei mit der ebenfalls in der C-Standardbibliothek enthaltenen Funktion fprintf()) verwenden kann. *Wie* fopen() das bewerkstelligt, bleibt dem Programmierer verborgen. Eine Implementation dieses APIs findet sich etwa im Open-Source-Computerprogramm „musl"[23], einer Implementation der C-Standardbibliothek, welche andere Computerprogramme dann verwenden können, um Zugriff auf fopen() zu erhalten. „musl" greift bei der Implementation von fopen() sei-

6 ISO/IEC 2382:2015 Ziff. 2121308 und 2124351; LINFO-Definition Interface, http://www.linfo.org/interface.html (zuletzt abgerufen am 22.9.2020).
7 *Pilny,* GRUR Int 1990, 431, 432; s. auch die beispielhafte Auflistung in den Gesetzmaterialien zur CPRL, EP-Sitzungsdokument PE 136.025 endg. Teil B, S. 4.
8 LINFO-Definition Interface (Fn. 6).
9 https://www.ncoic.org/what-is-interoperability/ (zuletzt abgerufen am 22.9.2020); EG 10 CPRL.
10 So dann auch EG 10 CPRL und BT-Drs. 12/4022, S. 9.
11 *Byrne,* https://www.vice.com/en_us/article/3dakwk/oracle-vs-google-what-an-api-is-and-why-its-worth-fighting-for (zuletzt abgerufen am 22.9.2020); *Gamma/Helm u. a.,* Entwurfsmuster, 2011, S. 18 f.; *Lehmann,* in: Lehmann (Hrsg.), Die Europäische Richtlinie über den Schutz von Computerprogrammen, 2. Aufl. 1993, Kap. I.A, Rn. 7 Fn. 36; *Spindler,* in: Schricker/Loewenheim (Begr.), UrhR, 6. Aufl. 2020, § 69a Rn. 13.
12 EP-Sitzungsdokument PE 136.025 endg. Teil A, S. 26.
13 *Jaeger/Metzger,* Open Source Software, 4. Aufl. 2016, Rn. 21a; *Byrne* (Fn. 11); LINFO-Definition Interface (Fn. 6).
14 Mit historischem Aufriss *Pilny,* GRUR Int 1990, 431, 433.
15 *Byrne* (Fn. 11).
16 *Byrne* (Fn. 11); LINFO-Definition Interface (Fn. 6).
17 Technische Übersicht über diesen bei kompilierten Sprachen wie C und C++ als (dynamisches) „Linking" bezeichneten Prozess bei *Drysdale,* https://www.lurklurk.org/linkers/linkers.html (zuletzt abgerufen am 30.9.2020).
18 ISO/IEC 2382:2015, Ziff. 2122335.
19 *Gamma/Helm u. a.* (Fn. 11), S. 18.
20 *Gamma/Helm u. a.* (Fn. 11), S. 18 f.; *Stroustrup,* The C++ Programming Language, 4. Aufl. 2013, S. 859.
21 Eine Standardbibliothek ist eine Sammlung von Computerprogrammen, die mit einem standardisierten API mit einer bestimmten Programmiersprache mitgeliefert wird, *Stroustrup* (Fn. 20), S. 859 (für die C++-Standardbibliothek).
22 *Kernighan/Ritchie,* The C Programming Language, 2. Aufl. 1988, S. 242. Strenggenommen gehören speziell in C die Namen der Parameter (bei fopen() „filename" und „mode") nicht zur Funktionssignatur und damit auch nicht zum API, *Kernighan/Ritchie* (Fn. 22), S. 217 f.
23 Quellcode von „musl" Version 1.2.0, Datei src/stdio/fopen.c, Zeilen 6-31, https://git.musl-libc.org/cgit/musl/tree/src/stdio/fopen.c?h=v1.2.0#n6 (zuletzt abgerufen am 30.9.2020).

nerseits auf die Betriebssystemschnittstellen zur Interaktion mit der Festplatte zurück.[24] Andere Implementationen, z. B. die des GNU-Projekts (Programmbibliothek „Glibc") oder die von Microsoft („MSVCRT") implementieren fopen() unterschiedlich und solch unterschiedliche Lösungen können auch unterschiedlich performant sein[25]. Das oben vorgestellte API dagegen bleibt stets dasselbe. Die nachfolgende Abbildung fasst das Verhältnis der Begriffe bei klassischen APIs zusammen.



b) Web-APIs

Demgegenüber sind Web-APIs eine vergleichsweise junge Erscheinung, die mit dem Siegeszug der Smartphones einherging[26]. Für das Verständnis ihrer technischen Konzeption ist es jedoch erforderlich, auf einen sehr viel älteren Topos zurückzukommen, der auch schon bei Erlass der CPRL eine Rolle gespielt hat[27]: die Kommunikationsprotokolle. „(Kommunikations-)Protokoll" ist die Bezeichnung des Regelwerks für den Aufbau, die Durchführung und den Abbau einer Datenübertragungsverbindung.[28] Kommunikationsprotokolle dienen damit dem Informationsaustausch zwischen zwei potenziell weit voneinander entfernten Computern. Ähnlich wie die Deklarationen bei „klassischen" APIs stellt das Kommunikationsprotokoll eine Fassade dar, die ein Computerprogramm ansprechen und von der es bestimmte Ergebnisse erwarten kann. Um eine Gegenstelle anzusprechen, ist für den Programmierer die Kenntnis des auf der Gegenstelle ablaufenden Codes nicht erforderlich und wird meist auch nicht vorliegen. Es ist daher nicht richtig, wenn *Pilny* schreibt, dass die „Protokolle oft integrierter Teil eines Computerprogramms und nicht einfach von diesem zu unterscheiden"[29] seien.

Beispielsweise leisten die Betriebssysteme bei der Kommunikation über Netzwerke wie das Internet eine komplexe Abstraktion, die man mit dem OSI-Schichtmodell beschreibt, das den gestuften Einsatz einer ganzen Reihe von Kommunikationsprotokollen vorsieht[30] und das hier nicht im Detail ausgebreitet werden soll. Nach dem Prinzip der Aufgabenteilung bekommt der typische Programmierer einer Anwendung, die über ein Netzwerk kommuniziert, hiervon in aller Regel nichts mit. Er konzentriert sich allein auf das Kommunikationsprotokoll der obersten Schicht. Wichtiges Beispiel für ein solches Kommunikationsprotokoll ist das in RFC 2616[31] beschriebene „Hypertext Transfer Protocol" (HTTP), welches die Übertragung von Webseiten zwischen Computern regelt, die man in diesem Kontext auf der Anbieterseite als „Server" und auf der Leserseite als „Client" bezeichnet (§ 1.3 RFC 2616). HTTP wird auf der Server-Seite von einer Vielzahl verschiedener Computerprogramme angeboten, darunter sowohl Open-Source-Programme (z. B. „Apache httpd") wie auch proprietäre Software (z. B. von Microsoft).[32] Auf der Client-Seite arbeiten Webbrowser wie „Mozilla Firefox" und „Google Chrome".[33] Auf diese Weise können sehr unterschiedliche Computerprogramme miteinander kommunizieren. In allen Fällen aber ist HTTP nicht Teil des Computerprogramms, sondern gibt bloß das Regelwerk vor, nach dem die Computerprogramme arbeiten. So enthält HTTP zum Beispiel eine Regel, die besagt, dass ein Client Daten von einem Server abrufen kann, indem er an den Server eine Zeichenkette im folgenden Format sendet (§ 5.1.2 RFC 2616):

GET /pfad/zum/dokument.html HTTP/1.1

Host: www.diedomain.tld

Mit diesen Vorkenntnissen kann man sich den Web-APIs zuwenden. Ein Web-API lässt sich beschreiben als ein Web-Dienst eines Anbieters, den ein Computerprogramm maschinell ansprechen kann, um bestimmte Ereignisse auszulösen oder Informationen zu erhalten. Dabei kommt bei den Web-APIs genau wie bei herkömmlicher Kommunikation über Netzwerke auch ein Kommunikationsprotokoll zum Einsatz. Die technische Besonderheit solcher Web-API-Kommunikationsprotokolle liegt einzig darin, dass sie ein eigenes, stark auf die spezifische Aufgabe zugeschnittenes Protokoll auf Basis von HTTP (bzw. dem verschlüsselten HTTPS) nutzen[34] und damit noch eine Stufe weiter abstrahieren. Diese Protokolle sind in aller Regel nicht durch ein Gremium standardisiert, sondern werden vom Anbieter der jeweiligen Web-API einseitig vorgegeben. Beispielsweise kann man im Ticketsystem des populären Code-Hosting-Dienstes des US-Unternehmens GitHub mithilfe von deren Web-API neue Tickets mithilfe einer HTTP-Anfrage der folgenden Form erstellen:[35]

POST /repos/:eigentümer/:archivname/issues

Die konsequente Nutzung des sehr weit verbreiteten HTTP macht Web-APIs im Vergleich zu herkömmlichen Kommunikationsprotokollen für Programmierer leichter verwendbar und ermöglicht so die schnelle Entwicklung namentlich von Apps auf Smartphones.[36]

24 Quellcode von „musl" Version 1.2.0, Datei src/stdio/fopen.c, Zeile 21, https://git.musl-libc.org/cgit/musl/tree/src/stdio/fopen.c?h=v1.2.0#n21 (zuletzt abgerufen am 30.9.2020).
25 *Stroustrup* (Fn. 20), S. 859 für die C++-Standardbibliothek.
26 *Ashby/Jensen,* APIs for dummies, 3. Aufl. 2018, S. 6.
27 EP-Sitzungsdokument PE 136.025 endg. Teil B, S. 4.
28 *Pilny,* GRUR Int 1990, 431, 435. Eher abstrakt, aber letztlich inhaltsgleich Ziff. 2124467 ISO/IEC 2382:2015: „set of rules that determines the behavior of functional units in achieving communication".
29 *Pilny,* GRUR Int 1990, 431, 435.
30 https://de.wikipedia.org/wiki/OSI-Modell (zuletzt abgerufen am 21.9.2020).
31 RFC 2616 – Hypertext Transfer Protocol (HTTP/1.1).
32 Zu Marktanteilen siehe https://news.netcraft.com/archives/2019/02/28/february-2019-web-server-survey.html (zuletzt abgerufen am 28.9.2020).
33 Zu Marktanteilen siehe https://gs.statcounter.com/browser-market-share/desktop/germany/#monthly-200901-202007 (zuletzt abgerufen am 28.9.2020).
34 *Ashby/Jensen* (Fn. 26), S. 6.
35 https://developer.github.com/v3/issues/#create-an-issue (zuletzt abgerufen am 30.9.2020).
36 *Ashby/Jensen* (Fn. 26), S. 6.

III. Zum Begriff des Computerprogramms

Bevor die Schutzfähigkeit von Schnittstellen untersucht werden kann, ist der Schutzgegenstand der §§ 69a ff. UrhG überhaupt näher zu beleuchten. Diesen bilden Computerprogramme, § 69a Abs. 1 UrhG. Die in § 69a Abs. 1 UrhG enthaltene Definition erläutert letztlich „Computerprogramm" durch „Programm", was schwerlich zu überzeugen vermag.[37] Es handelt sich um eine inhaltsleere Tautologie. Diesbezüglich wird geltend gemacht, die Praxis könne auf eine nähere Definition mangels Relevanz verzichten[38]. Die seltene Thematisierung des Problems in der Rechtsprechung[39] scheint dieser Auffassung auf den ersten Blick Recht zu geben, doch ist dem schon früher unter Verweis auf ungeklärte Abgrenzungsfälle widersprochen worden[40]. Dem ist zuzustimmen, denn die fortschreitende Entwicklung der Informationstechnik erschwert zunehmend die Abgrenzung des nach § 69a UrhG schutzfähigen Materials von sonstiger Software[41]. Gerade für die Klärung der Frage, was noch Daten sind und was schon Computerprogramm, kann auf eine Definition des Begriffs nicht verzichtet werden. Die bisher ergangenen Entscheidungen des EuGH machen eine Definition des Schutzgegenstands ebenfalls nicht entbehrlich, denn sie verhalten sich zur Abgrenzung zwischen Computerprogramm und Daten nicht. Die seit Jahren diskutierte Frage, ob HTML-Code nun als Computerprogramm gem. §§ 69a ff. UrhG geschützt ist oder nicht[42], lässt sich mit dem vom EuGH in diesen Entscheidungen für das Vorliegen einer Ausdrucksform herangezogenen Kriterium, ob das Computerprogramm vervielfältigt werden kann[43], nicht beantworten, denn vervielfältigt wird ja gerade der HTML-Code. Es ist dieses Abgrenzungsproblem, das das OLG Frankfurt a. M. mit seiner ikonischen Formulierung zu „digitalen Datenanhäufungen"[44] richtig benennt.

Eine Definition ist also erforderlich. Die Rechtsprechung definiert das Computerprogramm (ohne dies jemals dem EuGH vorgelegt zu haben) unter Rückgriff auf die WIPO-Mustervorschriften[45] und DIN 44300 Teil 4 Nr. 4.1.9 als „Folge von Befehlen, die nach Aufnahme in einen maschinenlesbaren Träger fähig sind zu bewirken, dass eine Maschine mit informationsverarbeitenden Fähigkeiten eine bestimmte Funktion oder Aufgabe oder ein bestimmtes Ergebnis anzeigt, ausführt oder erzielt"[46]. Diese Definition hat in der Literatur in dieser oder ähnlicher Form weitgehend Zuspruch gefunden[47] und entspricht auch den Vorstellungen der damaligen EG-Kommission[48]. Bedenken sind vor allem deshalb angemeldet worden, weil in Antizipation des damals erwarteten technischen Fortschritts auf eine womöglich schnell veraltende Definition in der CPRL bewusst verzichtet wurde[49] und DIN 44300 wie auch die WIPO-Mustervorschriften aus der informationstechnischen Steinzeit (1988 respektive 1977) stammen[50]. Überdies wurde DIN 44300 mittlerweile zurückgezogen.[51] Jedoch findet sich diese Definition weiter inhaltsgleich in der aktuell gültigen, von 2015 datierenden und in der Jurisprudenz bisher unbeachteten Nachfolgenorm ISO/IEC 2382:2015 in Ziff. 2121372. Sie ist technisch auch in Zeiten der Smartphone- und App-Ära weiterhin zutreffend[52]. Die technischen Grundlagen sind nämlich unverändert geblieben. Heute wie damals basieren Computer auf den von *Turing*[53] und *von Neumann*[54] aufgestellten Theorien.[55] In den Smartphones und Cloud-Servern unserer Zeit stecken noch immer *von Neumann*'sche Computer.[56] Die damaligen Befürchtungen haben sich also nicht bewahrheitet.[57] Bloß neue Programmiertechniken, die diese technischen Grundlagen unverändert lassen, sind mehr sozialer als technischer Natur und kein Grund, die Definition des Computerprogramms zu verändern.[58] Soweit etwa externe XSD-Dateien (eine Sonderform von XML) zur Laufzeit durch ein Computerprogramm verarbeitet werden[59], werden diese daher nicht Teil des Computerprogramms, sondern bleiben Konfigurationsdateien und damit Daten. Diese Überlegung wird dadurch bestätigt, dass man dieselbe Datei auch durch ein anderes ggf. selbst geschriebenes Computerprogramm verarbeiten lassen könnte. Da diese bewährte Definition überdies weitestgehend dem internationalen Verständnis entspricht[60], sollte von ihr nicht abgewichen werden. Einzig die behutsame Modernisierung der Bezeichnung „Maschine mit informationsverarbeitenden Fähigkeiten" zum weit weniger sperrigen Wort „Computer" empfiehlt sich.

IV. Der Schutz der Schnittstellen im Allgemeinen

1. Schnittstellen als schutzfähige Programmteile

Nach dieser grundlegenden Erkenntnis kann man sich nun der Frage zuwenden, ob auch die Schnittstellen gem.

37 *Marly,* GRUR 2012, 773, 774.
38 *Spindler,* in: Schricker/Loewenheim (Fn. 11), § 69a Rn. 2; *Dreier/Goldrian/Betten,* GRUR Int 1989, 200, 201.
39 Die letzte höchstrichterliche Erwähnung offenbar in BAG, 24.3.2011 – 2 AZR 282/10, NZA 2011, 1029, 1031, also vor neun Jahren.
40 *Ohst,* Computerprogramm und Datenbank, 2003, S. 17; ähnl. *Marly,* GRUR 2012, 773, 776: keine ontologische Frage.
41 S. nur die Abgrenzungsversuche bei *Nebel/Stiemerling,* CR 2016, 61, 65 ff.
42 Vgl. zum Streit mit zahlreichen Nachw. *Grützmacher,* in: Wandtke/Bullinger (Hrsg.), UrhR, 5. Aufl. 2019, § 69a Rn. 19.
43 EuGH, 22.12.2010 – C-393/09, K&R 2011, 105, Rn. 35 – BSA/Kulturministerium; EuGH, 2.5.2012 – C-406/10, WRP 2012, 802, Rn. 35 ff. – SAS Institute.
44 OLG Frankfurt am Main, 22.3.2005 – 11 U 64/04, GRUR-RR 2005, 299, 300 – Online-Stellenmarkt.
45 Abgedruckt in GRUR 1979, 300, 306 ff.
46 BGH, 9.5.1985 – I ZR 52/83, BGHZ 94, 276, 283 – Inkasso-Programm; BAG, 24.3.2011 – 2 AZR 282/10, NZA 2011, 1029, 1031; OLG Köln, 8.4.2005 – 6 U 194/04, K&R 2006, 43, 45; OLG Frankfurt am Main, 22.3.2005 – 11 U 64/04, GRUR-RR 2005, 299, 300 – Online-Stellenmarkt.
47 HM, s. nur *Spindler,* in: Schricker/Loewenheim (Fn. 11), § 69a Rn. 2; *Wiebe,* in: Spindler/Schuster (Hrsg.), Recht der elektronischen Medien, 4. Aufl. 2019, § 69a Rn. 3; *Czychowski,* in: Fromm/Nordemann (Begr.), UrhR, 12. Aufl. 2018, § 69a Rn. 5; *Dreier,* in: Dreier/Schulze (Hrsg.), UrhG, 6. Aufl. 2018, § 69a Rn. 12; *Schippel,* MMR 2020, 445, 446; aA *Nebel/Stiemerling,* CR 2016, 61, 65: Öffnung für moderne Programmiertechniken. Dazu sogleich im Text.
48 KOM (1988) 816 endg., § 1.1.
49 KOM (1988) 816 endg., § 1.1; *Dreier/Goldrian/Betten,* GRUR Int 1989, 200, 201.
50 Vgl. *Nebel/Stiemerling,* CR 2016, 61, 62 f.; *Marly,* GRUR 2012, 773, 774.
51 https://www.beuth.de/de/norm/din-44300-1/1401672 (zuletzt abgerufen am 28.9.2020).
52 *Marly,* GRUR 2012, 773, 774; *Marly,* GRUR 2011, 204.
53 *Turing,* Proc. Lond. Math. Soc. 1937, 230, 231 ff.
54 *v. Neumann,* IEEE Ann. Hist. Comput. 4/15 (1993), 27, 28 ff. (Nachdruck).
55 Zu diesen Theorien *Horns,* GRUR 2001, 1, 5 ff.
56 *Dumas,* Computer Architecture, 2. Aufl. 2016, S. 7 und 17 f.
57 *Marly,* GRUR 2012, 773, 774: „krasse Fehleinschätzung".
58 A.A. *Nebel/Stiemerling,* CR 2016, 61, 65; Schon *Koch,* GRUR 2000, 191, 195 ff. war der Meinung, die damals neue Technik der objektorientierten Programmierung mit dieser Definition erfassen zu können.
59 So ein Beispiel bei *Nebel/Stiemerling,* CR 2016, 61, 65 f.
60 Rechtsvergleichend *Ohst* (Fn. 40), S. 22 ff; *Marly,* GRUR 2012, 773, 775.

§§ 69a ff. UrhG geschützt sind. Dazu ist das Gesetz auszulegen. Die Schnittstellen werden erwähnt im Wortlaut des § 69a Abs. 2 S. 2. Aus der Grammatik des mit „einschließlich" beginnenden zweiten Halbsatzes dieser Norm folgt, dass ein Beispiel für die im ersten Halbsatz erwähnten „Ideen und Grundsätze" spezifiziert wird, die „einem Element eines Computerprogramms zugrunde liegen". Die beiden Halbsätze nutzen eine ähnlich aufgebaute Satzkonstruktion unter Ersetzung von „Element eines Computerprogramms" durch „Schnittstellen" bei sonst gleichbleibenden Wörtern.[61] Im ersten Halbsatz ist ungeschützt, was als Ideen und Grundsätze einem Element eines Computerprogramms zugrunde liegt; im zweiten Halbsatz ist ungeschützt, was als Ideen und Grundsätze den Schnittstellen zugrunde liegt. Der Wortlaut des § 69a Abs. 2 S. 2 UrhG beantwortet die Frage also dahingehend, dass die Schnittstellen als Teil des Computerprogramms angesehen und folglich für schutzfähig erachtet werden. Weil es sich bei dieser Norm um eine wortgetreue Übernahme aus Art. 1 Abs. 2 S. 2 CPRL handelt, ergibt sich hierfür nichts Abweichendes. Im Gegenteil formuliert es Erwägungsgrund 11 CPRL noch einmal etwas anders, aber nicht minder eindeutig. Erwägungsgrund 10 CPRL nennt die Schnittstellen gar ausdrücklich „Teile des Programms".

Die systematische Betrachtung verläuft, wohl auch wegen der Sonderstellung der Computerprogramme im Urheberrecht, ergebnislos. Aufschlussreich ist demgegenüber ein teleologischer Aspekt. Zweck der CPRL und der sie umsetzenden §§ 69a ff. UrhG ist es, Computerprogramme dem urheberrechtlichen Schutz zu unterstellen (Erwägungsgründe 3 und 6 sowie Art. 1 Abs. 1 CPRL). Die Richtlinie verfolgt zwar darüber hinaus auch noch das Ziel der Ermöglichung von Interoperabilität zwischen Computerprogrammen verschiedener Hersteller (Erwägungsgründe 10 und 15 CPRL), doch darf das wegen des ersteren Ziels nicht so weit gehen, dass ganze Computerprogramme schutzlos werden. Nun ist es aber möglich, ein ganzes Computerprogramm als Schnittstelle zu betrachten[62], womit man bei einer Schutzunfähigkeit von Schnittstellen zu genau diesem Ergebnis gelangte. Weil Interoperabilität eben nicht nur heißt, das neue Computerprogramm mit dem alten zu verbinden (z. B. Plug-In), sondern auch, dieses durch Nachbildung seiner Schnittstellen vollständig zu ersetzen und mit ihm zu konkurrieren[63], wäre auf diese Weise der urheberrechtliche Schutzumfang des ursprünglichen Computerprogramms in das Belieben von Wettbewerbern gestellt[64]. Man hat dieses Verständnis des Computerprogramms als Schnittstellenaneinanderreihung als „überspitzt" kritisiert[65], jedoch ist es das keineswegs. Es ist nämlich nicht so, dass ein Computerprogramm nur „offizielle" Schnittstellen (in geringer Zahl) aufweisen würde, die ein interoperables Computerprogramm nachbilden müsste, sondern es sind gerade die zahlreichen Stellen, die der ursprüngliche Hersteller nicht als Anknüpfungspunkte vorgesehen hat, die für eine erfolgreiche Interoperabilität repliziert werden müssen.[66] Ein frühes Beispiel aus diesem Bereich bildet das IBM-BIOS[67], welches eigentlich nach dem Willen des Herstellers von normalen Computerprogrammen nicht benutzt werden sollte. Genau dies geschah, und PC-Hersteller mussten all die „inoffiziellen" Schnittstellen des IBM-BIOS nachbauen. Ein aktuelleres Beispiel bildet der Spielkonsolen-Emulator „Dolphin", denn ein fehlerfreier Ablauf der für die ursprünglichen Spielkonsolen geschriebenen Spiele ist nur gewährleistet, wenn auch mit dem Emulator all die Tricks möglich sind, die der Spielkonsolen-Hersteller zwar nicht beabsichtigt hatte, die aber trotzdem genutzt wurden[68]. Aus teleologischer Sicht muss deshalb von einer Schutzfähigkeit der Schnittstellen ausgegangen werden.

In historischer Hinsicht gilt es zu beachten, dass schon der ursprüngliche Kommissionsentwurf die Schnittstellen ausdrücklich als Programmteile begriff[69] und damit Erwägungen, Schnittstellen vom Urheberrechtsschutz auszunehmen, eine Absage erteilte[70]. Vom Schutz ausgenommen werden sollte danach lediglich die „Spezifizierung von Schnittstellen" in Art. 1 Abs. 3 S. 2 des Entwurfs. Das hat das Parlament dann auf Antrag des Rechtsausschusses[71] in eine Fassung geändert, die in den verschiedenen Sprachfassungen subtil unterschiedlich war und dahingehend missverstanden werden konnte, dass man die Schnittstellen insgesamt den ungeschützten Ideen und Grundsätzen zuordnen wolle[72]. Aus dem Studium der Begründung wird jedoch klar, dass dem Parlament nur der Begriff der „Spezifizierung" zu eng war, insgesamt aber eine vermittelnde Position zwischen Schutzfähigkeit und Schutzunfähigkeit eingenommen werden sollte[73]. Auch wäre das im Kontext mit anderen Änderungen unlogisch gewesen.[74] Der Rat hat die missverständliche Formulierung später in diesem Sinne redaktionell bereinigt[75]. Im europäischen Gesetzgebungsverfahren haben sich demnach alle beteiligten Organe aktiv gegen eine vollständige Schutzunfähigkeit von Schnittstellen ausgesprochen.[76]

Insgesamt spricht also vieles dafür, die Schnittstellen als schutzfähigen Programmteil einzuordnen. Das Ziel der Interoperabilität wird dann in erster Linie dadurch erreicht, dass, wie § 69a Abs. 2 Satz 2 UrhG es ausdrückt, die „den Schnittstellen zugrundeliegenden Ideen und Grundsätze" nicht geschützt werden. Diese nicht schutzfähigen Elemente pflegt man als „Schnittstellenspezifikation" zu bezeichnen[77], doch ist dieser Begriff missverständlich und vor dem Hintergrund seiner ausdrücklichen Ablehnung durch das europäische Parlament insgesamt unglücklich. Dennoch soll er hier wegen seiner Bekanntheit beibehalten werden, doch nicht ohne Klarstellung seiner Bedeutung. Der Begriff er-

61 Abgesehen von der bloß orthographischen Abweichung der Verwendung eines Wortabstands bei „zugrunde liegen", aber nicht bei „zugrundeliegenden". Die CPRL schreibt interessanterweise beides einheitlich mit Abstand.
62 *Czarnota/Hart,* Legal Protection of Computer Programs in Europe, 1991, S. 37.
63 *Falker,* in: Taeger (Hrsg.), Die Macht der Daten und der Algorithmen, 2019, S. 595, 601; *Dreier,* CR 1991, 577, 582; *Kindermann,* CR 1990, 638.
64 *Czarnota/Hart* (Fn. 62), S. 37 f.
65 *Schulte,* CR 1992, 648, 649 Fn. 4.
66 *Kindermann,* CR 1990, 638, 639 f; *Czarnota/Hart* (Fn. 62), S. 40.
67 Zum folgenden *Vinje,* GRUR Int 1992, 250, 252 m. w. N.
68 S. z. B. https://dolphin-emu.org/blog/3S/ (zuletzt abgerufen am 22.9.2020) und https://dolphin-emu.org/blog/48/ (zuletzt abgerufen am 22.9.2020).
69 KOM (1988) 816 endg., § 3.10.
70 *Lehmann,* NJW 1991, 2112, 2113.
71 EP-Sitzungsdokument PE 136.025 endg. Teil A, S. 6 (Änderungsantrag Nr. 3).
72 *Czarnota/Hart* (Fn. 62), S. 41.
73 EP-Sitzungsdokument PE 136.025 endg. Teil B, S. 4 f.
74 *Czarnota/Hart* (Fn. 62), S. 41.
75 Ratsdokument C3-0018/91 = EP-Sitzungsdokument PE 147.961, S. 8.
76 *Czarnota/Hart* (Fn. 62), S. 42.
77 *Grützmacher,* CR 2016, 485, 494; *Vinje,* GRUR Int 1992, 250, 254; *Lehmann,* CR 1989, 1057, 1059; *Falker,* in: Taeger (Fn. 63), S. 600; *Czychowski,* in: Fromm/Nordemann (Fn. 47), § 69a Rn. 32.

weckt den Eindruck, als handle es sich um ein Dokument, das die Funktionsweise der Schnittstelle beschreibe.[78] Tatsächlich geht es aber nicht um ein solches Dokument – das vielfach, gerade bei „inoffiziellen" Schnittstellen, auch gar nicht existieren wird –, sondern um die Funktionsweise selbst. Wird diese Funktionsweise in einem Dokument beschrieben, so sind die im Dokument enthaltenen Ideen mit den in der Schnittstelle enthaltenen Ideen identisch und gem. § 69a Abs. 2 S. 2 UrhG ungeschützt. Sie bilden die „Schnittstellenspezifikation". Das Dokument selbst genießt, anders als es die Rede von der Schutzunfähigkeit von „Spezifikationen" anzudeuten scheint, selbstverständlich Schutz als Sprachwerk gem. § 2 Abs. 1 Nr. 1 UrhG[79]. § 69a Abs. 2 S. 2 UrhG gibt damit bloß einen allgemeinen urheberrechtlichen Grundsatz wider, nämlich die auch in Deutschland geltende Trennung von Form und Inhalt[80]. Die Norm ist der Sache nach überflüssig.

2. Die Abgrenzung von Schnittstelle und „Schnittstellenspezifikation"

Bis hierher war es nicht nötig, den Schnittstellenbegriff mit einem konkreten technischen Inhalt zu füllen. Das gefundene Zwischenergebnis – Schnittstellen sind schutzfähige Programmteile – ist für sich genommen aber nicht nützlich, weil es die Frage der Zuordnung konkreter technischer Phänomene zur geschützten Schnittstelle einerseits und zur ungeschützten „Schnittstellenspezifikation" andererseits offen lässt. Wie sich zeigen wird, kann hierfür weder auf einen rein technischen Bedeutungsgehalt (unten a) noch auf eine einheitliche juristische Betrachtung (unten b) zurückgegriffen werden.

a) Untauglicher technischer Schnittstellenbegriff

Es liegt nahe, als geschützte Schnittstelle nun das zu erfassen, was ein Techniker als Schnittstelle verstehen würde, d. h. die für die Interoperabilität notwendigen Teile. Ein solches Verständnis ist aus zwei Gründen abzulehnen. Erstens verlangen die Grammatik der Norm und Erwägungsgrund 10 CPRL, wie gezeigt, dass die Schnittstellen Programmteile sein müssen und deshalb die Computerprogramm-Definition erfüllen müssen, was bei diversen Schnittstellen (z. B. USB-Buchsen) nicht der Fall ist. Zweitens widerspräche es dem Telos der Interoperabilität, was auch von Seiten der Programmierer geltend gemacht wird[81]. Diese Bedenken berücksichtigt das Gesetz an anderer Stelle durch die Freistellung der Ideen und Grundsätze, d. h. der „Schnittstellenspezifikation" (Erwägungsgrund 11 CPRL). Daher kann man den Begriff der Schnittstelle im Rechtssinne nicht mit dem im technischen Sinne gleichsetzen. Der rechtliche Schnittstellenbegriff entspricht in der Ausdrucksweise der Techniker vielmehr der *Implementation* einer Schnittstelle. Diese sprachliche Divergenz ist in der Diskussion mit Vertretern der Programmierer unbedingt zu berücksichtigen, um gegenseitige Missverständnisse zu verhindern.

b) Unausweichliche Code-Übernahmen und die Unmöglichkeit einer allgemeinen Abgrenzung

Es irrt die Kommission, wenn sie meint, die Herstellung von Interoperabilität erfordere *nie* die Wiedergabe von Code[82]. Es existieren nämlich Schnittstellen, bei denen zur Herstellung von Interoperabilität genau das nötig ist.[83] Genau solche Schnittstellen sind Gegenstand des Java-Verfahrens zwischen Google und Oracle. Bei solchen Schnittstellen geraten die beiden Ziele der CPRL miteinander in Konflikt. Dieser Zielkonflikt ist aber kein Grund, das zuvor gefundene Verhältnis von Schnittstelle und „Schnittstellenspezifikation" insgesamt in Frage zu stellen, wenn eine andere Lösung den Konflikt ebenfalls auflösen kann. Schon früh im Gesetzgebungsverfahren ist nämlich aufgefallen, dass es zahlreiche, recht unterschiedliche Schnittstellenarten gibt[84]. Für alle diese unterschiedlichen Schnittstellenarten die Trennung zwischen der Schnittstelle und ihrer „Spezifikation", also zwischen Ausdruck und Idee bzw. Form und Inhalt, einheitlich vornehmen zu wollen, kommt der Quadratur des Kreises gleich. Dafür ist die Vielfalt unterschiedlicher Schnittstellen einfach zu groß.[85] Sie ist auch der Grund dafür, weshalb der Gesetzgeber auf einen generellen Schutzausschluss verzichtet hat: die Reichweite einer solchen Ausschlussbestimmung wäre kaum zu ermitteln.[86] Hieraus ist richtigerweise die Konsequenz zu ziehen, dass pauschale Aussagen über die Trennung von Schnittstelle und „Schnittstellenspezifikation" untunlich sind. Weder lässt sich für alle Schnittstellen pauschal sagen, sie müssten aus Wettbewerbsgründen (Interoperabilität) vom Schutz frei bleiben[87], noch kann man annehmen, sie erfüllten ausnahmslos die Computerprogramm-Definition und müssten geschützt werden[88]. Auch die im Java-Verfahren von den Programmierern geltend gemachten Befürchtungen[89] können daher immer nur anhand der konkret in Rede stehenden Schnittstelle beurteilt werden.

V. Zwischenfazit

Der erste Teil hat technische Grundlagen verschiedener Schnittstellenarten analysiert, dabei den Unterschied zwischen Regelwerk und Implementation aufgezeigt und herausgearbeitet, dass allgemeingültige Aussagen über die Schutzfähigkeit aller Arten von Schnittstellen nicht möglich sind. Auch hat sich gezeigt, dass die althergebrachte Definition des Computerprogramms weiterhin Gültigkeit genießt. Bei den APIs schließlich müssen „klassische" APIs wie die im Java-Verfahren streitgegenständlichen unterschieden werden von den neueren Web-APIs. Im zweiten Teil soll ein näherer Blick auf diese beiden Arten von APIs geworfen und ein neuer Ansatz zur Prüfung von deren Schutzfähigkeit vorgestellt werden.

78 Entsprechende Verwechslung bei *Czarnota/Hart* (Fn. 62), S. 38.
79 *Czarnota/Hart* (Fn. 62), S. 38.
80 Amtl. Begr. BT-Drs. 12/4022, S. 9. Ebenso SEK (1991) 87 endg, § 4.2.a, EG 11 CPRL und *Dreier,* CR 1991, 577, 578; wohl auch *Pilny,* GRUR Int 1990, 431, 439.
81 *Duan,* https://arstechnica.com/tech-policy/2020/01/oracle-copied-amazons-api-was-that-copyright-infringement/ (zuletzt abgerufen am 28.9.2020); Amici-Curiae-Brief von 83 Informatikern v. 13.1.2020 im Verfahren Google v. Oracle, Supreme Court of the United States No. 18-956, S. 17 ff.
82 KOM (1989) 816 endg., § 3.11 dürfte so zu verstehen sein.
83 *Dreier,* CR 1991, 577, 583; *Pilny,* GRUR Int 1990, 431, 440; *Schulte,* CR 1992, 648, 650; *Vinje,* GRUR Int 1992, 250, 259.
84 EP-Sitzungsdokument PE 136.025 endg. Teil B, S. 4.
85 Vgl. *Pilny,* GRUR Int 1990, 431, 432.
86 *Schulte,* CR 1992, 648, 649.
87 So aber *Grützmacher,* CR 2016, 485, 494, unter Verweis auf *Karl,* Der urheberrechtliche Schutzbereich von Computerprogrammen, 2009, S. 238.
88 So aber *Schulte,* CR 1992, 648, 649.
89 S. Fn. 81.