Das Sachverständigengutachten über die Urheberrechtsschutzfähigkeit von Computer-Programmen

Ulrich Eckhardt*

Zusammenfassung

Es wird das Problem der Feststellung der Urheberrechtsschutzfähigkeit von Computer-Programmen aus der Sicht des Sachverständigen behandelt. Nachdem der Bundesgerichtshof in dem „Inkassoprogramm"-Urteil von 1985 die Frage der Schutzfähigkeit prinzipiell bejaht hat, muß man nunmehr feststellen, daß bei den Betroffenen eine erhebliche Rechtsunsicherheit besteht, so daß die Situation keineswegs als befriedigend zu bezeichnen ist.

1 Einleitung

Gegenstand dieses Beitrags ist allein die Frage der Feststellung der Urheberrechtsfähigkeit von Computerprogrammen. Das Problem der Feststellung des Verletzungstatbestandes wird hier nicht behandelt werden. Die letztgenannte Fragestellung ist die einfachere, sie läßt sich im Prinzip mechanisch, also durch Anwendung eines Computerprogramms, lösen.

Bei der Behandlung der Urheberrechtsfähigkeit von Computerprogrammen werde ich mich auf professionell erstellte und genutzte Programme für Großrechner konzentrieren. Auf diesem Gebiet verfüge ich über langjährige Erfahrungen. Zwar habe ich mich auch mit Programmen für kleine und privat genutzte Rechner befaßt, insbesondere in meiner Eigenschaft als Vorstandsvorsitzender der Hamburger Microcomputer Hochschulgruppe, einer der größten Amateurgruppen für Microcomputer in Europa, jedoch ist auf diesem Gebiet die Situation eher chaotisch und gekennzeichnet durch ein bemerkenswertes Unrechtsbewußtsein bei den Verletzern und durch Resignation bei den Herstellern.

Weiterhin möchte ich hier das Computerprogramm und seine Vorläuferphasen sowie das Begleitmaterial im Sinne des „Inkassoprogramm"-Urteils des Bundesgerichtshofs ([11]; vgl. die Literaturangaben am Ende der Arbeit) als eine Einheit betrachten. Das Problem mehrerer Urheber und die gegenseitigen Abhängigkeiten der einzelnen Werkteile sollen hier nicht berührt werden.

Mit dem Gebiet der Urheberrechtsfähigkeit von Computerprogrammen komme ich in dreifacher Hinsicht in Berührung. Als Produzent von Computerprogrammen bin ich an einer klaren Rechtssituation interessiert, als Nutzer von Computerprogrammen werde ich durch Urheberrechtsprobleme Dritter unter Umständen unangenehm überrascht. Schließlich bin ich gelegentlich als Gutachter oder Berater aufgetreten.

*Prof. Dr. Ulrich Eckhardt, Institut für Angewandte Mathematik der Universität Hamburg, Bundesstraße 55, D-2000 Hamburg 13
Vortrag auf dem Symposium „Softwareschutz" der DGIR am 9. Dezember 1988 in Frankfurt

2 Begriffsbestimmungen

Ein Computerprogramm besteht aus einer Folge von Anweisungen in einer künstlichen Programmiersprache. Es gibt zahlreiche solche Sprachen. Eine Programmiersprache besitzt eine sehr stark eingeschränkte Syntax und einen „Wortschatz“ von etwa hundert Befehlsworten, die durchweg eine wohldefinierte und eng begrenzte Bedeutung haben. Diese spartanischen Sprachmittel bedeuten keinesfalls, daß durch solche Programmiersprachen nur Triviales ausdrückbar sei und daß die schöpferischen Möglichkeiten beschränkt seien. Ähnlich wie in der Sprache der Mathematik (aber auch der Rechtswissenschaften) werden hier Begriffe durch Beschränkung des Bedeutungsumfangs präzisiert oder komplizierte Sachverhalte durch eine formelhafte Wendung ausgedrückt.

Jede Programmiersprache erlaubt es zudem, die vorhandenen Möglichkeiten durch Hinzufügen eigener Befehle zu erweitern. Diese müssen nach den für die betreffende Sprache aufgestellten Regeln exakt definiert werden. Die zusätzlichen Befehle an den Rechner und die vom Programm zu verarbeitenden Größen (Daten) werden durch relativ frei wählbare Namen identifiziert.

Ein Programm kann zudem noch erklärende Zusätze in natürlicher Sprache erhalten, die einem menschlichen Leser das Verstehen des Programms ermöglichen. Derartige Kommentare werden vom Computer ignoriert. Der Autor des Programms hat erhebliche Freiheiten bei ihrer Gestaltung.

Bei dem Prozeß der Programmproduktion entstehen üblicherweise noch Vorläuferprodukte wie das Pflichtenheft, das in einer frühen Planungsstufe die Problemstellung und die Methoden zur Lösung des Problems beschreibt, die Ablaufdiagramme und Datenflußpläne, die z.T. mit graphischen Mitteln auf einer mittleren Abstraktionsebene die Problemlösung in allgemeinen Begriffen veranschaulichen und bei größeren Datenmengen darstellen, auf welche Weise diese in welchen Programmzuständen modifiziert werden (vgl. das „Inkassoprogramm“-Urteil des Bundesgerichtshofs [11]).

Für die Anwendung des Programms benötigt man in der Regel Begleitmaterial, zum Beispiel Installationsanweisungen und Benutzungshandbücher. Hierbei handelt es sich um Gebrauchsanweisungen, in denen abstrakt oder beispielhaft die Benutzung des Programms beschrieben wird. Dieses Material kann sich aller Ausdrucksformen menschlicher Kommunikation bedienen.

In den Anfangsjahren der Datenverarbeitung – hier in Deutschland bis zum Ende der sechziger Jahre – wurden Programme häufig so erstellt, daß allein der Programmierer sie verstehen konnte. Dies war darauf zurückzuführen, daß durch die begrenzten Möglichkeiten, die die damaligen Anlagen boten, das Programmieren äußerst trickreich war und daß zu der Zeit die Programmierer durchweg Amateure waren ohne eine reguläre, allgemein anerkannte Ausbildung. Auf diese Weise entstanden in vielen Fällen nicht mehr tolerierbare Abhängigkeiten. Programme waren nicht modifizierbar und konnten demgemäß auch nicht ohne erheblichen Aufwand geänderten Anwenderwünschen oder technischen Entwicklungen angepaßt werden. Die äußerst rasch fallenden Kosten für Rechner, die enorme Steigerung ihrer Leistungsfähigkeit sowie die gleichzeitig ansteigenden Kosten für Programme und deren Wartung und Pflege führten zu der Notwendigkeit, Programme übertragbar und modifizierbar zu konzipieren. Die Methoden hierzu werden unter dem Begriff „strukturierte Programmierung“ zusammengefaßt. Hierunter versteht man einen mehr oder weniger verbindlichen Satz von Regeln, Normen und Methoden, die die Gestaltungsfreiheit des Programmierers einschränken, dafür aber zu einem vorhersagbaren Resultat der Programmproduktion führen und diese nachvollziehbar machen. Zu den verbreitetsten Methoden zur Strukturierung von Programmen gehören neben den rein textgestalterischen Möglichkeiten der Editierung:

- Die ausführliche Kommentierung aller wesentlichen Vorgänge im Programm,
- Die Modularisierung, d.h. die Zerlegung der durch das Programm zu lösenden Aufgabe in wohldefinierte Teilaufgaben, deren gegenseitige Abhängigkeiten exakt beschreibbar sind (z.B. durch Schnittstellendefinitionen),
- Die Hierarchisierung, d.h. die Definition von globalen Grobstrukturen, die wiederum in Feinstrukturen aufgelöst werden.

In der letzten Zeit spielen bei der Programmproduktion – von der Erstellung des Pflichtenheftes bis zur Codierung – Hilfsprogramme zur Unterstützung des Programmierers eine wesentliche Rolle. Diese sogenannten Programmierwerkzeuge erlauben es, ganze Programme oder Programmteile automatisch zu erzeugen. Die einfachsten derartigen Werkzeuge sind die sogenannten Editoren, die das Schreiben und Modifizieren der Programme unterstützen.

Ziel der Programmproduktion ist es, ein Produkt zu erzeugen, das die Eigenschaften der Korrektheit, Effizienz, Robustheit und Wartbarkeit hat und das zusätzlich den Anwendungszweck ergonomisch erreicht (vgl. das „Baustatikprogramm“-Urteil des OLG Frankfurt [12] und [13]).

Die Gesamtheit aller dem normalen Benutzer bei ordnungsgemäßem Gebrauch zugänglichen Wirkungen des Programms nennt man die Benutzeroberfläche. Diese besteht insbesondere aus Bildschirmmasken, Ausgabeformaten, Eingabemenüs usw. Hier hat der Ersteller des Programms eine große Vielfalt von Möglichkeiten zur Gestaltung, die leider nicht immer genutzt werden, wie etwa der Vergleich eines professionellen Textverarbeitungsprogramms mit einem Spielprogramm lehrt (vgl. Carroll [2]).

Es gibt in der Informatik zahlreiche Vorschläge zur zahlenmäßigen Erfassung der „Kompliziertheit“ eines Programmprodukts. Diese Komplexitätsmaße können einen Anhalt geben für den Wert eines Programms oder den Aufwand, der zu seiner Erstellung notwendig war. Das einfachste solche Maß ist die Anzahl der Programmzeilen. Diese Zahl ist allerdings nicht sehr signifikant (vgl. Conte, Dunsmore, Shen [3]). Ähnlich wird versucht, die Qualität eines Programmprodukts objektiv zu erfassen (vgl. etwa Hausen, Müllerburg, Schmidt [4]).

Für die urheberrechtliche Einschätzung eines Programms ist die Unterteilung desselben in Verarbeitungsteil und Ein/Ausgabeteil nützlich. Da der letzgenannte Teil vom verwendeten Rechner und den Peripheriegeräten (Drucker, Plattenlaufwerke usw.) abhängt, ist er stärkeren Veränderungen unterworfen als der Verarbeitungsteil, der nur von dem zu lösenden Problem abhängt. Bei ordnungsgemäßem Programmieren sind diese beiden Teile im Programm sauber voneinander getrennt. Bei Übernahme eines Programms wird man den Ein/Ausgabeteil in der Regel ändern müssen, um gegebenenfalls das Programm an eine neue Maschinenkonfiguration anzupassen (vgl. das „Baustatikprogramm“-Urteil des OLG Frankfurt [12] und [13]).

Je nach Anwendungsbereich unterscheidet man verschiedene Klassen von Programmen und Programmierern. Ich nenne drei Beispiele:

- *Systemprogramme.* Diese Programme steuern – für den Benutzer im allgemeinen nicht direkt bemerkbar – die Abläufe des oder der Benutzerprogramme im Rechner. Zum Beispiel lösen gewisse Systemprogramme einen Druckbefehl im Benutzerprogramm in zahlreiche Anweisungen und Unterprogramme auf, die – unter Umständen parallel zu anderen Vorgängen im Rechner – den Drucker veranlassen, die fraglichen Daten richtig in der gewünschten Form auszudrucken.

- *Anwenderprogramme.* Hierbei handelt es sich um Programme, die ein gewisses Problem des Benutzers lösen. Sie sind – im Gegensatz zu den Systemprogrammen – in der Regel maschinenunabhängig geschrieben und benutzen – zumindest an der Benutzeroberfläche – eine dem Anwender vertraute Terminologie. Bei einem Programm zur Finanzbuchhaltung beispielsweise ist nicht von „Dateien“ die Rede, sondern von „Büchern“ usw.

- *Echtzeitprogramme.* Diese Programme steuern und regeln einen außerhalb des Rechners ablaufenden Vorgang. Hierbei ist typisch, daß der externe Vorgang die Ablaufgeschwindigkeit bestimmt.

Die unterschiedlichen Klassen von Programmen unterscheiden sich z.T. erheblich voneinander, desgleichen natürlich auch die Methoden der Programmproduktion, die benutzten Sprachen, die benötigten Fachkenntnisse usw.

Ein besonders wichtiger Gesichtspunkt zur Einschätzung von Computerprogrammen ist angesichts der rasanten technischen Entwicklung deren Entstehungszeit. Am Regionalen Rechenzentrum der Universität Hamburg hat sich in den letzten dreißig Jahren die Rechengeschwindigkeit von 300 auf 11 Millionen Rechenoperationen pro Sekunde gesteigert und die Hauptspeichergröße von 10 kByte (entsprechend etwa 3 1/2 Schreibmaschinenseiten DIN A 4) auf 48 MByte (ca. 15 Aktenordner). Diese beiden Maßzahlen für die Leistungsfähigkeit eines Rechners haben sich demgemäß in der betrachteten Zeit im Mittel in je drei Jahren verdoppelt. Ein solches Entwicklungstempo ist ohne Beispiel in der Geschichte oder in anderen Zweigen der Technik. Im Vergleich dazu verläuft die Entwicklung von Computerprogrammen sehr langsam. Ein Beispiel: Das weitverbreitete Datenbanksystem IMS/VS von IBM kam 1968 (in einer Vorversion) auf den Markt. Inzwischen hat es mehrere Rechnergenerationen überlebt. Gegenwärtig hat es einen Gesamtumfang von mehr als einer Million Programmzeilen und ist in schätzungsweise etwa zehntausend Installationen in Benutzung.

Es ist interessant, diese Zahlen mit denen für Bücher zu vergleichen: Goethes Gesamtwerk in der Artemis-Ausgabe umfaßt etwa 30 MByte an Information. Diese Zahl liegt in der Größenordnung des Umfangs des IMS/VS Systems. Die Auflage eines Buches ist im allgemeinen wesentlich höher als die eines Programmprodukts. Änderungen, Ergänzungen und Korrekturen werden beim Buch in sehr großen Zeitabständen ausgeführt (bei Neuauflagen). Im Gegensatz zum Buch „lebt“ das Programm, d.h. es ändert sich ständig und ist u.U. in jeder Installation anders implementiert.

Noch einige Worte zum Erstellungsaufwand von Computerprogrammen: Nach Kitchenham und Taylor [6] erstellt ein Programmierer ganz grob im Monat etwa 500 Programmzeilen. Nach einer Firmeninformation [10] kostet ein Programmbefehl mehr als 20 DM. Weiterhin (vgl. etwa Schulz [9]) hat man durchschnittlich auf 100 Befehle bei der Auslieferung eines Programms fünf Fehler. Von diesen sind 2/3 in der Entwurfsphase (Pflichtenheft) entstanden. Die bei der Programmproduktion entstehenden Kosten sind etwa zu 30 % Entwicklungskosten und zu 70 % Wartungskosten. Es sei angemerkt, daß im Einzelfalle stark abweichende Zahlenwerte angegeben werden.

3 Die urheberrechtliche Beurteilung von Computerprogrammen

In der Regel kann man davon ausgehen, daß bei der Hinzuziehung eines Gutachters die folgende Situation vorliegt:

Der Verletzungsfall ist konkret gegeben. Hieraus folgt, daß das betreffende Programm bereits als stabiles Produkt auf dem Markt ist. Andernfalls nämlich würde sich eine Urheberrechtsverletzung nicht lohnen. Bei einem Programm, welches sich noch in der Entwicklung befindet, steht der größte Teil des Gesamtaufwandes noch bevor. Selbst wenn das Produkt in einer frühen Phase ohne Einwilligung des Urhebers verwendet würde, wäre es wegen des erforderlichen hohen Aufwandes für Fehlerkorrektur, Pflege und Wartung nur sehr schwer, anschließend bei Kopie und Original noch Gemeinsamkeiten zu finden. Unabhängig davon ist es häufig einfacher, ein Produkt selbst neu zu entwickeln, als aus möglicherweise noch nicht ausgereiften frühen Entwurfsphasen ein Programm nachzubauen.

Bei einem Programm, das umfangreich genug ist, um Streitobjekt zu sein, muß man mit einem Entwicklungsaufwand von Jahren rechnen und mit einer Einführungszeit in der gleichen Größenordnung. Der Gutachter wird somit hinzugezogen, wenn der Entstehungszeitpunkt des fraglichen Produktes bereits längere Zeit zurückliegt (häufig mehr als zehn Jahre). Man muß sich auch vor Augen halten, daß dann das Produkt, welches zur Begutachtung vorliegt, nicht mehr das Produkt ist, welches ursprünglich entwickelt wurde. Für die Nutzung des Programms sind die Vorläuferphasen nicht notwendig. Da sich das Programm ständig weiterentwickelt, müßten im Prinzip auch die Vorläuferprodukte entsprechend aktualisiert werden. Dies geschieht in der Regel nicht. Die Folge ist, daß zum Verletzungszeitpunkt der Programmcode und das Begleitmaterial vorliegen, jedoch die Vorläuferphasen nicht oder doch wenigstens nicht in brauchbarer Form. Sie sind dann allenfalls noch für die Rekonstruktion des ursprünglichen Produkts interessant.

Der Gutachter wird sich zunächst über die verwendete Programmiersprache informieren, über die zugrundeliegende Anwendung, möglichst auch über den Rechner, auf dem das Programm entwickelt wurde bzw. auf dem es eingesetzt wird. Konkret heißt das, daß man sich durch das Studium von Fachliteratur nach Möglichkeit auf den Kenntnisstand des Programmentwicklers zum Zeitpunkt der Programmerstellung zu bringen versucht.

Danach beginnt die Sichtung des Programmcodes, soweit dieser verfügbar ist. Verständlicherweise sind Hersteller zuweilen zurückhaltend, was die Herausgabe des Quellcodes anbelangt. Man wird sich von Fall zu Fall einigen müssen, inwieweit man unter Berücksichtigung der Interessen Beteiligter und der Gegebenheiten des Falles einen gangbaren Weg zur Ermittlung der Urheberrechtsfähigkeit finden kann. Bislang lag mir allerdings der Quellcode stets vor.

Die Sichtung des Programmtextes durch den Gutachter ist – insbesondere wenn der Code auf eine Weise geschrieben wurde, die dem menschlichen Leser nicht entgegenkommt, oder wenn erhellende Vorläuferphasen nicht mehr vorhanden sind – eine sehr mühsame Aufgabe. Der Gedanke läge nahe, diese Arbeit zu automatisieren, das heißt durch ein Computerprogramm Maßzahlen – etwa Komplexitätsmaße – aus dem zu untersuchenden Programm zu extrahieren und sich dann bei der Beurteilung der Schutzfähigkeit allein auf diese zu konzentrieren. Ich halte ein solches Vorgehen nicht für zweckmäßig, obwohl es natürlich zu aufschlußreichen Informationen führen kann. Meines Erachtens wollte der Bundesgerichtshof im „Inkassoprogramm“-Urteil [11] genau dies vermeiden, als er ausdrücklich forderte:

> ‚Für die Frage der schöpferischen Gestaltungshöhe kommt es grundsätzlich nicht auf den quantitativen Umfang des Programms an; ebensowenig darauf, mit welchem Aufwand und welchen Kosten es konzipiert worden ist . . . ‘

Diese Forderung hat ihren guten Sinn. Würden nämlich algorithmisch verifizierbare Kriterien in irgendeiner Form verbindlich für die Werkhöhe, dann würden sich die Hersteller naturgemäß auf diese Kriterien einstellen zum Schaden des Produkts. Man kann sich unschwer vorstellen, welche

Folgen es beispielsweise hätte, wenn eine gewisse Minimalzahl von Programmzeilen Kriterium für die Schutzfähigkeit wäre.

Auf jeden Fall sollte man sich aber bei der Sichtung des Programms die Mühe machen, einige charakteristische Merkmale, die sich in Zahlen fassen lassen, herauszudestillieren. Hierzu gehört ganz sicher der Programmumfang (d.h. die Anzahl der Zeilen), die Anzahl der Unterprogramme, die gegenseitige Aufrufverschachtelung der Unterprogramme usw. Hierdurch gewinnt man einen Eindruck von der „Komplexität“ des vorgelegten Produkts. Wenigstens für einige typische Beispielfälle sollte man die Datenstrukturen ermitteln. Dies ist ohne Begleitmaterial recht schwierig, da man aus dem Programm allein häufig nur indirekt auf die Strukturierung der Daten schließen kann. Generell kann man die Datenstrukturen und sogar die Vorläuferphasen aus dem Code rekonstruieren („reverse engineering“), es ist dies aber nicht immer einfach.

Bei der Sichtung wird man zunächst jedes einzelne Konstruktions- und Stilelement des Programms (Verwendung von Namen, Umgang mit dem Befehlsvorrat der Sprache, Strukturierung der Daten und des Programms usw.; vgl. das „Baustatikprogramm“-Urteil [12], [13], wo für den Ausführungsteil eines Programms neun solche Stilmerkmale angeführt werden) nach den folgenden Kriterien prüfen:

- War das betreffende Element bei der Entstehung des Programms verbreitet?
- Ergibt sich die verwendete Konstruktion zwingend oder doch naheliegend aus urheberrechtlich nicht relevanten Ursachen, z.B. der Logik der verwendeten Sprache oder Traditionen, Vereinbarungen und Notwendigkeiten des Anwendungsgebiets?
- Welche Teile des Programms wurden mechanisch durch einen Editor oder Programmgenerator erstellt?
- Welche Programmteile wurden aus bereits existierenden Programmen oder anderen Quellen übernommen?

Der verbleibende Rest des Programms (einschließlich der Kommentare) kann als der eigenständige Anteil des Programmierers betrachtet werden. Die verbleibenden Strukturelemente sind also individuell und originell, aber damit noch nicht notwendig schon schöpferisch. Dieser Punkt wird häufig mißverstanden, ich nenne einige Beispiele aus der Prozeßpraxis:

Orthographische Fehler in den Kommentaren sind sicherlich individuelle Merkmale, ohne deshalb schöpferische Qualität zu besitzen.

Variable, Unterprogramme, Datensätze usw. müssen benannt werden. Hierbei kann man phantasielos vorgehen (die erste auftretende Variable heißt A, die zweite B usw.) oder systematisch (alle Unterprogramme, die mit Dateneingabe zu tun haben, beginnen mit dem Buchstaben I wie `INPUT`, ...) oder ausgesprochen phantasievoll (ein Unterprogramm heißt `DONALD DUCK`, ...). Auf jeden Fall dürfte es sich dabei nur in den allerseltensten Fällen um nennenswerte schöpferische Leistungen handeln.

Auf ähnlicher Stufe stehen auffällige Abweichungen vom „guten Ton“ des Programmierens: Maschinenabhängige Konstruktionen, Ausnutzung von Nebeneffekten von nicht präzise definierten Sprachelementen, unsaubere Namensgebung, Nutzung experimentell erworbener Kenntnisse, die nicht theoretisch fundiert sind usw. Diese Unsauberkeiten mindern die Qualität des Produktes und sollten keinesfalls bei der Beurteilung der Urheberrechtsfähigkeit eine Rolle spielen, auch wenn sie noch so „trickreich“ und raffiniert sind.

Diese individuellen und originellen Konstrukte sind – dies sei hier nur angemerkt – jedoch äußerst nützlich, um Plagiate festzustellen.

Nach Elimination des in diesem Sinne Trivialen verbleiben diejenigen Teile des Programms und der Nebenprodukte, die für eine urheberrechtliche Einschätzung bedeutsam sein können. Hierzu gehören:

Die Auswahl der für die Lösung des Problems anwendbaren Methoden und Algorithmen. Hierbei gibt es Abgrenzungsprobleme nach dem Urheberrecht. Im „Inkassoprogramm"-Urteil [11] heißt es:

> ‚Dem Urheberrechtsschutz ist daher die in dem Computerprogramm berücksichtigte, sich auf einen vorgegebenen Rechner beziehende Rechenregel (der sogenannte Algorithmus) ebensowenig zugänglich wie andere bei der Erstellung des Programms herangezogene mathematische oder technische Lehren oder Regeln, die als Bestandteil der wissenschaftlichen Lehre frei sind und jedermann zugänglich sein müssen.'

Die Auswahl der geeigneten Algorithmen allein ist nicht unbedingt eine schöpferische Leistung, es muß deutlich werden, daß diese Auswahl nicht zufällig erfolgte, sondern als Ergebnis eines umfassenden Verständnisses des zu lösenden Problems und der dazu erforderlichen technischen Möglichkeiten entstanden ist. Für die Lösung eines linearen Gleichungssystems, welche eine Grundaufgabe der angewandten Mathematik darstellt, gibt es eine große Anzahl von Algorithmen. Wählt ein Programmierer unter diesen Möglichkeiten diejenige aus, die am verbreitetsten ist, dann kann dies durchaus Ergebnis einer eingehenden Auseinandersetzung mit dem Problem sein, jedoch wird man ohne zusätzliches Material vermuten, daß die Auswahl aufs Geratewohl erfolgte.

Die Unterteilung des Gesamtproblems in Teilprobleme und deren Realisierung durch Unterprogramme dient der Übersichtlichkeit und der Komplexitätsreduktion und ist daher ein wichtiges Werkzeug bei der Produktion eines Programms. Gleichzeitig ergeben sich hierbei zahlreiche schöpferische Gestaltungsmöglichkeiten für den Programmierer. Neuere Untersuchungen (Jankowitz [5]) zeigen, daß diese Art der Strukturierung des Problems typisch ist für den Programmierer, so daß die von ihm benutzte logische Struktur gleichsam ein „Fingerabdruck" ist, der sein Produkt von anderen unterscheidet. In zahlreichen Fällen ist auch die Strukturierung der verwendeten Daten ein schöpferisches Merkmal, jedoch nur dann, wenn die verwendeten Strukturen nicht trivial sind und wenn sie nicht bereits in dem betreffenden Anwendungsgebiet allgemein bekannt sind. Ein Beispiel: Ein Bibliotheksverwaltungsprogramm, bei dem die Daten nach dem Bilde der „klassischen" Kartei strukturiert sind mit einer „Verfasserkarte" und „Sachgebietskarten" usw. ist keine schöpferische Leistung. Der erste Programmierer, der für diesen Zweck moderne Datenstrukturen einführte, vollbrachte sicherlich eine schöpferische Leistung (vgl. etwa Meadow [7]). Heute gehört die Verwendung einer sachgemäßen Datenstruktur für diesen Zweck in den Bereich des rein Handwerklichen. Ein leider nur wenig genutztes Betätigungsfeld für schöpferische Leistungen bieten die Kommentare im Programm und die Gestaltung des Begleitmaterials. Hierbei ist eine methodische und didaktische Leistung zu vollbringen: Der Programmierer teilt dem Benutzer bzw. dem Bearbeiter des Programms mit, worauf beim Gebrauch bzw. bei der Fehlerkorrektur und Programmpflege zu achten ist. Hierbei kann und sollte man sich aller sinnvollen Möglichkeiten und aller Erkenntnisse auf den Gebieten der Pädagogik und der Benutzerpsychologie bedienen. Daß dies unglücklicherweise nicht geschieht und daß dieser Aspekt von den Programmgestaltern fast völlig vernachlässigt wird, lehrt ein Blick in eine beliebige Programmbeschreibung oder in ein Benutzerhandbuch. Die Kommentare innerhalb eines Programms sind häufig äußerst lakonisch und wenig hilfreich, nicht selten werden die zu kommentierenden Befehle nur leicht abgewandelt

wiederholt. Dies mag ein Erbe aus der Zeit sein, als Kommentare aus Speicherplatzgründen nur sparsam verwendet werden konnten.

Ähnliches gilt für die Benutzeroberfläche, das heißt für die Eingabemasken und Ausgabeformate sowie die System- und Fehlermitteilungen und Hilfsfunktionen. Während bei den ersteren noch Aufwand betrieben wird – kein Kunde kauft ein Produkt, dessen Ausgaben unverständlich sind und dessen Eingabemasken die Fehlerhäufigkeit erhöhen – sind die Mitteilungen an den Benutzer häufig grob irreführend. Carroll [2] gibt Beispiele dafür an.

Eine interessante Frage ist es, welche Rolle die Fehlerfreiheit und Funktionstüchtigkeit eines Programms bei der Feststellung der Urheberrechtsfähigkeit spielt. Persönlich neige ich zu der Ansicht, daß eine Ausführung sehr wohl schöpferisch sein kann ohne daß das resultierende Programm korrekt ist. Hierbei sollte man berücksichtigen, daß es kaum ein Programm ohne Fehler gibt. In der „Baustatikprogramm"-Entscheidung [12, 13] wurde die Tatsache, daß die Programme offenbare Schwächen und Fehler zeigten, nicht als Argument gegen die Schutzfähigkeit nach dem Urheberrecht verwendet.

4 Der Durchschnittsprogrammierer

Nach dem „Inkassoprogramm"-Urteil [11] des Bundesgerichtshofs hat man die als individuell und nichttrivial erkannten Merkmale des Programms und des Begleitmaterials sowie der Vorläuferphasen mit dem Schaffen eines Durchschnittsprogrammierers zu vergleichen. Durch die Hilfskonstruktion des Durchschnittsprogrammierers, die dem Durchschnittsfachmann im Patentrecht entspricht, erreicht der Bundesgerichtshof, daß der Begutachtungsprozeß sich der laufenden Entwicklung anpaßt. Es ist daher notwendig, sich mit diesem Begriff auseinanderzusetzen.

Ganz sicher ist in diesem Zusammenhang unter dem „Programmierer" nicht der „Codierer", also der Erzeuger des eigentlichen Programmcodes zu verstehen. In zunehmendem Maße entwickeln sich die Methoden und Werkzeuge dahin, daß Programme und ihre Vorstufen mehr und mehr automatisch erzeugt werden. Das Gewicht wird sich also in Richtung Datenfluß- und Ablaufdiagramme bzw. Pflichtenheft verschieben. Gleichzeitig muß berücksichtigt werden, daß natürlich auch Rückwirkungen vom Code auf die Datenfluß- und Ablaufpläne und von diesen auf das Pflichtenheft existieren. Ich möchte hier als „Programmierer" denjenigen (bzw. diejenigen) verstehen, der den für das Gesamtwerk eigentümlichen Beitrag leistet.

Formal erhält man den Durchschnittsprogrammierer, indem man die Fähigkeiten und Kenntnisse aller Programmierer einer gewissen Grundgesamtheit unter diesen „demokratisch" aufteilt. Hierbei stellt sich zunächst die Frage, was man unter einem Programmierer verstehen soll. Sicher ist es im Kontext des Urheberrechts nicht zulässig, die Definition des Programmierers „ständisch" zu fassen: Ein Programmierer ist eine Person, die nach Tarif als Programmierer bezahlt wird oder Mitglied der „Gesellschaft für Informatik" ist etc. Eine solche Einengung würde den begabten Amateur, der auf diesem Gebiet häufig anzutreffen ist, diskriminieren.

Andererseits führt eine zu liberale Definition des Begriffs „Programmierer" auf ein Anspruchsniveau, das sicher nicht dem Sinn des Inkassoprogramm-Urteils entspricht. Sind zum Vergleich auch die Fähigkeiten des Schülers, der einen BASIC-Kurs belegt hat, zugelassen, dann wird damit der Durchschnitt unzulässig niedrig angesetzt. Da auch auf diesem Gebiet die das Wissen in Form einer Pyramide verteilt ist – es gibt sehr wenige Spitzenprogrammierer und eine große Menge von wenig qualifizierten Programmierern – führt eine Nivellierung zwangsläufig auf ein zu niedriges Durchschnittsniveau.

Es gibt wohl kaum ein anderes Fachgebiet, in dem die Qualifikationen so breit gestreut und so schwer verbindlich definierbar sind, wie in der Datenverarbeitung. Dies liegt einmal an der Datenverarbeitung selbst. Bedingt durch die äußerst rasche Entwicklung entstehen ständig neue Anforderungen, die geänderte Qualifikationen notwendig machen. Beispiele solcher neuentstandenen Anforderungen sind: Verteilte Systeme und Netzwerke, Vektor- und Parallelrechner, „künstliche Intelligenz“. Es entstehen mithin also ständig neue, zunächst diffus abgegrenzte, später dann möglicherweise etablierte „Durchschnittsprogrammierer“.

Die Datenverarbeitung ist eine typische Hilfswissenschaft – jedenfalls soweit von urheberrechtsfähigen Produkten die Rede ist. Damit ist jede Betätigung auf diesem Gebiet nur in Symbiose mit einer Anwendung denkbar und sinnvoll – sieht man einmal von Systemprogrammen ab, bei denen der Anwendungsbezug nicht unmittelbar sichtbar wird. Die Zahl der möglichen Anwendungen ist nicht beschränkt, jedes einzelne Anwendungsgebiet fordert vom Programmierer Zusatzqualifikationen. Die Fähigkeiten eines typischen Programmierers für kommerzielle Anwendungen und die Fähigkeiten eines typischen Programmierers für wissenschaftlich-technische Anwendungen sind nicht miteinander vergleichbar – ich bin mit beiden Bereichen intim vertraut. Analoges dürfte für andere Anwendungsgebiete gelten. Diese große Vielfalt der möglichen Spezialisierungen und die rasche Entwicklung auf diesem Gebiet, die eine ständige Neudefinition der Begriffe erfordert, machen es technisch sehr schwer, den Durchschnittsprogrammierer in einem vorgegebenen Sachzusammenhang und zu einem gegebenen Zeitpunkt zweifelsfrei zu definieren. Man muß davon ausgehen, daß zwei verschiedene Gutachter mit verschiedenem Erfahrungshintergrund auch von verschiedenen Durchschnittsprogrammierern ausgehen und damit im Einzelfalle rational einwandfrei zu konträren Urteilen über ein vorgelegtes Programm gelangen müssen.

Diese Situation wäre tolerierbar, wenn voneinander abweichende Einschätzungen der Werkhöhe eines Produkts aufgrund abweichender Bilder des Durchschnittsprogrammierers verhältnismäßig selten wären. Es ist aber zu befürchten, daß dies nicht so ist.

Ein Vergleich der Datenverarbeitung mit anderen Fachgebieten macht einen weiteren fundamentalen Unterschied deutlich. Ein Durchschnittsfachmann auf einem „klassischen“ Wissensgebiet (beispielsweise der Medizin, den Ingenieurwissenschaften oder der Chemie) läßt sich unzweideutig definieren, etwa durch Angabe seiner Ausbildung bzw. der absolvierten Prüfungen oder durch Zugehörigkeit zu einer berufstypischen Organisation. Völlig anders ist die Situation im Falle des Programmierers. Man sollte sich vergegenwärtigen, daß es erst seit Anfang der siebziger Jahre wissenschaftliche Informatikinstitute in Deutschland gibt. Auch heute noch besteht der überwiegende Teil aller Programmierer aus Amateuren. Im Gegensatz zu anderen Fachgebieten leisten auf dem Gebiet der Programmierung von Computern gerade Amateure einen wesentlichen Beitrag. Spektakuläre Beispiele hierfür sind aus der Tagespresse bekannt. Man mag einwenden, daß sich die Verhältnisse in dem Maße stabilisieren werden, in dem professionelle Programmierer mit nachvollziehbaren Qualifikationen auf den Markt kommen. Ich möchte dies bezweifeln. Es gehört zum Wesen des Programmierens, daß immer wieder in noch unerschlossenen Anwendungsgebieten die Anwender selbst Programme erstellen. Diese Erscheinung garantiert eine beständige dynamische Weiterentwicklung. Unterstützt wird diese Dynamik durch eine umfassende Popularisierung des Programmierens, begleitet durch die Weiterentwicklung von Programmierwerkzeugen, Programmierumgebungen und „höheren“ Programmiersprachen, die ja letzlich dem Ziel dienen, auch dem Unkundigen das Programmieren zu ermöglichen. Man kann schließlich abschätzen, daß die weitere Entwicklung der Rechner auch in der nächsten Zukunft so stürmisch verlaufen wird wie bisher. Neue technische Möglichkeiten schaffen in rascher Folge völlig neue Situationen für den Programmierer, die bewirken, daß etablierte Abgrenzungen wieder fließend werden.

Natürlich wird es in vielen Fällen zweifelsfrei möglich sein, das Können des Durchschnittsprogrammierers hinlänglich abzugrenzen. Man sollte sich aber vor Augen halten, daß diese Figur in

jedem Einzelfalle wieder neu geschaffen werden muß in Abhängigkeit von der Entstehungszeit des untersuchten Programms und auch in Abhängigkeit von den konkret vorliegenden Umständen.

5 Schlußfolgerungen

Aus dem Gesagten ergibt sich, daß es im Einzelfalle nicht möglich ist, verbindlich auszusagen, ob ein vorgelegtes Produkt urheberrechtsfähig ist oder nicht. Der Spielraum der Einschätzung der einzelnen Programmkonstruktionen und der Nebenprodukte ist beträchtlich und hängt unter anderem von der Erfahrungsbreite des Gutachters ab. Die Konstruktion des Durchschnittsprogrammierers verhindert zwar das unzulässige „Einfrieren“ der Kriterien, andererseits ist sie zu vage, um reproduzierbare Schlußfolgerungen zuzulassen. Diese Situation ist für alle Beteiligten äußerst unbefriedigend.

Der Produzent von Computerprogrammen hat kein verläßliches Kriterium in der Hand, nach dem er sein Produkt urheberrechtlich einschätzen kann. Üblicherweise wird die Schutzfähigkeit erst im Verletzungsfalle festgestellt werden. Das heißt aber doch, daß der Ersteller des Programms zunächst einmal erheblich in den Produktionsprozeß investiert hat, danach sein Produkt mit allen damit verbundenen unternehmerischen Risiken auf den Markt gebracht hat und schließlich, nachdem das Produkt erfolgreich eingeführt ist, riskiert, daß ein anderer die Früchte dieser Mühen straflos erntet. Man kann durchaus sagen, daß derzeit das Risiko des Verletzers geringer ist als das des Produzenten.

Es kommt hinzu, daß Computerprogramme häufig nicht isoliert auftreten, sondern innerhalb einer Produktfamilie oder im Zusammenhang mit Geräten oder Anlagen, welche beispielsweise durch die Programme gesteuert werden. Der Verletzer kann sich hier die „Rosinen“ heraussuchen und dem seriösen Hersteller die weniger attraktiven Teile überlassen. Unter Umständen bewirken die Produkte des Verletzers eine Destabilisierung des gesamten Systems. Die Möglichkeiten des Herstellers, sich dagegen zur Wehr zu setzen, sind begrenzt, da er auf seine Kunden Rücksicht nehmen muß. Ich möchte hier nicht einer Monopolisierung von Produktfamilien das Wort reden, ich spreche über schamlose und offenkundige Übernahme von Teilprogrammen und Konzepten, wobei die wettbewerbsrechtlichen Einschränkungen durch geeignete Überarbeitung der Programme gerade eben umgangen werden.

Der Käufer bzw. Benutzer von Computerprogrammen wird durch ungeklärte – und zum Kaufzeitpunkt nicht klärbare – urheberrechtliche Verhältnisse unter Umständen empfindlich getroffen. Dies ist besonders dann schwer erträglich, wenn alle Beteiligten offenbar seriös gehandelt haben. Selbst ein fachkundiger und gut informierter Käufer von Computerprogrammen muß mit dem Risiko leben, daß das von ihm gekaufte Produkt durch in der Tat nicht vorhersehbare Entwicklungen urheberrechtlich bedenklich wird. Zwar wird man dieses Risiko niemals völlig eliminieren können, jedoch glaube ich, daß es hier und heute entschieden zu groß ist. Als zwar indirekt aber trotzdem empfindlich Betroffener einer solchen Auseinandersetzung, die gerade jetzt außergerichtlich entschieden wurde [14], bin ich bezüglich dieses Punktes möglicherweise übermäßig sensibilisiert.

Ich habe jetzt einige Dinge aufgezählt, die ich für unbefriedigend halte. Man kann diese Liste verlängern. Schwieriger ist es, Abhilfe zu schaffen.

Eine Vermutung der Urheberrechtsfähigkeit bei hinreichend komplexen Programmen wäre hilfreich (vgl. Moritz, Tybusseck [8], S. 26). Das Problem ist hier die Definition der Komplexität. Die Anzahl der Programmzeilen ist sicher ein ungeeignetes Maß. Natürlich könnte man ein Komplexitätsmaß als Grenze für die Vermutung der Urheberrechtsfähigkeit nicht ein für alle Mal

festschreiben, durch zunehmend „intelligentere“ Programmierwerkzeuge wird jede algorithmisch formulierte Komplexitätsschranke früher oder später automatisch, das heißt, buchstäblich „auf Knopfdruck“ zu erreichen sein.

Ich hatte eingangs erwähnt, daß die Plagiatsfeststellung sehr viel leichter algorithmisch ausgeführt werden kann, als die Feststellung der Schutzfähigkeit. Zunehmend werden in Gerichtsverhandlungen diese beiden Teile des Begutachtungsprozesses getrennt: Zunächst wird die Schutzfähigkeit festgestellt. Ist diese gegeben, dann wird in einem zweiten Durchgang über die Verletzung befunden. Für eine solche Vorgehensweise sprechen scheinbar logische Argumente. Ich glaube jedoch, daß es sinnvoll wäre, diese Reihenfolge umzukehren. Man muß davon ausgehen, daß sich Plagiat und Vorlage selten allzusehr unterscheiden. Ein Plagiator, der die Vorlage entscheidend verändert und bei dieser Veränderung möglicherweise noch eigene schöpferische Leistungen einfließen läßt, ist eben kein Plagiator mehr (man denke etwa an Brecht).

Der Sinn des Plagiats ist es, Arbeitsaufwand zu sparen, sei es nun schöpferische oder nicht schöpferische Arbeit. Der Aufwand der erforderlich wäre, ein widerrechtlich entwendetes Progamm soweit zu modifizieren, daß die ursprüngliche Struktur nicht mehr erkennbar ist, liegt bei einem einigermaßen komplexen Programm in der Größenordnung des Erstellungsaufwandes.

Bei der Plagiatsfeststellung vergleicht man ein Produkt nicht mit den Leistungen eines fiktiven „Durchschnittsprogrammierers“, sondern man vergleicht zwei konkrete Programme miteinander. Hier spielt tatsächlich Individualität und Originalität ohne jede Wertung eine Rolle. Man kann uneingeschränkt behaupten, daß zwei Programme, die unabhängig voneinander entstanden sind, sich notwendigerweise signifikant voneinander unterscheiden. In meiner Eigenschaft als Hochschullehrer gehört es zu meinen Aufgaben, Studenten mathematische Problemstellungen durch Erstellung relativ kleiner Programme lösen zu lassen. Wenn eine Aufgabenstellung überhaupt präzise definierbar ist, dann sicher eine mathematische. Zudem sind bei derartigen Aufgabenstellungen die verwendeten Datenstrukturen in aller Regel primitiv, Ein- und Ausgabeteile rudimentär. Es verbleibt somit allein die programmtechnische Gestaltung des Algorithmus. Nichtsdestoweniger sind die Unterschiede der einzelnen „Autoren“ stets hochsignifikant.

Alle diese Überlegungen bestätigen, daß die Plagiatsfeststellung einfach sein muß. Man kann umgekehrt sagen: Ist die Feststellung des Plagiats nicht einwandfrei möglich, dann erübrigt sich die Frage nach der Schutzfähigkeit.

Hat man einmal festgestellt, daß zwei verschiedene Programme nicht unabhängig voneinander entstanden sind, dann stützt diese Tatsache die Hypothese der Urheberrechtsschutzfähigkeit. Selbstverständlich hat man damit allein noch keinen schlüssigen Beweis, allerdings ein Indiz, das man mit gebührender Vorsicht verwenden kann. Das Risiko des Plagiats und auch das Risiko eines Prozesses um ein Programmprodukt lohnt sich tatsächlich nur dann, wenn das Programm aufwendig ist. Dieser Aufwand kann in der schöpferischen Leistung stecken oder aber in reiner „Sklavenarbeit“ ohne höheren Anspruch. Zur Feststellung der Schutzfähigkeit muß dann nur noch entschieden werden, ob die in dem Programm enthaltene erhebliche Leistung den Ansprüchen des Urheberrechts genügt.

In der Begründung der „Baustatikprogramm“-Entscheidung weist das OLG Frankfurt einen pragmatischen Weg zur Feststellung der Schutzfähigkeit auf, der den expliziten Bezug auf den Durchschnittsprogrammierer weitgehend vermeidet ([13], S. 18):

> ‚Der Senat ... bejaht daher die urheberrechtliche Schutzfähigkeit eines Datenverarbeitungsprogrammes, wenn
>
> a) es sich nicht nur um ein einfaches Programm handelt, wenn sich also bei dem

Programm der Progammablauf und die Befehlsstruktur nicht notwendig aus der Aufgabenstellung ergeben,

b) der Urheber bei der Gestaltung frei zwischen verschiedenen Formeln und Abläufen wählen und die Variablen frei festlegen konnte,

c) das Programm sich nicht in der mechanisch-technischen Fortführung des allgemein Vorbekannten erschöpft,

d) und somit ein bedeutendes schöpferisches auf einen Urheber zurückzuführendes Übertragen der Gestaltungsfähigkeit in Auswahl, Sammlung, Sichtung, Anordnung und Einteilung der Informationen und Anweisungen gegenüber dem allgemeinen Durchschnittskönnen erkennen läßt.‘

Man hat in der Tat an diese Entscheidung die Hoffnung geknüpft, daß damit ein zuverlässiges und klares Kriterium für die Urheberrechtsfähigkeit gefunden sei. Diese Hoffnung wurde vom Herausgeber der Zeitschrift „GRUR“ auch in einer Anmerkung zum „Baustatikprogramm“-Urteil artikuliert [12]. Inwieweit der hierdurch aufgezeigte Weg auch tatsächlich begangen wird, entzieht sich meiner direkten Kenntnis. Sporadische Informationen und Einzelbeobachtungen scheinen darauf hinzudeuten, daß der Schutz von Computerprogrammen nach wie vor mit den hier genannten Risiken behaftet ist und daß sich daher die skeptische Sicht, die Bauer [1] unmittelbar nach der „Inkassoprogramm“-Entscheidung publizierte, voll berechtigt war.

Ich halte es für dringend erforderlich, daß auf diesem Markt, der für die wirtschaftliche Zukunft unseres Landes von höchster Bedeutung ist, möglichst bald klare und überschaubare Verhältnisse herrschen. Wir können es uns ganz sicher nicht leisten, daß ein Industriezweig, der überdurchschnittliches Wachstum und immense Impulse für andere Wirtschaftszweige verspricht, ohne die für die industrielle Produktion typischen Umweltbelastungen aufzuweisen, durch eine unklare Rechtssituation in seiner Entwicklung behindert wird.

Literatur

[1] Bauer, K. A. *Rechtsschutz von Computerprogrammen in der Bundesrepublik Deutschland – eine Bestandsaufnahme nach dem Urteil des Bundesgerichtshofs vom 9. Mai 1985. Computer und Recht* 1/85:5–12, 1985.

[2] Carroll, J. M. *The adventure of getting to know a computer. IEEE Computer* November 1982: 49–58.

[3] Conte, S. D., Dunsmore, H. E. and Shen, V. Y. *Software Engineering Metrics and Models.* Benjamin/Cummings, Menlo Park, Cal., 1986.

[4] Hausen, H. L., Müllerburg, M. und Schmidt, M. *Über das Prüfen, Messen und Bewerten von Software. Methoden und Techniken der analytischen Software-Qualitätssicherung. Informatik-Spektrum* 10:123–144, 1987.

[5] Jankowitz, H. T. *Detecting plagiarism in student Pascal programs. The Computer Journal* 31:1–8, 1988.

[6] Kitchenham, B. A. and Taylor, N. R. *Software project development cost estimation. The Journal of Systems and Software* 5:267–278, 1985.

[7] Meadow, C. T. *The Analysis of Information Systems. A Programmer's Introduction to Information Retrieval.* John Wiley & Sons, Inc., New York, London, Sydney, 1967.

[8] Moritz, H. W. und Tybusseck, B. *Computersoftware, Rechtsschutz und Vertragsgestaltung. Eine systematische Darstellung nach deutschem und EG-Recht.* Verlag C. H. Beck, München, 1986.

[9] Schulz, A. *Programmentwurf*. In H. R. Hansen, editor, *Entwicklungstendenzen der Systemanalyse. Fachberichte und Referate, Band 10*, S. 371–400, R. Oldenbourg Verlag, München, Wien, 1978

[10] Sperry Univac *Universelles Bildschirm – Anwendungssystem MAPPER.* D 575 (Rev. 1). 10 81 05 03.

[11] Rechtssprechung zum Wirtschaftsrecht. *BGH: Urheberrechtsschutz für Computerprogramme. UrhG §2 Abs. 1 Nr. 1, 7; §2 Abs. 2. Inkasso – Programm. Computer und Recht* 1/85:22–32, 1985.

[12] *UrhG §§2 Abs. 1, Nr. 1,7, Abs. 2, 97 Abs. 1. – „Baustatikprogramm". Zur Überprüfung und Ermittlung der Urheberrechtsfähigkeit eines Computerprogramms. OLG Frankfurt, Urt. v. 6.11.1984 – 14 U 188/81. GRUR* 1985:1049–1052.

[13] *OLG Frankfurt: Baustatikprogramme. Computer und Recht* 1/86:13–22, 1986.

[14] *IBM bittet zur Kasse. Markt & Technik* Nr. 49 vom 9. Dezember 1988, S. 1 und 5, 1988