Der Sinn des Code

Peter J. Bentley

Computerprogramme haben sich mittlerweile durchgesetzt. Einst das Revier verschrobener Mathematiker, ist der Code, der in jedem Computer steckt, mittlerweile so alltäglich wie das Auto, in dem wir umherfahren. Jeder scheint das World Wide Web zu kennen, gleichgültig, ob er auf dessen stürmischen Wellen gesurft ist oder nicht. Jeder weiß, was E-Mails sind, egal, ob er allmorgendlich mit Spams überhäuft wird oder nicht ein einziges Mal ein Millionenangebot von einem Cousin eines verstorbenen nigerianischen Präsidenten erhalten hat. Alle reden bei fensterorientierten Betriebssystemen mit und kennen zumindest drei Schauergeschichten, aus denen die Opfer eines Computerabsturzes traumatisiert hervorgingen. Jedes Kind weiß, dass die schlimmsten *Bugs* jene hartnäckige Spezies sind, denen man nicht mit Insektensprays zu Leibe rücken kann. Software steuert unsere Welt. Sie sorgt dafür, dass unser Geld sicher ist, der Motor in unserem Wagen läuft, wir miteinander telefonieren können und Fernsehsendungen und Filme uns reibungslos unterhalten. Software verändert unser Leben. Darüber gibt es sogar Filme. Der Satz „Sie haben Post“ wird niemals wieder dasselbe bedeuten. Computer sind überall, und damit auch ihre Sprache: Code.

Irgendwie erscheint im Code alles möglich. Es gibt keine Zweideutigkeiten, keine versteckten Bedeutungen. Probleme werden meisterlich bis ins Kleinste seziert und Lösungen bereitgestellt, die den Computer wie von Zauberhand die notwendigen Schritte tun lassen. Ein guter Programmierer ist ein Virtuose. Seine Meisterstücke sind Kunstwerke – vergleichbar einem Konzert oder Gedicht. Doch sein ganzes Schaffen steckt unsichtbar im Inneren eines Computers, und dem Publikum bleibt sein Genie auf immer verborgen. Ein schlechter Programmierer (und davon gibt es wahrlich genügend) ist nur ein „Hansdampf in allen Gassen“, kaum kreativer oder talentierter als jemand, der sich in „Malen nach Zahlen“ übt. Die Ergebnisse sind schauerlich und nur von kurzem Bestand.

Hat man lange genug Computer programmiert, ist also quasi damit „aufgewachsen“, dann eignet man sich eine differenzierte Denkweise an. Es erscheinen einem jedoch keine Einsen und Nullen vor den Augen, wie in jener unvergesslichen Szene im Film *The Matrix.* Es ist etwas subtiler. Man gewöhnt sich an, Probleme in kleinere, leichter lösbare Einheiten zu zerlegen. Das wird

zur natürlichen Denkart. Und dabei ist es egal, ob man einen Roboter konstruieren oder von einem Baum herunterklettern will. Gute Programmierer sind Problemlösungstalente, denn so schreibt man Code. Code kann einen aber auch entmenschlichen. Code lässt keinen Platz für Feinsinniges, Humor oder Emotionen. Der Programmierer muss zwar kreativ und künstlerisch sein, er muss aber auch sehr buchstabengetreu arbeiten. Funktioniert etwas nicht, dann liegt es nicht daran, dass der Computer ärgerlich ist, die Anweisungen missverstanden hat oder sich langweilt, sondern dass der Programmierer verärgert war, seinen eigenen Code falsch verstanden oder sich gelangweilt hat. Code ist so wörtlich, so unzweideutig, dass man seinen Verstand erst darauf trainieren muss, ebenso zu denken. Diese Einschränkungen des Codes zeigen mitunter Nebenwirkungen bei den Menschen, die ihn schreiben – ein Witz kommt nicht an, ein philosophischer Gedanke wird verpasst, oder eine Zweideutigkeit verursacht unbeschreibliche Verwirrung. Auch mir ging es einmal so (und geht es gelegentlich noch immer). Ich habe aber gelernt, die Kunst zu schätzen und die Mehrdeutigkeit der Sprache zu lieben. Die Auswirkungen, die reiner, nackter Code auf die Menschen hat, sind nicht immer günstig. Sagen Sie einmal einem Programmierer absichtlich etwas Zweideutiges und beobachten Sie, wie lange er braucht, um die Bedeutung zu verstehen. Sagen Sie dasselbe einem Künstler, und er wird im Nu antworten.

Dennoch stecken die meisten von uns nicht tagtäglich bis über beide Ohren in Code; es steigt vielmehr die Anzahl derer, die den Code nutzen. Wir sind die *User* (eine sehr abwertende Bezeichnung für einen Programmierer) und müssen uns mit all den Möglichkeiten, die uns Software bietet, herumschlagen. Sie gibt uns so viel Spielraum, so viele Optionen, dass der Einsatz von Software unsere Arbeit zur bloßen Informationssuche verkommen lässt. Wir stöbern in Hunderten von Funktionen unseres Textverarbeitungsprogramms (von denen sich die Hälfte zum Glück selbst unsichtbar macht), um ein einfaches Zeichen wie ein „ï“ einzufügen. (Über die Menüs meiner Textverarbeitung habe ich dafür 90 Sekunden gebraucht. Und dabei wusste ich, wie es geht. Mit einem Bleistift könnte ich dasselbe Zeichen in weniger als einer Sekunde schreiben.) Wir durchforsten Tausende Funktionen eines Grafikprogramms, nur um eine Linie mit einer Pfeilspitze zu erstellen. Im Zug der E-Mail-Inflation der letzten Jahre suchen wir in einem Wust unerwünschter Spam-Mails nach den wenigen wirklichen Nachrichten, die darunter verborgen sind. Und obwohl wir Internet-Suchmaschinen einsetzen, durchsuchen wir die Ergebnisse erst Recht wieder nach brauchbaren Dingen. Informationsüberschuss ist eine weitere Nebenwirkung von Code.

Code schadet aber nicht nur den Menschen, sondern in zunehmenden Maß auch den Computern. Software wird kaum noch von Programmiervirtuosen geschrieben. Code wird zur „Bloatware“ – ineffizient, langsam, randvoll mit veraltetem Code (so genanntem Legacy Code). Statt dem Beispiel der Mikrochip-Industrie zu folgen, die immer kleinere, schnellere, effizientere und billigere Prozessoren erzeugte, wählten die Softwarefirmen ein unsinniges Modell und fügten immer neue Funktionen hinzu. Ähnlich wie bei einem schlecht konzipierten Wagen, der mit immer mehr Extras ausgestattet wird, bis er trotz seines extrem leistungsstarken Motors kaum noch zu fahren ist. Manchen kommt das bekannt vor? Werden nicht auch unsere Gesetze so geschrieben? Der Kunst oder der Biologie jedoch ist diese ineffiziente Anhäufung von unnötigem Code fremd. Die beste Kunst ist radikal, neu und beruht auf einem einfachen Konzept. Und obwohl die Gene aller Lebewesen auf kumulative Weise geschrieben werden, so ist Effizienz dennoch das oberste Gebot – sonst können sie nämlich nicht überleben. Aus der Sicht eines Computerfachmanns, der mit Code groß geworden ist, ist der Großteil der derzeit angebotenen Software schlicht scheußlich. Code ist fantasielos geworden, ohne Effizienz und Eleganz. Heute werden nur noch vereinzelt Meisterleistungen erbracht. Aktuelle Software von der Stange hat überhaupt keinen Pepp mehr.

Ein Grund für den Niedergang von Code liegt in der so genannten Komplexitätsgrenze. Software ist mittlerweile so komplex, dass selbst ein riesiges Expertenteam nicht in der Lage ist

zu prüfen, wie alles funktioniert – oder *ob* überhaupt alles funktioniert. Es gibt so viele Elemente (Subroutinen, Module, Dateien, Variablen), die auf unterschiedlichste Weise miteinander interagieren, dass wir damit einfach überfordert sind. Daher wird Software heute nicht als voll funktionsfähiges Produkt ausgeliefert, sondern nach und nach mit „Service Packs" oder „Upgrades" aktualisiert, um die Fehler im Ausgangscode auszumerzen. Code funktioniert heute einfach nicht mehr. Wir haben die Komplexitätsgrenze für Software erreicht. Als Alternative bleibt uns entweder die Komplexität zurückzuschrauben oder neue Wege für das Programmieren zu finden.

Interessanterweise erreichen wir auch in anderen Bereichen bereits die Komplexitätsgrenze. So kämpfen z. B. so viele verschiedene Armeen mit dermaßen diversifizierten Waffen auf den Kriegsschauplätzen von heute, dass die Unterscheidung zwischen Freund und Feind im Gefecht fast unmöglich wird. Vor allem dann, wenn der Feind die Erkennungsmechanismen, die man zum Einsatz bringen möchte, sabotiert. In der modernen Kriegsführung fordert „freundliches Feuer" die meisten Opfer. Auch technische Meisterleistungen bleiben davon nicht verschont: Selbst Weltraumsatelliten sind vom Problem „zu vieler Einzelelemente" betroffen, die durchaus grundlegender Natur sein können, wie die Verwechslung von britischen und metrischen Maßeinheiten. Und als nächstes wird es das Human-Genom-Projekt treffen. Die exakte Dekodierung Tausender Gene übersteigt die heutigen Technologien, denn jedes einzelne Gen beeinflusst so viele andere Gene in millionenfacher Weise. Und es lässt sich nicht erkunden, was ein einzelnes Gen isoliert macht, denn isoliert funktioniert es nicht.

Das Schicksal des traditionellen Programmiercode mag besiegelt sein, doch was bleibt uns sonst? Womit sollen wir ihn ersetzen? Wir müssen uns an der Biologie orientieren, denn die Natur hat ihren eigenen Code. Das Leben zu studieren ist besonders attraktiv, denn die Systeme der Natur sind höchst fantasievoll, kreativ, effizient und elegant. Und darüber hinaus scheint die Natur nicht durch irgend welche Komplexitätsgrenzen eingeschränkt zu sein (zumindest keine, die wir mit unseren derzeitigen Mitteln wahrnehmen können).

Die Evolution ist ein Meister im Programmieren natürlicher Systeme. Gene sind der Code der Natur. Die Evolution nützt dabei genetische Neuerungen in Populationen von Individuen. Diejenigen, die aufgrund ihrer Gene ihrer Umwelt besser angepasst sind, überleben länger und vermehren sich. Bei der Reproduktion geben sie modifizierte Kopien ihrer Gene an ihre Nachkommen weiter und erzeugen so neue, bessere Varietäten. Die menschliche Kreativität stellt man sich oft so vor, dass bestehende Ideen neu zusammengefügt werden oder man eine neue Idee hat, die der alten einen Schritt voraus ist. Und genau so arbeitet die Evolution. Allerdings entwirft sich die Natur immer selbst. Niemand steuert die Evolution; sie ist ein sich selbst überlassener, zufälliger Prozess. Funktioniert etwas, wird es in Form von DNA gespeichert. Alles andere wird verworfen. Das ist eine geradezu brutal effiziente Prozedur zur Codegenerierung.

Auch die nächste Generation von Computercode wird auf diesen Prinzipien beruhen. Evolutionäre Algorithmen sind Computerprogramme, die Problemlösungen entwickeln. Gleichgültig, ob man dem Computer die Aufgabe „optimiere die Turbinenschaufel eines Düsentriebwerks" oder „komponiere ein Musikstück" stellt, er wird die Lösung dafür selbst finden. Diese neuartigen Computercodes (die u. a. Schwarmalgorithmen und Chaossysteme einsetzen) sind selbstorganisierende Systeme. Sie sind die Problemlöser und die Programmierer der Zukunft.

Beziehungsweise werden sie es sein, sobald wir herausgefunden haben, wie man sie wirklich zum Funktionieren bringt. Denn wieder stoßen wir an eine Komplexitätsgrenze: Die Natur ist für unseren Verstand einfach zu komplex. Und es ist schwierig, einen natürlichen Prozess im Computer nachzubilden, wenn man ihn nicht vollständig durchschaut. Während wir also erfolgreiche Problemlösungen mit unseren Computern entwickeln, haben wir diese Grenze noch nicht durchstoßen. Unsere evolutionären Algorithmen erreichen nämlich nur einen

bestimmten Komplexitätsgrad. Will man noch weiter, so muss man entweder die Evolution selbst steuern oder von der Natur noch eine letzte Sache lernen:
Natürlicher Code ist anders geartet als Computercode. Wir unterscheiden in unserer Technologie klar zwischen Hardware und Software. Die Hardware wird entworfen, dann produziert und manifestiert sich schließlich als ein Stück Silizium – ein Chip. Die Software ist da schon dynamischer: Man schreibt sie, ändert sie, lädt sie in den Chip oder löscht sie daraus. Die Natur kennt diese Unterscheidung nicht.
Das ist der springende Punkt. Natürliche Systeme bestehen nicht aus Hard- und Software, sondern vereinen beides. Der Code eines Lebewesens wurde von der Evolution entworfen. Aber dieser Code ist ein physisches Molekül, ein komplexes DNA-Molekül. Dieses Molekül ermöglicht die Produktion von Proteinen. Bei der Proteinproduktion durch die Gene interagieren die Eiweißbausteine mit den Genen und aktivieren bzw. deaktivieren sie. Der natürliche Code wird auf der Grundlage physikalischer Gesetze „ausgeführt" und veranlasst die Zellen sich zu teilen, zu wachsen, sich zu bewegen, zu verändern, Substanzen zu extrudieren und zu sterben. Ehe man sich versieht, ist ein vollständig ausgeformter, vielzelliger Organismus entstanden. Der enthält aber weder Software noch Hardware. Solange die Bedingungen nicht optimal sind, tut die DNA gar nichts – sie muss sich in der richtigen Zelle befinden, die Temperatur muss stimmen, und sie muss von passenden Proteinen umgeben sein. DNA ist nicht nur Information, sie ist *dinglich.* Sie funktioniert nur, weil sie physisch ist. Sie fußt auf zig anderen physikalischen Gegebenheiten und Gesetzen, um überhaupt eine Bedeutung zu haben. Daher ähnelt die DNA kaum unserem üblichen Computercode, der für jeden Computer Relevanz hat (in Arbeitsanweisungen übersetzt werden kann). DNA ist in ihre Umgebung eingebettet. Sie bildet einen integralen Bestandteil eines Organismus und nutzt die Feinheiten der sie umgebenden Zellen und Proteine. In einer anderen Umgebung verliert die DNA ihre Bedeutung. Verpflanzt man meine Gene in die Zelle einer Maus oder eines Schimpansen, so funktionieren sie nicht. Ein oder zwei werden vielleicht noch etwas bewirken, aber auf dem Rücken einer Maus wird kein Peter Bentley heranwachsen (sofern man den Mäuserücken nicht erfolgreich in eine menschliche Gebärmutter umgewandelt hat).
Die Natur lehrt uns also, diese willkürliche Unterscheidung zwischen Hard- und Software aufzugeben, wenn wir selbstorganisierende Prinzipien wie die Evolution nutzen wollen, um leistungsfähigen Code jenseits jeder Komplexitätsgrenze zu generieren. Code muss physisch werden (oder die Hardware in die Software integriert sein). Verwischt man die Grenze zwischen Code und Computer, so wird der Code millionenfach mächtiger. Stellen Sie sich ein Gerät vor, dessen Struktur denkt und das durch Veränderung seiner eigenen Struktur auch seine Denkweise verändert. Das klingt ausgesprochen organisch. Kann Kunst durch Form definiert werden und kann sie sich selbst neu definieren, indem sie die Form ändert? Mit diesen Konzepten haben wir so unsere Not, denn wir sind es einfach gewöhnt, das Wissen von den Dingen zu trennen so wie die Software von der Hardware. Wenn das Wissen und das Objekt eins werden, was haben wir dann geschaffen? Einen neuen Computertyp? Eine neue Kunstform? Ich weiß es nicht, aber ich würde es liebend gerne herausfinden.

Aus dem Englischen von Michael Kaufmann