Kreativität in der Informatik: Anwendungsbeispiele der innovativen Prinzipien aus TRIZ

(Langbeitrag)

Janine Willms, Ina Wentzlaff, Markus Specker

Abt. Lehr-/Lernsysteme, Fachbereich Informatik, Universität Oldenburg
Janine.Willms@Informatik.Uni-Oldenburg.de

1 Einleitung

Dieser Beitrag entstand im Rahmen eines Fortgeschrittenenpraktikums, das die Untersuchung und Nutzung kreativer Aspekte in der Informatik zum Thema hat. Es ist eingelagert in eine laufende Doktorarbeit zur Konzeption und prototypischen Entwicklung eines Entscheidungsunterstützungssystems für die Patentanmeldung und -prüfung mit besonderem bezug zur Informatik (Willms, 1999) (Willms, Möbus, 2000). Patente werden für innovative Entwicklungen erteilt, die aus der Kreativität des Entwicklers hervorgehen. Es soll untersucht werden, mit welchen Methoden eine Patentanmeldung, die nicht erfinderisch genug ist, um ein Patent zu erhalten, kreativ verändert werden kann. In diesem Beitrag wird Fragen nachgegangen, was Kreativität ist, welche Rolle die Kreativität für das Patentwesen spielt und ob man bestehende Prinzipen des kreativen Problemlösens aus der Theorie des erfinderischen Problemlösens TRIZ auch in der Informatik wiederfindet bzw. diese anwenden kann, um neue Entwicklungen voranzutreiben. Ein Hauptziel dieses Beitrags liegt somit darin, die in TRIZ definierten innovativen Prinzipien anhand von Beispielen aus dem Informatikumfeld vorzustellen.

2 Was ist Kreativität?

Was ist eigentlich Kreativität? Ob in der Kunst, Musik, Architektur, Schriftstellerei, Technologie, wissenschaftlich Entdeckungen, aber auch im Problemlösen und Handeln – der Mensch ist fähig, kreativ zu sein, und Kreativität gilt als eine positive Eigenschaft. Eine allgemein anerkannte Definition von Kreativität existiert jedoch bisher nicht. Psychologen versuchen durch Interviews mit Personen, die von anderen als kreativ eingestuft wurden und bereits diverse Preise gewonnen hatten, deren besondere Eigenschaften zu identifizieren, darunter Naturwissenschaftler, Künstler, Schriftsteller und Musiker (Czikszentmihalyi, 1997). Andere Forscher analysieren die historische Entwicklung herausragender Personen, wie zum Beispiel Leonardo da Vinci, Albert Einstein oder Picasso, anhand ihrer Bibliographie (Simonton, 1990) oder von ganzen Fachgebieten auf Basis der fortschreitenden Technologien

(Dasgupta, 1996). Mit Hilfe von Computersimulationen wird versucht, die kognitiven Fähigkeiten kreativer Personen erklärbar zu machen (Langley et al., 1987).
Kreativität wird auch heute noch vielfach als unerklärbarer, teils mystischer Prozeß angesehen. Betrachtet man die Begriffe Inspiration, Eingebung, Geistesblitz oder Erleuchtung, so wird deutlich, daß Kreativität nicht als Eigenschaft des Menschen beurteilt wurde, sondern als Eingriff einer äußeren, höheren Instanz. Aus diesem Alltagsverständnis von Kreativität ergibt sich das Problem der „Definition der immer höher gesteckten Ziele“ (Kurzweil, 1993). Die Analyse und Erklärung kreativer Prozesse sowie deren Simulation durch Computerprogramme führt lediglich dazu, daß die erreichte Leistung nicht mehr als kreativ gilt, da sie nun erklärbar geworden ist und „einfachen Regeln und Mustern“ folgt. Dennoch wird weiter versucht, Kreativitätsmuster zu entdecken und auch im Computer nutzbar zu machen.

3 Kreativität und Patente

Kreativität wird verstanden als:

> „Fähigkeit, möglichst viele verschiedenartige und ungewöhnliche (originelle) Lösungen einer Aufgabe produzieren zu können“ (Wörterbuch der Kognitionswissenschaft, Strube 1996)
>
> „Production of an idea, action, or object that is new and valued“ (MIT Encyclopedia of the Cognitive Sciences, Wilson, Keil, 1999)
>
> „the connecting and rearranging of knowledge - in the minds of people who will allow themselves to think flexibly - to generate new, often surprising ideas that others judge to be useful“ (Creativity, Innovation and Quality, Plsek 1997)

Die Definitionen für Kreativität lassen sich auf den folgenden gemeinsamen Grundtenor zurückführen: *Kreativität wird als Prozeß angesehen, dessen Resultat von der Gesellschaft als neu und ungewöhnlich angesehen wird. Dabei wird auf das bisher vorhandene Vorwissen zurückgegriffen, dieses kombiniert, verändert und erweitert.*
M. Boden (Boden, 1990) definiert psychologische Kreativität (P) als Leistungen, deren Ergebnisse für einen bestimmten Menschen persönlich fundamental neu sind. Das Ergebnis ist außerdem historisch kreativ (H), wenn es bezogen auf die gesamte menschliche Geschichte fundamental neu ist. Dasgupta erweitert diese Definition und trennt die fundamentale Neuheit in Neuheit (N) und Originalität (O) (Dasgupta, 1996). Dadurch ergibt sich eine Vier-Felder-Matrix. Die vier Kreativitätsarten von Dasgupta spiegeln die Phasen des Patentierungsprozesses wider. Ein Produkt des menschlichen Denkens, das PO-kreativ ist, kann als Ausgangspunkt für eine Patentanmeldung dienen. Im Patentamt wird dann geprüft, ob dieses Produkt auch bezogen auf die gesamte Menschheit neu (HN-kreativ) und erfinderisch (HO-kreativ) ist.

	Neuheit (N)	Originalität (O)
Psychologische Kreativität (P)	PN – Kreativität Es wird entwickelt...	PO – Kreativität Das Ergebnis erscheint dem Entwickler als patentwürdig – er meldet ein Patent an...
Historische Kreativität (H)	HN – Kreativität Das Patentamt prüft die Anmeldung auf Neuheit bezogen auf den Stand der Technik...	HO – Kreativität Das Patentamt prüft die Anmeldung auf erfinderische Tätigkeit bezogen auf den Stand der Technik...

Fig. 1. Bezug der Kreativitätsarten nach Dasgupta zum Patentwesen

Der Begriff der Kreativität als solcher wird im Patentwesen nicht genutzt. Patente werden für Erfindungen erteilt, die sich laut §4 PatG durch Neuheit, erfinderische Höhe und gewerbliche Anwendbarkeit auszeichnen, das heißt, sie beschreiben neue, vorteilhafte Lösungen für Problemstellungen, die einem Durchschnittsfachmann beim Sichten des Standes der Technik nicht als naheliegend erscheinen. Sie können aber aus dem Stand der Technik heraus entwickelt worden sein. Die Neuheit einer Problemlösung kann auch durch Kombination bereits bekannter Lösungswege erzielt worden sein. Nach der Definition von Dasgupta wird durch die Bewertung, die das Patentamt vornimmt, einem Anmeldegegenstand Kreativität bescheinigt, bzw. der Prozeß, der zu dem Gegenstand geführt hat, als kreative Leistung anerkannt.
Kategorisiert man Patente gemäß ihrer erfindungsstrukturellen Merkmale so ergeben sich folgende Erfindungsklassen (Witte, Vollrath, 1997):

- Übertragungserfindungen
- Anwendungs- oder Verwendungserfindungen
- Auswahlerfindungen
- Kombinationserfindungen
- Fortlassungserfindungen
- *Die* Überwindung eines fachlichen Vorurteils *als Metakategorie, da für jede der vorigen Erfindungsarten möglicherweise eine Überwindung eines fachlichen Vorurteil notwendig war. Sie wird explizit aufgeführt, da sie das Urteil des Prüfers über die erfinderische Tätigkeit positiv beeinflußt.*

Es stellt sich die Frage, inwiefern der Begriff der Kreativität und der der erfinderischen Höhe des Patentwesens in Einklang zu bringen sind. Der Anspruch an die erfinderische Tätigkeit des Patentwesens ist allgemein niedriger als das allgemeine Verständnis kreativer Leistungen. Bezogen auf Dasguptas Definition von Kreativität ergibt sich der endgültige kreative Charakter jedoch gerade durch die Beurteilung des Anmeldegegenstands durch das Patentamt.

4 Unterstützung kreativer und konstruktiver Prozesse des Menschen

Es gibt eine große Anzahl von Techniken, die die menschliche Kreativität fördern sollen. Da fehlende Motivation als hinderlich für kreative Leistungen angesehen wird, muß für ein angenehmes Umfeld und Interesse an der Fragestellung gesorgt werden. Durch die Bildung von Teams können verschiedene Sichtweisen in die Bemühung ein kreatives Ergebnis zu erreichen mit einfließen. Randomisierungsmethoden (Brainstorming, Reizwortanalyse)beruhen auf dem Abbau von sogenannen Denkblockaden (psychological inertia (Kowalick, 1998)). Neben solchen Motivations- und Organisationstechniken wird versucht, durch Fokussierungstechniken und festgelegten Vorgehensmodellen (Morphologische Analyse, Quality Function Deployment (QFD), Failure Modes and Effects Analysis (FMEA)) das vorhandene Wissen zu strukturieren, zu visualisieren und neues Wissen systematisch zu erheben. Hierbei können Computerprogramme eine wertvolle Hilfestellung bieten.
Eine neuere Form der Kreativitätsförderung sind Systeme auf Basis von Änderungsmustern und -methoden, die aus der Analyse historischer Entwicklungen gewonnen wurden. Solche Ideendatenbanken wurden aufbauend auf der Theorie des erfinderischen Problemlösens (russisches Akronym TRIZ) (Terninko, 1998) zum Beispiel in die Invention Machine der Invention Machine Corp. integriert.
TRIZ wurde in den 50er Jahren von Altshuller, einem Angestellten des russischen Patentamtes entwickelt, um den kreativen Prozeß, der zu Innovationen führt, zu erklären und darauf aufbauend Erfindern helfen zu können. TRIZ beinhaltet Methoden zur Strukturierung von Systemen, Wissen darüber, wie Systeme sich im Laufe der Zeit verändern und Regeln zur Anwendung von Analogien. Bei der Analyse von Patenten entdeckte Altshuller 40 Grundprinzipien (Altshuller, 1998), mit deren Hilfe man ein bestehendes System verändern kann. Diese basieren wiederum auf der Veränderung von technischen und physikalischen Parametern, wie zum Beispiel Gewicht eines Objektes oder die Meßgenauigkeit eines Verfahrens.
Vielfach ist eine erfinderische Lösung eines Problems die Überwindung eine Konfliktsituation zwischen zwei Parametern. Der eine Parameter soll verändert werden, beeinflußt damit aber auch einen anderen Parameter, der eigentlich konstant gehalten werden soll. Als typisches Beispiel mag der Widerspruch zwischen Speicherplatz- und Zeitbedarf einer Funktion gelten. Das Ergebnis kann schnell geliefert werden, wenn es bereits im Speicher oder einer Datenbank vorliegt. Das ergibt jedoch gegebenenfalls einen hohen Speicherplatzbedarf. Anderseits kann der Speicherplatzbedarf reduziert werden, wenn das Ergebnis der Funktion erst berechnet wird, wenn es benötigt wird. Dieses führt jedoch zu einem erhöhten Zeitbedarf der Funktion.

5 Anwendung der 40 Prinzipien von TRIZ auf die Informatikdomäne

TRIZ wurde zu einer Zeit entwickelt, als die Informatik noch nicht existierte. Die analysierten Patente bezogen sich auf mechanische Vorrichtungen und den Umgang damit sowie chemische Verfahren. Es stellte sich daher die Frage, ob und wie man die Prinzipien von TRIZ auch verwenden kann, um Entwicklungen in der Informatik zu erklären bzw. voranzutreiben.
Im folgenden werden die einzelnen Prinzipien vorgestellt und anhand von Beispielen aus der Informatik erläutert. Die Übersetzung der ursprünglich englischen Begriffe wurde entsprechend (Terninko, 1998) vorgenommen. Vielfach wird deutlich, daß die Prinzipen im Hardwarebereich direkt sinnvoll einsetzbar sind, für die übrige Informatik jedoch weitreichendere Interpretationen notwendig sind. Es muß also ein neuer semantischer Rahmen geschaffen werden, um die Prinzipien in der Informatik einzusetzen. In vielen Prinzipien taucht der Begriff des „Objekts“ auf. Was ist aber das Objekt der Betrachtung in der Informatik? Wir haben versucht, durch jeweilige Ersetzung des Objektbegriffs durch Begriffe wie „Software(-architektur)“, „Hardware(-architektur)“, „Information“, „Daten“, sowie „Vorrichtung“, „System“ und „Verfahren“ Anregungen für Anwendungsbeispiele zu finden. Das Ergebnis finden Sie in den folgenden Abschnitten.
Wir wollen verschiedene mögliche Interpretationen zur Diskussion stellen, um kreative Überlegungen über die Informatik anzuregen. Die Prinzipien und Beispiele sollten jedoch nicht als Standardlösungen angesehen werden, die ein tieferes Nachdenken über Problemsituationen nicht mehr erforderlich machen.
Prinzipien, für die kein Beispiel gefunden werden konnte, werden grau dargestellt.

Prinzip 1: Segmentierung

A: Zerlege ein Objekt in unabhängige Teile.

- *Adressraum im Internet:* Durch die verschiedenen Segmente der IP-Adressierung können Router die Teile der IP-Adresse nutzen, die sie speziell zur Weiterleitung von Paketen benötigen.
- *Imperative vs. modulare/objektorientierte Programmierung:* Anstatt die zur Ausführung eines Programmes notwendigen Anweisungen in einem großen, schwer überschaubaren Block zu implementieren, werden kleinere, logisch zusammenhängende Programmteile in Klassen unterteilt. Diese Klassen (oder auch Module) dienen jeweils der Lösung eines ganz speziellen Teilproblems und lassen sich in verschiedenen Programmen einbinden (Unabhängigkeit, Wiederverwendbarkeit).

B: Mache ein Objekt zerlegbar.

- *Mounten von Festplatten unter UNIX/Linux:* Um die Speicherkapazität des Betriebssystems flexibel zu gestalten, ist es möglich, neuen (Festplatten-)Speicher zum System hinzuzufügen (zu mounten) oder abzukoppeln (unmounten).
- *Aufrüsten von RAM oder weiteren Systemkomponenten:* Analog zum Mounten von Festplattenspeicher ist es auf jedem Rechnersystem möglich (bis zu einer maximalen Obergrenze) bei Bedarf weitere Systemkomponenten in dafür zur

Verfügung stehenden Slots zwecks Verbesserung der Systemleistung hinzuzufügen.

- *Verkettete Listen:* Je nach Bedarf können zur Speicherung von Daten weitere Elemente an die Liste angehängt oder abgenommen werden. Speicherplatz wird somit im Gegensatz zu einem Array nicht statisch sondern dynamisch beansprucht und nicht im Voraus uU. überflüssigerweise allokiert.

C: Erhöhe den Grad der Zerlegbarkeit.

- *Back- und Frontends im Compilerbau:* Durch die Unterteilung in ein Back-End und ein Front-End wird die Verwendung eines Compilers ingesamt flexibler. Dadurch kann sich der Implementierer einer neuen Programmiersprache zunächst auf die Zwischencodeerzeugung konzentrieren, unabhängig davon, auf welchem Betriebssystem (Windows, Linux, Sun-OS) der Compiler später laufen soll. Ist einmal der Zwischencodealgorithmus implementiert, kann man später das Programm auf beliebige Systeme portieren. Umgekehrt kann ein bereits erzeugtes Back-End ohne größeren Aufwand mit einer anderen Programmiersprache verwendet werden.
- *Programme für Multiprozessorsysteme:* Je weiter ein Programm in separate, unabhängige Teile zerlegt werden kann, die parallel auf verschiedenen Prozessoren verarbeitet werden, desto schneller kann es ablaufen.

Prinzip 2: Abtrennung

A: Entfernung oder Abtrennung des störenden Objektes.

- *Einsatz von Filtern:* Um kostbaren Speicherplatz und Online-Zeit zu sparen, werden beim Download von E-Mails Filter vorgeschaltet, die das Downloaden von SPAM oder anderen unerwünschten Mailinhalten verhindern können. Der Einsatz von Filtern kann auch der Übersichtlichkeit im Mailordner dienen, da die E-Mails in für sie jeweils vorgegebenen Verzeichnissen abgelegt werden können.
- *Einsatz von Masken:* (Meta-)Suchmaschinen im Internet verwenden die vom Benutzer eingegebenen Begriffe zur gezielten Suche nach gewünschten Informationen. Dadurch werden unwichtige Informationen ausgeblendet (beispielsweise kann man, wenn man nach dem Wort „Blume“ sucht, und nicht die Blume auf dem Bier meint, sondern die Pflanze, z.B. zusätzlich den Begriff „Botanik“ angeben.
- *One-Click-Procedures:* Damit bei Online-Shops der Kunde nicht bei jeder neuen Bestellung seine kompletten Daten erneut eingeben und übermitteln muß, werden diese lokal gespeichert, und durch einfache Aktivierung (z.B. mittels Maus-Klick) zur Verfügung gestellt.

B: Den notwendigen Teil bzw. wesentliche Eigenschaften alleine einsetzen

- *Stationierung/Tankstellen:* In Datenbanksystemen oder Netzwerkumgebungen werden Client-Server-Strukturen verwendet, um wesentliche Informationen zentral verwaltet nutzen zu können.
- *virtuelle/abstrakte Funktionen, Klassen/Polymorphie:* Um Klassen, welche durch Vererbung miteinander verbunden sind oder die ähnliche Probleme lösen sollen, einheitlich zu strukturieren, definiert man abstrakte Klassen oder verwendet Polymorphie. Abstrakte Klassen definieren nur, welche Methoden von einer ererbten Klasse implementiert und zur Verfügung gestellt werden müssen. Mittels

der Polymorphie ist es möglich verschiedene Implementierungen einer (überladenen) Funktionen zu nutzen.

Prinzip 3: Örtliche Qualität

A: Übergang von homogener Struktur des Objektes oder seiner Umgebung zu einer heterogenen Struktur.

- *Symmetric vs. Massiv Parallel Processing:* Während beim SPP die Prozessoren der Architektur alle Daten und Speicher teilen, besitzt beim MPP jeder Prozessor seinen eigenen Speicher und Festplatte. Dadurch kann der gemeinsame Systembus entlastet werden.
- *Konservative vs. Optimistische Simulationsverfahren/Datenbankzugriffe:* Anstatt Änderungen im System nur zuzulassen, wenn feststeht, daß diese Änderungen keine Inkonsistenzen im System hervorrufen, werden gemachte Änderungen validiert und bei auftretenden Fehlern zurückgenommen.

B: Die verschiedenen Teile eines Systems sollen verschiedene Funktionen erfüllen.

- *Speicherhierarchien:* Aus Effizienz und Kostengründen unterteilt man den Speicher eines Rechnersystems in: Cache, Hauptspeicher (RAM) und Hintergrundspeicher (Festplatte).
- *Konstruktor/Destruktor:* Der Konstruktor und der Destruktor eines Objektes gehören logisch zusammen, auch wenn sie unterschiedliche (gegensätzliche) Aufgaben betreffs dieses Objektes erfüllen (erzeugen, zerstören); deshalb werden sie in der gleichen Klasse implementiert.

C: Jede Komponente eines Systems unter für sie individuell optimalen Bedingungen einsetzen.

- *Informationhiding:* Um den Zugriff auf bestimmte Variablen oder Methoden einer Klasse bzw. eines Objektes zu verhindern, werden diese durch Kapselung (private, protected, public etc.) geschützt oder zur Verfügung gestellt.

Prinzip 4: Asymmetrie

A: Ersetze symmetrische Formen durch asymmetrische.

- *Takten:* Statt synchroner Kommunikationsformen, wie sie z.B. bei Busprotokollen vorkommen, werden asynchrone Kommunikationsprotokolle eingesetzt.
- *Schleifen vs. Sprünge:* Unter Umständen kann es von Vorteil sein, statt {\bf while} oder {\bf for}-Schleifen besser {\bf if}- oder {\bf case/switch}-Anweisungen zu verwenden.
- *Public Key-Verschüsselung:* Statt der Verwendung eines Schlüssels, der sowohl der Ver- als auch der Entschlüsselung einer Nachricht dient, und deshalb nur den Personen zur Verfügung stehen darf, für die jene Nachricht gedacht ist, werden zwei verschiedene Schlüssel verwendet. Den private key kennt nur sein jeweiliger Besitzer, nur mit diesem key können Nachrichten entschlüsselt werden, die vorher mit dem zugehörigen public key verschlüsselt wurden. Weiter garantiert die Entschlüsselung einer Nachricht mit einem entsprechenden public key, daß diese nur von dem Besitzer des zugehörigen private key erstellt und verschlüsselt werden konnte.

B: Erhöhe den Grad an Asymmetrie, wenn diese schon vorliegt

- *(Automatische) Schrittweitenregulierung:* Bei numerischen Integrationsverfahren in Simulatoren ist es sinnvoll keine feste Schrittweite vorzugeben, sondern mittels automatischer Schrittweitenregulierung den nächsten Integrationsschritt zu berechnen. Dies erhöht die Genauigkeit der Integration und verhindert überflüssige Berechnungen.
- *Von zeitdiskreten zu ereignisdiskreten Modellklassen:* Statt zeitdiskrete Modelle mit äquidistanten Zeitabschnitten in der Simulation von diskreten Modellteilen einzusetzen, kann es sich lohnen ereignisdiskrete Modelle zu verwenden, in denen die Simulationszeit in Abhängigkeit zum Auftreten von Ereignissen (in der Ereignisliste) voranschreitet.

Prinzip 5: Vereinen

A: Guppiere gleichartige oder zur Zusammenarbeit bestimmte Objekte räumlich zusammen

- *Pipelining:* Um die Systemressourcen optimal auszunutzen, werden Daten parallel (fließbandartig) abgearbeitet, so daß keine Pausen in irgendwelchen Teilen des Systems entstehen.
- *Multiprozessorsysteme:* Zur Verbesserung des Systemleistung wird statt eines Prozessors ein Multiprozessorsystem eingesetzt.
- *Rechnernetze:* Statt der Ausnutzung eines Systems werden zur Verbesserung der Systemleistung mehrere Rechner gekoppelt.

B: Vereine gleichartige oder zur Zusammenarbeit bestimmte Objekte, d.h. kopple sie zeitgleich

- *Multitasking/Multiusersysteme:* Zur gleichen Zeit kann ein System mehrere Programme ausführen oder mehrere Nutzer können zur gleichen Zeit auf ein System arbeiten.

Prinzip 6: Universalität

A: Das System erfüllt mehrere unterschiedliche Funktionen, wodurch andere Systeme oder Objekte überflüssig werden.

- *Emulatoren:* Ein Emulator stellt auf einem bestehenden Betriebssystem ein anderes (emuliertes) Betriebsystem und die damit zusammenhängenden Befehle und Eigenschaften zur Verfügung.
- *Videokarte im PC:* Eine Videokarte in einem PC ermöglicht das Fernsehen auf einem PC-Monitor und macht dadurch den Fernseher überflüssig.
- *Designpattern:* Dabei handelt es sich um Verfahrensbeschreibungen, nach denen z.B. ein Programmierer vorgehen kann, um wiederverwendbare Software zu erstellen.

Prinzip 7: Verschachtelung

A: Ein Objekt befindet sich im Inneren eines anderen Objektes, das sich ebenfalls im Inneren eines dritten befindet.

- *Rekursion:* Rekursion ermöglicht einen verschachtelten Aufruf des gleichen Codes mit anderen Parametern. Der Vorteil dieses Verfahrens besteht darin, daß der Codeaufwand eines Programms auf diese Weise klein gehalten wird.

- *Frames in Webseiten:* Durch Frames können unabhängige Teilseiten in einer Webseite erstellt werden und dadurch z.B. die Navigation vereinfacht werden.

B: Ein Objekt paßt in oder durch den Hohlraum eines anderen.

- *Komprimierung:* Komprimierung erlaubt schnellere Datenübertragung und ermöglicht bessere Ausnutzung von Speichern.
- *Steganographie*: Durch Steganographie können Nachrichten in Bilddaten verschlüsselt versendet werden. Es wird dabei ausgenutzt, daß die geringfügingen Änderungen in den Daten für das menschliche Auge nicht auszumachen sind.

Prinzip 8: Gegengewicht

A: Das Gewicht des Objekts kann durch Kopplung an ein anderes, entsprechend tragfähiges Objekt kompensiert werden.

- *Customer/Producer-Lösungen:* Der Erzeuger-Prozeß schreibt Werte in einen gemeinsamen Speicher (Puffer), die der Verbraucher-Prozeß auslesen kann. Bedingung hierbei ist, daß der Erzeuger den Puffer nur mit Werten füllen kann, wenn dieser leer ist, und der Verbraucher erst mit dem Auslesen der Werte beginnen darf, bis der Erzeuger den Puffer aufgefüllt hat (Petri-Netze).

B: Das Gewicht des Objekts kann durch aerodynamische oder hydraulische Kräfte kompensiert werden.

- *Vakuum im Festplatteninneren:* Damit der Schreib-/Lesekopf nicht die Oberfläche der Festplatte beschädigt, wird er durch eine Vacuum im Festplatteninneren in der richtigen Position gehalten.

Prinzip 9: Vorgezogene Gegenaktion

A: Vor der Ausführung einer Aktion muß eine erforderliche Gegenaktion vorab ausgeführt werden.

- *Antivirenimpfung von Programmen:* Um Programme resistent gegen bestimmte Typen von Computerviren zu machen, können diese vorher geimpft werden.
- *Initialisierung von Variablen:* Variablen werden vorab mit Werten initialisiert, damit keine Laufzeitfehler auftreten.

B: Muß ein Objekt in Spannung sein, dann muß vorab die Gegenspannung erzeugt werden.

Prinzip 10: Vorgezogene Aktion

A: Führe die notwendige Aktion - teilweise oder ganz - im voraus aus.

- *Automatisches Zwischen-)Speichern:* eines Dokumentes, das früher oder später sowieso gespeichert wurden wäre.
- *Look ahead caching:* z.B. bei Prozessoren oder Web Cache, werden schon mal vorzeitig weitere Daten geladen, die möglicherweise bald gebraucht werden.

B: Ordne Objekte so an, daß sie ohne Zeitverlust vom richtigen Ort aus arbeiten können.

- *Trojanische Pferde:* (möglicherweise schon bei der Installation des Betriebssystems mit installiert) erleichtern die Spionage auf betriebsfremden Rechnersystemen erheblich, und liefern (wenn entsprechend programmiert) vom richtigen Ort ohne Zeitverlust (der durch einen Einbruch in den jeweiligen Betrieb oder das Aushorchen von Angestellten einhergehen würde) wenn vorhanden die

gewünschten Informationen. (Man sieht hier, daß Innovationen leider nicht immer positiv sind)

Prinzip 11: Vorbeugemaßnahmen

A: Kompensiere die schlechte Zuverlässigkeit eines Systems durch vorher ergriffene Gegenmaßnahmen.

- *Exception(handling):* wird verwendet, um mögliche Fehler in einem Programm abzufangen und so zu behandeln, daß das Programm beim Auftreten eines Fehlers (noch) korrekt weiterarbeiten kann.
- *Sicherheitshardware:* Damit Systemkomponenten im Rechner nicht aufgrund von Überhitzung ausfallen, werden vorbeugend Lüfter eingebaut.
- *Backups:* welche in regelmäßigen (automatisch) durchgeführt werden, können helfen wenigstens einen gewissen Teil an Daten zur Verfügung zu stellen, sollten diese (ausversehen) durch Löschung oder jedweilige Systemabstürze verlorengehen.
- *Sicherheitsmarkierungen:* Es gibt Protokolle, die festlegen, daß ein Prozessor erst auf den Bus zugreifen darf, wenn er sich beim jeweiligen Arbiter einen Grant (eine Erlaubnis-Token) abgeholt hat. Das Grant signalisiert, daß der Prozessor, welcher dieses Grant besitzt, das Recht hat, auf den Bus zuzugreifen.
- *Deadlock-Detection:* In Betriebsystemen findet man Verfahren, die Deadlocks bei der Vergabe von Ressourcen frühzeitig genug erkennen, um sie zu verhindern.

Prinzip 12: Äquipotential

A: Verändere die Bedingungen so, daß das Objekt mit konstantem Energiepotential arbeiten kann, also weder angehoben, noch abgesenkt werden muß.

- *Zeiger in Programmiersprachen:* Um Speicherplatz zu sparen, wird anstelle des Kopierens einer Variablen ein Zeiger auf das Original gesetzt.

Prinzip 13: Umkehrung

A: Implementiere anstelle der durch Spezifikation diktierten Aktion genau die gegenteilige Aktion.

- *positive vs. negative Gatter:* Statt zwei positive Gatter (OR, NOT) zur Implementierung der gewünschten Funktion einzusetzen, wird nur ein negatives Gatter (NOR) zur Implementierung der gleichen Funktion eingesetzt. Die NOR-Gatters ist kostengünstiger und schneller.

B: Mache ein unbewegliches Objekt beweglich oder ein bewegliches unbeweglich.

- *Variablen/Konstanten:* Nutze Variablen anstelle von Konstanten bzw. umgekehrt.

C: Stelle das System „auf den Kopf“, kehr es um.

- *Maus vs. Trackball:* Statt die Maus zu bewegen, um eine Bewegung der Cursor-steuernden Kugel im Mausinneren zu erreichen, wird beim Trackball, direkt die Kugel, welche zur Messung und Bestimmung der aktuellen Cursorposition eingesetzt wird, bewegt.
- *Top-Down- vs. Buttom-Up-Verfahren:* Vielfach können bei Analyse oder Planung alternativ Top-Down- oder Buttom-Up-Verfahren angewendet werden.

Prinzip 14: Krümmung

A: Ersetze lineare Teile oder flache Oberflächen durch gebogene, kubische Strukturen druch sphärische.

- *Maus und Trackball vs. Grafiktablett*

B: Benutze Rollen, Kugeln, Spiralen. sphärische.

- *Maus und Trackball*

C: Ersetze lineare Bewegungen durch rotierende, nutze die Zentrifugalkraft.

- *Festplatte und CD vs. Band:* Die kreisförmige Struktur der Festplatte erlaubt einen schnelleren Zugriff auf verschiedenste (im örtlichen Sinne) Daten, als das Bandlaufwerk, welches unter Umständen erst zu den gewünschten Daten hinspulen muß, das kostet Zeit.

Prinzip 15: Dynamisierung

A: Gestalte ein System oder dessen Umgebung so, daß es sich automatisch unter allen Betriebszuständen auf optimale Performance einstellt.

- *dynamische, flexible Bandbreite(nanpassung):* Bei geringer Netzwerkkapazität wird die Pixelanzahl eines zu übertragenden Videos reduziert, damit es trotzdem noch übertragen werden kann.

B: Zerteile ein System in Elemente, die sich untereinander optimal arrangieren können.

- *Costumer-Setup oder benutzerspezifische Arbeitsoberflächen:* Bei der Installation von Software gibt es verschiedene Konfigurationsmöglichkeiten (Standard-, Costumer-, light-Installation), welche der Benutzer je nach seinen Konfigurationswünschen verwenden kann. Benutzerdefinierte Arbeitsoberflächen erlauben die für den jeweiligen Benutzer optimale Verwendung des Systems.
- *Mikrooperationen:* sind Implementierungen atomarer Befehle auf Registertransferebene. Z.B. kann einen Mikrooperation beschreiben, von welchem Register, welche Werte, über welchen Weg (durch Schaltung von entsprechenden Signalen) zu welchen Registern übertragen werden sollen. Kombiniert man nun diese Mikrooperationen können Befehle wie LOAD, BEQZ oder ADD flexibel erzeugt und von einem Compiler genutzt werden.

C: mache ein unbewegliches Objekt beweglich, verstellbar oder austauschbar.

- *CD-Wechsler oder Wechselfestplatten:* Wenn kein Netzwerk zur übertragung gespeicherter Daten zur Verfügung steht, oder Diskette in ihrer Speicherkapazität zu klein sind, um die gewünschten Daten auf ein anderes Rechnersystem ökonomisch zu übermitteln, kann der flexible Einsatz einer wechselbaren Festplatten von kostengünstigen Nutzen sein.

Prinzip 16: Partielle oder überschüssige Wirkung

A: Wenn es schwierig ist, 100% einer geforderten Funktion zu erreichen, verwirkliche etwas mehr oder weniger, um so das Problem deutlich zu vereinfachen.

- *Codeoptimierung:* Beim Compileren eines Quelltextes optimiert der Compiler automatisch entsprechende Anweisungen im Quelltext zwecks besserer Performance.

Prinzip 17: Höhere Dimension

A: Umgehe Schwierigkeiten bei der Bewegung eines Objektes entlang einer Linie durch eine zweidimensionale Bewegung (in einer Ebene). Analog wird ein Bewegungsproblem in der Ebene vereinfacht durch Übergang in die dritte Dimension.

- *Holographische (3D)-Speicher.*

B: Ordne Objekte in mehreren statt in einer Ebene an.

- *Lineare Listen und Vektoren vs. (mehrdimensionale) Arrays.*
- *Nutzung von Metainterpretern*

C: Plaziere das Objekt geneigt, oder kippe es.

- *Desktop vs. Tower.*

D: Nutze Projektionen in die Nachbarschaft oder auf die Rückseite des Objektes.

- *Höhere Programmiersprachen vs. Assembler:* Der in einer höheren Programmiersprache formulierte Quelltext wird von einem Compiler in Assembler abgebildet/übertragen.

Prinzip 18: (Mechanische) Schwingungen

A: Versetze ein Objekt in Schwingungen.

- *Endlosschleifen:* im übertragenen Sinne können Endlosschleifen in einem Programm als Oziallation aufgefaßt werden: die mehr oder weniger (un)erwünschte Wiederholung ein und desselben Vorgangs. Bei Controllern (oder anderen Automaten) können sie sinnvoll eingesetzt werden.

B: Ozilliert das Objekt bereits, erhöhe seine Frequenz.

- *Taktfrequenz eines Rechnersystems.*

C: Benutze die Resonanzfrequenz(en).

D: Ersetze mechanische Schwingungen durch Piezovibrationen.

E: Setze Ultraschall in Verbindung mit elektromagnetischen Feldern ein.

Prinzip 19: Periodische Wirkung

A: Übertragung von kontinuierlicher zu periodischer Wirkung.

- *Time-Division Duplex:* wird auf einem Kanal verwendet, um statt Simplex Halbduplex zu simulieren.

B: liegt bereits eine periodische Aktion vor, verändere deren Frequenz.

- *Die Clock:* des PCs.

C: Benutze Pausen zwischen einzelnen Impulsen, um andere Aktionen einfügen zu können.

- *Hidden arbitration:* während auf dem Bus eine Transaktion stattfindet, kann bereits ein neuer Arbitrierungszyklus beginnen. So werden Pausen (während des Wartens auf das Freiwerden des Busses) verringert und sinnvoll ausgenutzt.
- *Seti@Home-Projekt*: Beim Seti@Home Projekt wird die Prozessorleistung eines ans Internet angeschlossenen Rechners in den Pausen des Benutzers dazu verwendet, Radio-Teleskopdaten auf außerirdische Intelligenz hin zu untersuchen.

Prinzip 20: Kontinuität

A: Führe eine Aktion ohne Unterbrechung aus, alle Komponenten sollen ständig mit gleichmäßiger Belastung arbeiten.

- *Schreibende Datenbankzugriffe:* dürfen nicht unterbrochen werden. Sie werden erst ausgeführt, wenn klar ist, daß durch diese Aktion keine Inkonsistenzen auftreten werden.

B: Schalte Leerläufe und Unterbrechungen aus.

- *Exceptions oder Interrupts:* Treten Exceptions (z.B. eines Device-Handlers) auf, während die CPU ihren Status speichert, werden diese zwar vermerkt, aber vorerst nicht behandelt. Auch hier wird die schreibende Aktion zwecks Vermeidung von Inkonsistenzen nicht unterbrochen.

C: Ersetze eine „vor-und-zurück"-Bewegung durch eine rotierende.

- *Bandlaufwerk vs. Festplatte.*
- *FIFO statt LIFO.*

Prinzip 21: Überspringen

A: Führe schädliche oder gefährliche Aktionen mit sehr hoher Geschwindigkeit durch.

- *Rapid-Prototyping:* (schädlich = hier: die hohen Entwicklungskosten) setzt man ein, um nicht erst nach einer langen Programmentwicklung feststellen zu müssen, daß das Programm nicht die vom Kunden gewünschte Funktionalität bietet.

Prinzip 22: Schädliches in Nützliches wandeln.

A: Nutze schädliche Faktoren oder Effekte – speziell aus der Umgebung – positiv aus.

- *Defragmentieren durch Überschreibung der Daten mit 0:* leere Cluster können mit Nullen überschrieben werden. Überschreiben bedeutet zwar den Verlust jener Daten, die überschrieben werden, in diesem Fall dient das Überschreiben aber einem positivem Zweck, indem Bereiche gut erkennbar als frei markiert werden und betreffs der (unerwünschten) Wiederherstellung der nun mit Null überschrieben Daten ein (kleiner) Riegel vorgeschoben wurde.

B: Beseitige einen schädlichen Faktor durch Kombination mit einem anderen schädlichen Faktor.

C: Verstärke einen schädlichen Einfluß soweit, bis er aufhört, schädlich zu sein.

- *SATAN:* ein Programm, welches Sicherheitslücken in einem System aufdecken kann, indem es versucht, in das System einzubrechen. Schädlich aufgrund dessen, daß Hacker es genauso benutzen können wie die Administratoren des jeweiligen Systems. Nützlich, da aufgedeckte Schwächen erkannt und behoben werden können.

Prinzip 23: Rückkopplung

A: Führe eine Rückkopplung ein.

- *Energiesparmodus:* Nach einer gewissen Zeit, in der keine Eingaben getätigt wurden, stellt sich das System auf „standby".

B: Ist eine Rückkopplung vorhanden, ändere sie oder kehre sie um.

- *Netzwerkprotokolle:* handshake etc.

Prinzip 24: Mediatoren, Vermittler

A: Nutze ein Zwischenobjekt, um eine Aktion weiterzugeben oder auszuführen.

- *Router:* helfen bei der Paketweiterleitung.
- *Arbiter:* können z.B. den Zugriff von CPUs auf den Bus steuern.

- *Agentensysteme:* helfen z.B. bei der Suche nach bestimmten Informationen.

B: Verbinde das System zeitweise mit einem anderen, leicht zu entfernenden Objekt.

- *Lokale Variablen:* zur Speicherung temporär-interessierender Werte.
- *Design Pattern " Mediator":* Korrespondierendes Design-Pattern ermöglicht die Wiederverwendbarkeit von Software

Prinzip 25: Selbstversorgung

A: Das System soll sich selbst bedienen und Hilfs- sowie Reparaturfunktionen selbst ausführen.

- *Selbst-Diagnose:* Programme, die beim Booten des Rechnersystems oder beim Scannen benutzt werden.

B: Nutze Abfall und Verlustenergie.

- *Zur sicheren Löschung von Daten, werden sie mit unsinnigen Daten überschrieben*

Prinzip 26: Prinzip 26: Kopieren

A: Benutze eine billige, einfache Kopie anstatt eines komplexen, teueren, zerbrechlichen oder schlecht handhabbaren Objektes.

- *Simulatoren/Simulation:* zwecks Beobachtung und Bewertung von Systemverhalten.
- *Unix-Filesystem*: Anstatt auf den Originaldateien arbeitet UNIX immer auf Kopien der Dateien. Erst beim Speichern wird die Originaldatei verändert.

B: Ersetze ein System oder ein Objekt durch eine optische Kopie oder Abbildung. Hierbei kann der Maßstab (vergrößern, verkleinern) verändert werden.

- *Touchscreen statt Tastatur*.

C: Werden bereits optische Kopien benutzt, dann gehe zu infrarot oder ultravioletten Abbildungen über.

Prinzip 27: Billige Kurzlebigkeit

A: Ersetze ein teures System durch ein Sortiment billiger Teile, wobei auf einige Eigenschaften (Langlebigkeit beispielsweise) verzichtet wird.

- *PIN und TAN.*
- *Zufallszahlen.*

Prinzip 28: Mechanik ersetzen

A: Ersetze ein mechanisches System durch ein optisches, akustisches oder geruchsbasierendes System.

- *CD(-ROM) statt HDD***.**

B: Benutze elektrische, magnetische oder elektromagnetische Felder.

- *HDD statt Lochkarten.*

C: Ersetze Felder: statinäre durch bewegliche, konstante durch periodische, strukturlose durch strukturierte.
D: Setze Felder in Verbindung mit ferromagnetischen Teilchen ein.

Prinzip 29: Pneumatik und Hydraulik

A: Ersetze feste, schwere Teile eines Systems durch gasförmige oder flüssige. Nutze Wasser oder Luft zum Aufpumpen, Luftkissen, hydrostatische Elemente.

- *Verwendung bio-chemischer Speicher.*

Prinzip 30: Flexible Hüllen und Filme

A: Ersetze übliche Konstruktionen durch flexible Hüllen oder dünne Filme.

- *Wrapper:* Durch Definition einer allgemeinen Schnittstelle ist es möglich Anwendungssoftware auf verschiedenen Betriebssystemen wiederverwendbar zu machen.
- *Adaptive Benutzungsschnittstellen, Adaptive Portale:* Informationen werden entsprechen den Vorlieben des Benutzers angepaßt und visualisiert.

B: Isoliere ein Objekt von der Umwelt durch einen dünnen Film oder eine Membran.

- *Firewalls:* Firmenintranets werden durch eine Firewall vor Störungen und Eindringlingen von außen geschützt. Es handelt sich dabei um eine imaginäre, softwarebasierte Schutzhülle um die Firma.

Prinzip 31: Poröse Materialien

A: Gestalte ein Objekt porös oder füge poröse Materialien (Einsätze, Überzüge...) zu.

- *Puffer in Rechnernetzen:* In Rechnernetzen spielen Puffer zum Sichern von Paketen für den Fall, daß zu viel Pakete auf einmal kommen, eine wichtige Rolle.

B: Ist ein Objekt bereits porös, dann fülle die Poren mit einem vorteilhaften Stoff im voraus.

- *Halbleiter:* Durch Hinzufügen bestimmter Atome wird die Leitfähigkeit erhöht.

Prinzip 32: Farbänderungen

A: Verändere die Farbe eines Objektes oder die der Umgebung.

- *Laser im CD-ROM-Laufwerk:* Verwendung verschiedenfarbener Laser.
- *Farbunterschied zwischen aktiven und nichtaktiven Anwendungen in GUIs.*

B: Verändere die Durchsichtigkeit eines Objektes oder die der Umgebung.

- *Abstrakte Datentypen:* In ADTs wie auch in der objektorientireten Programmierung können sichtbare (public) als auch unsichtbare (private) Elemente existieren.

C: Nutze zur Beobachtung schlecht sichtbarer Objekte oder Prozesse geeignete Farbzusätze.

- *Colorierte Petri-Netze:* ermöglichen die explizite Repräsentation verschiedeneartiger Objekte in einem Petri-Netz.

D: Existieren derartige Farbzusätze bereits, setze Leuchtstoffe, lumineszente oder anderweitige markierte Substanzen ein. .

- *Blinken, Bewegung in Benutzungsoberflächen:* verwende auffällige Farben oder bewegte/blinkende Objekte um Aufmerksamkeit zu erregen.

Prinzip 33: Homogenität

A: Fertige interagierende Objekte aus demselben oder aus ähnlichem Material.

- *Metadaten:* werden eingesetzt, um Daten unterschiedlichster Herkunft (einheitlich) verwenden zu können, siehe OIM, Repositorys, Datacleansing oder Internetsuchmaschinen.

Prinzip 34: Beseitigung und Regeneration

A: Beseitige oder verwerte (ablegen, auflösen, verdampfen) diejenigen Teile des Systems, die ihre Funktion erfüllt haben oder unbrauchbar geworden sind.

- *Pakete:* welche Daten über ein Netzwerk transportieren, werden nach der Übermittlung zerstört.

B: Stelle verbrauchte Systemteile unmittelbar – im Arbeitsvorgang – wieder her.

- *Token-Erzeugung* (z.B. Petrinetze oder Daisy-Chain): Token dienen hier der Berechtigung zu einem Buszugriff oder der Schaltung einer Transition, sind sie verbraucht, werden (auf Anfrage) neue erzeugt.

Prinzip 35: Eigenschaftsänderung

A: Ändere den Aggregatzustand eines Objektes: fest, flüssig, gasförmig, aber auch quasiflüssig oder ändere Eigenschaften wie Konzentration, Dichte, Elastizität, Temperatur.

- *Änderung der physikalischen Dichtevon Daten auf Festplatten*: Festplatten neueren Typs speichern Daten physikalisch dichter als alte Festplatten. Dadurch erhöht sich der Grad an Speicherkapazität sowie die Zugriffsgeschwindigkeit auf die Daten.
- *Defragmentierung*: Bei der Defragmentierung einer Festplatte werden die vorhandenen Daten verdichtet, um größere Blöcke am Stück frei zu haben und beim Zugriff auf die Daten unnötige Bewegungen der Leseeinheit zu vermeiden.

Prinzip 36: Phasenübergänge

A: Nutze die Effekte während des Phasenüberganges einer Substanz aus: Volumenänderung, Wärmeentwicklung oder -absorption.

Prinzip 37: Wärmeausdehnung

A: Nutze die thermische Expansion oder Kontraktion von Materialien aus.

B: Benutze Materialien mit unterschiedlichen Wärmeausdehnungskoeffizienten.

Prinzip 38: Prinzip 38: Starkes Oxidationsmittel

A: Ersetze normale Luft durch sauerstoffangereicherte Luft.

B: Ersetze angereicherte Luft durch reinen Sauerstoff.

C: Setze Luft oder Sauerstoff ionisierenden Strahlen aus.

D: Benutze Ozon.

Prinzip 39: Inertes (veraltet für: träges) Medium

A: Ersetze die übliche Umgebung durch eine inerte.

B: Führe den Prozeß im Vakuum aus.

Prinzip 40: Verbundmaterial

A: Ersetze homogene Stoffe durch Verbundmaterialien.

- *Array vs. Record:* Records können Daten verschiedenster Typen speichern, während Arrays nur Daten eines bestimmten Datentyps speichern können

7 Diskussion

Durch den hohen Abstraktionsgrad einiger Prinzipien aus TRIZ sind diese problemlos auch in der Informatik anwendbar und wurden auch bereits angewendet, wie in den vorigen Abschnitten gezeigt werden konnte. Sie können somit auch in der Informatik als Ideenpool genutzt werden. Einige Prinzipien basieren jedoch stark auf Effekten aus der Natur und sind somit im Hardwarebereich, weniger aber in der Welt der Algorithmen und Verfahren einsetzbar.
Für die Zukunft wäre eine Analyse der informations- und softwaretechnischen Verfahren interessant, um in diesen speziellen Gebieten *neue Innovationsprinzipien* aufzudecken und diese auch prospektiv für die Unterstützung von Informatikern bei der Entwicklung von Erfindungen einsetzen zu können.

8 Referenzen

Altshuller, G., “40 Priciples: TRIZ Keys to Technical Innovation”, TRIZ Tools, Vol. 1, Worcester, MA: Technical Innovation Center, 1998.

Boden, M., „The Creative Mind”, London: Abacus, 1990.

Csikszentmihalyi, M., „Kreativität“, Stuttgart: Klett-Cotta, 1997.

Dasgupta, S., „Technology and Creativity”, Oxford: Oxford University Press, 1996.

Kowalick, J., „Psychological Inertia“, in: TRIZ Journal, August, 1998.

Kurzweil, R., “Das Zeitalter der künstlichen Intelligenz”, München:Carl Hanser Verlag, 1993.

Langley, P., Simon, H.A., Bradshaw, G.L, Zytkow, J.M., „Scientific Discovery“, MIT Press, 1987.

Plsek, P.E., ”Creativity, Innovation and Quality”, Milwaukee: ASQC Quality Press, 1997.

Simonton, D.K., „Scientific Genius“, Cambridge University Press, 1990.

Strube, G., “Wörterbuch der Kognitionswissenschaft”, Stuttgart: Klett-Cotta, 1996.

Terninko, J., “TRIZ – der Weg zum konkurrenzlosen Erfolgsprodukt”, R.Herb (Hrsg.), Verlag Moderne Industrie, 1998.

Willms, J., “Eine intelligente Problemlöseumgebung auf Basis kooperativen Hypothesentestens im Patentwesen”, in Proc. 9. Arbeitstreffen der GI-Fachgruppe 1.1.5/7.0.1 "Intelligente Lehr-/Lernsysteme", LWA99 Sammelband, S. 333-343, Univ. Magdeburg, September 1999

Willms, J., Möbus, C., “Evolution of the Hypotheses Testing Approach in Intelligent Problem Solving Environments”, Proceedings of ITS 2000, Fifth International Conference on Intelligent Tutoring Systems, 19.06.2000-23.06.2000, Montreal, Canada, 2000

Wilson, R.A., Keil, F.C., “The MIT Encyclopedia of the Cognitive Sciences”, London: MIT Press, 1999.

Witte, J., Vollrath, U., “Praxis der Patent- und Gebrauchsmusteranmeldung”, C. Heymanns Verlag, Köln, 1997.