Matthew Fuller

Hurra! Die endlose, vielschichtige, superschnelle, ungeheuer komplexe Langeweile ist da!

Warum hüpft Software? Welche Worte, welche Denkweisen sind nötig, um Programmcode und seine vielschichtigen Zusammensetzungen zu verstehen? Und wie – wenn überhaupt – setzt sich Software in Bezug zu dem, was man als Freiheit bezeichnen könnte? Die Frage, wie man sich im Umgang mit komplexen Technologien und in Live-Situationen verhalten, wie man sie verstehen soll, stellt sich nicht nur in der Technologie. Um eine Metapher zum besseren Verständnis zu finden, ist es oft hilfreich, am „falschen" Ende anzusetzen, d. h. in diesem Fall nicht bei der Software, sondern bei einem Frosch. In seinem Buch *Darwins gefährliche Erben* beschreibt der Biologe Steven Rose, wie fünf seiner Kollegen – ein Physiologe, ein Verhaltensbiologe, ein Developmentalist, ein Evolutionist und ein Molekularbiologe – am Ufer eines Teichs sitzen und miteinander darum wetteifern, den Sprung eines Frosches zu beschreiben. Dabei setzt jeder einzelne den spezifischen Wahrnehmungsmaßstab seiner eigenen Disziplin dem der anderen entgegen. Der Frosch seinerseits kümmert sich nicht um die schwatzenden Wissenschaftler, sondern vielmehr um eine Schlange, die er auf einem nahen Baum erspäht hat, und bringt sich mit einem eleganten Sprung in den Teich in Sicherheit. Die Vertreter der einzelnen Disziplinen führen nun einer nach dem anderen dieses „Sprung-Ereignis" auf folgende Faktoren zurück: die Interaktion von Nerven, Muskeln und Knochen, die strukturierte Energie- und Bewegungsmuster beinhalten und freisetzen; angelernte oder gewachsene Verhaltensreaktionen; das Ergebnis des spezifischen Wachstumsmusters des Organismus; die Aktion eines ererbten genetischen Imperativs; oder die biochemischen Eigenschaften der Muskeln.

So wie sich die durch das „Sprung-Ereignis" ausgelösten Wellen über den Teich ausbreiten und mit anderen Bewegungen des Wassers interagieren, argumentiert auch Rose für multivalente Denkweisen hinsichtlich einer nicht-reduktiven Biologie der Lebensmuster. Und mag es auch nur wenige Menschen auf der Welt geben, die mehr frisch geschlüpften Küken die Köpfe abgeschnitten haben als Rose in seinen Experimenten zur Physiologie der Erinnerung, so ist sein Appetit auf eine feuchte, komplexe, lebendige Biologie doch etwas, woran wir uns – bei aller nötigen Ironie – in unserem Softwareverständnis nur ein Beispiel nehmen können. Für die Biologie in ihrer Gesamtheit käme es nämlich laut Rose darauf an, sowohl den von den Einzeldisziplinen – die jeweils auf ganz bestimmte, konstitutive Wirklichkeitsmaßstäbe ausgerichtet sind, jene des Gens, des Moleküls, des Organismus usw. – demonstrierten Willen zum Detail als auch die radikalen Verflechtungen dieser Maßstäbe anzuerkennen.

Wenn wir in Bezug auf Software von Freiheit sprechen – auch wenn wir dabei ein paar Mal ausspucken müssen, um den inzwischen nachhaltig bitteren Geschmack dieses Worts loszuwerden, das uns nichtsdestotrotz noch immer das Wasser im Mund zusammenlaufen lässt und dem wir mit hechelnder Zunge hinterherhetzen –, so könnten wir vielleicht dafür ähnliche Maßstäbe geltend machen. Man stelle sich eine Gruppe von Menschen vor, die einen Computer betrachten. Einer ist der Meinung, die vom Computer ausgeführten Aktionen seien von der Hardware bestimmt, seine Möglichkeiten also durch die Mineralarchitektur der Datenverarbeitung abgesteckt. Ein anderer orientiert sich an der Sprachgeschichte und sieht Software durch die verschiedenen Arten der Syntaxkonstruktion bestimmt, durch die in den einzelnen Umgebungen verfügbaren logischen Strukturen. Der Dritte geht von einer Kritik der politischen Ökonomie von Software aus und argumentiert, die Möglichkeiten von Software würden durch die darin eingebetteten und in ihr zirkulierenden Vermögensverhältnisse konstituiert, wobei er vielleicht noch auf die Erkenntnisse der *Free Software*-Bewegung verweist. Die vierte

Person schließlich behauptet, man könne Software nur durch die Analyse der Benutzeroberfläche verstehen, durch die ethnografische Untersuchung der Signifikationsprozesse der Maschine und ihrer Verwendungsweisen. Was wir mit dem Computer tun, bestimmt die Qualität seiner Freiheit. Auch wenn diese vier Typen nicht in dieser „reinen" Form existieren, so symbolisieren sie doch bestehende Tendenzen in unserem Verständnis von Software sowie die Arbeitsteilung in der Softwareproduktion.

In einem klassischen Essay in der *Actor-Network*-Tradition – einer soziologischen Strömung, die sich auf die Interaktion der Elemente in soziotechnischen Gefügen konzentriert – beschreibt Madeleine Akrich in einem Exkurs verschiedene Möglichkeiten, wie man bei der Analyse eines Autos vorgehen könnte. Für eine solche Studie, argumentiert sie, gebe es einen natürlichen Maßstab: „Zweifellos könnte es befriedigend sein, alles auf einer großen Leinwand aufzumalen – angefangen von den Muttern und Bolzen, Kolben und Kurbelwellen, Stirnräder und Keilriemen weiter zu Wahlsystemen, den Strategien großer Industriekonzerne, zur Definition der Familie und zur Festkörperphysik … Was würde die Analyse stoppen – abgesehen von der willkürlichen Grenze der Ermattung? Ganz abgesehen davon, wie viel Zeit eine solche Studie erfordern würde, stellt sich auch die Frage, ob sie überhaupt interessant wäre."

Im Fall von Software wäre es vermutlich ein noch viel mühsameres Unterfangen, genau darzustellen, wie jedes Einzelteil eines solch komplexen technischen Gegenstands die Beziehungen zwischen „heterogenen Elementen" zugleich verkörpert und auslotet.

Müssen wir diesen Umweg über die Langeweile gehen? Womit müsste man da rechnen? Der 1973 im Lighthill-Bericht zur künstlichen Intelligenz geprägte Ausdruck „kombinatorische Explosion" ist inzwischen in unser alltägliches Verständnis von Datenverarbeitung eingeflossen. Man versteht darunter eine Situation, in der jedes Teilchen eines logischen Puzzles mit jedem anderen Teilchen verglichen werden muss. Mit steigender Anzahl der Puzzleteile steigt die Zahl der möglichen Kombinationen exponenziell an, und das Problem wird zu groß. Damit, so meinte man, sei dem damaligen Programm zur Entwicklung künstlicher Intelligenz eine natürliche Grenze gesetzt; der Bericht diente in der Folge dazu, den drastischen Rückgang der institutionellen Beteiligung an diesbezüglichen Forschungsarbeiten in Großbritannien zu legitimieren. Solche scheinbar natürlichen Grenzen des Verständnisses und der „Intelligenz" setzen Komplexitätsschwellen fest, die sich nur um den Preis beträchtlicher Schwierigkeiten oder lähmender Langeweile überwinden lassen. Laut Edsgar Dijkstra wird die Situation noch durch folgenden Sachverhalt verkompliziert: „Beim Computer-Programmieren braucht unser Grundbaustein, der Befehl, weniger als eine Mikrosekunde, doch unser Programm kann stundenlange Berechnungszeiten erfordern. Ich kenne keine andere Technik außer das Programmieren, von der erwartet wird, dass sie eine Granularität von 10^{10} oder mehr abdeckt. Der automatische Rechner war aufgrund seiner fantastischen Geschwindigkeit das erste Gerät, das eine Umgebung mit genügend „Raum" für hochgradig hierarchische Artefakte bot. Und in dieser Hinsicht ist die Herausforderung beim Programmieren etwas nie Dagewesenes."

Die vollständige Erfassung der Bestandteile von Software – egal nach welchem der von unseren vier Beobachtern vertretenen Maßstäbe – gestaltet sich also aufgrund des Zeitfaktors noch komplizierter. Rose betrachtet die Geschichte, die Bewegung eines Organismus in der Zeit, durch – und als – das Wechselspiel dieser verhältnismäßig selbstbestimmenden Schichten, sieht sie als Weg, die Biologie zu verstehen, ohne in die Falle des fetischisierten, letztendlich disfunktionalen Reduktivismus zu tappen. Ein Leben läuft nicht nach Bedingungen ab, die der jeweilige Organismus selbst erzeugt; vielmehr generiert jedes Leben seine eigene kombinatorische Explosion.

Wären wir so unklug, Akrichs zweifellos zutreffende Warnung in den Wind zu schlagen und

den Versuch zu unternehmen, jeden möglichen Bestandteil des katalytischen Gewebes einer bestimmten Software abzubilden, und zwar in jeder möglichen skalaren Ausprägung und sowohl in der Geschwindigkeit, die der Berechnungsrate der dazugehörigen Hardware (in jeder ihrer realen Konfigurationen) entspricht, als auch in jener, die sämtlichen sozialen, semiotischen und empirischen Kombinationen, denen dieser Bestandteil innewohnt, angemessenen ist – was käme dabei heraus? Man kann sich die endlosen Weiten feinster soziologischer Langeweile vorstellen, die man beim Studium von Software zu durchmessen hätte. Sollten wir weltweite, generationenübergreifende Forschungsinstitute gründen, um sämtliche Einzelschritte in einem unbedeutenden Telefonspiel oder sämtliche in *Lotus 1-2-3* durchgeführten Berechnungen oder das Sasser-Virus zu untersuchen? Vielleicht gibt es solche Untersuchungen bereits, doch bleiben ihre Ergebnisse bisher verblüffenderweise unkommunizierbar. Wonach also sollten wir Ausschau halten? Was für Instrumente verwenden? Wird es einen idealen Zeitpunkt geben, um die vier – oder mehr – Maßstäbe aufeinander abzustimmen, und wird sich zu einem solchen Zeitpunkt ein erkennbarer Riss oder Zusammenbruch manifestieren?

Vielleicht haben wir hier etwas vor uns, das der umfassenden Aufzeichnung sämtlicher Interaktionen bei der Autoherstellung gleichkäme. Komplexe Artefakte aus multiplen und parallelen Kombinationen mikrometerfeiner Teilchen, die sich mit einer gewissen Geschwindigkeit fortbewegen, können plötzlich – bei einer bestimmten Frequenz, bei einer bestimmten Kombination aus Rauschen, Ausrichtung, Wärme und anderen Faktoren – einen Zusammenbruch oder Schlimmeres auslösen. Ein Zylinder, der nach leicht fehlerhaften Spezifikationen erzeugt wurde, bricht bei hoher Belastung auseinander: Die Festkörperphysik synchronisiert sich abrupt mit der Geschichte des Autobenutzers. Im Softwarebereich kommt es bei Bugs, Abstürzen und Fehlern besonders jener Softwaresorten, die mit der Sorte von Benutzern interagieren, die gemeinhin als *Konsument* bekannt ist, zu ähnlichen Formen der Interaktion.

Multiskalare Synchronisation kann völlig unterschiedliche Ergebnisse zeitigen. Bei lebenden Systemen können Veränderungen des Aggregatzustands sowie der Häufigkeit und Intensität der Interaktionen zwischen den Bestandteilen den Übergang in einen anderen Stoffwechselzustand, Veränderungen in der Population (Artenbildung oder Aussterben) oder die Spezialisierung von Zellen bewirken. Unter Bedingungen, die als politisch gelten, und sowohl im sozialen als auch im zwischenmenschlichen Maßstab kann die Kombination einzelner Elemente solch eindeutige Siedepunkte erreichen. Wesentlich ist auch die Art und Weise, wie diese Elemente aufeinander eingehen, wie sie im Wettbewerb oder durch Zusammenhalt versuchen, die zwischen ihnen bestehenden Zusammenhänge beizubehalten, zu fixieren, zu verschieben, zu verändern oder aufzubrechen, ohne diese Schwellen zu überschreiten.

Ein Beispiel, bei dem natürlich das allzu heterogene Auto eine Rolle spielt, wäre der Imperativ, sich eine Rückwirkung des ökologischen Zustands unseres Planeten auf die wirtschaftlichen und technischen Ausprägungen der ihn derzeit dominierenden Spezies vorzustellen. Zur Beschreibung der Änderungen, die unsere aktuellen Umweltbedingungen erfordern würden, bräuchten wir ein besseres praktisches Vokabular als das der Reform oder der Revolution; derartiges Feedback sollte man sich nicht nur vorstellen, sondern es auch tatsächlich herstellen und danach handeln. Software scheint dafür ein interessanter Ausgangspunkt zu sein. Nicht nur, weil in diesem Bereich die Tätigkeit des Modellierens, Kombinierens und Zahlenknackens eingesetzt werden kann (und schon seit jeher eingesetzt wird), um Zeit in die richtige Richtung zu verstärken, um virtuelle Zukunftsvariationen abzuspielen, bevor sie überhaupt passieren, so dass wir auf ihre Auswirkungen reagieren können – nein, auch deshalb, weil Software einen Teil jenes Bereichs darstellt, der den Kapitalismus und seine bevorzugten, besitzbaren Energiequellen unter Umständen durch Heterogenisieren überwinden kann.

Bereist man die immensen multiskalaren Felder der Ermattung, die Akrich den Kraftlosen verheißt, so stellt man fest, dass sich zwischen den Maßstäben hie und da Falten bilden – eine Form des Wissens erfordert die andere, um selbst Sinn zu ergeben. Mitten in den unermesslichen Weiten der Leere und Wiederholung stößt man unter Umständen auf seltsame Assoziationsklümpchen, Teilchen, die sich ineinander verbeißen; mitten in der unendlichen sanften Abbildung unzählbar ähnlicher Teile erstarren Tausende Teilchen gleichzeitig, stoßen zusammen oder verschlingen einander gierig, erzeugen sich zu neuem Terrain hin öffnende Risse und neue Gedankengebilde und erzwingen neue Skalen und Konjugationen des Verstehens.

Für Software gibt es keinen „natürlichen Maßstab". Alle vier der genannten Metaphern – und unausweichlich mehr als vier – müssen zusammenwirken, an sich selbst und gemeinsam mit den anderen arbeiten, das eigene Vorkommen in der lebendigen Geschichte anerkennen. Die Herausforderungen, vor die uns ein derartiges Softwareverständnis stellt, sind mannigfaltig; es genügt nicht, einfach verschiedene Schichten miteinander zu verbinden. Bei bestimmten metabolischen Netzwerken verbinden sich bis dahin getrennte Elemente so stark miteinander, dass sie kristallisieren: Es entsteht effektiv eine neue Verbindung. Multidisziplinäre Arbeitsmodelle, die diese Größenordnungen miteinander verbinden und Hardware-Designer, Programmierer, Anwälte, Ökonomen, Benutzeroberflächen-Designer, Soziologen und Anhänger der freien Marktwirtschaft vernetzen, existieren zweifellos – man nennt sie Großunternehmen. Doch sogar innerhalb bzw. zwischen derartigen Entitäten ist es schwierig, Wissensstile und Arbeitsformen monolithisch zu gestalten. Die skalare Solidarität der in der *Free/Libre* und *Open Source Software*-Bewegung engagierten Programmierer hat uns eine Reihe von Möglichkeiten gezeigt, wie man Software und Freiheit in Bezug zueinander setzen kann. Zugleich schärft sie unseren Blick für andere Verwendungen dieses Worts mit dem bitteren Beigeschmack: Bei seiner Suche nach absturzsicheren Tötungsmaschinen hat sich nämlich das US-Militär zum Teil die Linux-Entwicklung einverleibt. Doch wer oder was wäre besser geeignet, wird argumentiert, den Kommunismus der Ideen zu finanzieren? Die vier oben skizzierten Softwaremaßstäbe legen nahe, dass Freiheit auch auf anderen Ebenen erzeugt werden muss.

Wie können wir auf unserer Reise durch die endlosen Weiten der Langeweile erkennen, ob wir am Rand einer besonders interessanten Faltenlinie oder Verstärkung angelangt sind? Wie können wir eine potenzielle Synchronisation der Teile innerhalb von Größenordnungen erahnen, die eine potenzielle Freiheitsmetapher abgeben könnten? Mit welchen Instrumenten können wir entschlüsseln, ob wir knapp vor einem Augenblick stehen, in dem alles, in jedem erdenklichen Maßstab, kristallisiert und ganz widerlich klar wird, oder – wie in den Diskus-

sionen über die Militarisierung von Linux – ob die Vorherrschaft vielleicht gerade überstrapaziert, zu ihrem eigenen Anderen wird, um rasch genug gegen den Lauf der Geschichte anzurennen, um sicherzustellen, dass alles beim Alten bleibt? Eine ähnliche Frage könnte man in Zusammenhang mit den Schritten stellen, die unternommen werden, um für diesen Zweck ungeeignete Patentgesetze auf Software anzuwenden. Gibt es einen Algorithmus, mit dem man die potenziellen Interaktionen innerhalb dieser vier Softwaremaßstäbe errechnen könnte? Ein solcher Algorithmus müsste sich erstens durch ein Paradoxon konstituieren und eine rekursive Zerlegung und Rekonstituierung des Freiheitsbegriffs selbst beinhalten sowie in gleichem Maß die Vorstellung transzendentaler Regeln mit einem Lachen abtun. Seine ironischen Faustregeln könnte man in Anlehnung an „Tu, was du willst", Rabelais' minimalen Regelsatz für die Utopie von Théleme, oder an den praktischen Leitsatz „Verbieten verboten" der Freidenker formulieren. Natürlich gibt es hier noch zahllose weitere Beispiele, doch Maximen allein sind nicht genug. Sowohl Freiheit als auch Software erfordern eine bestimmte Leichtigkeit, gepaart mit der hartnäckigen Erkenntnis, dass es in jedem Maßstab möglich ist, Dinge geschehen zu lassen und Sprünge zu vollführen. Solche Algorithmen wären in der Lage, innerhalb der vier oben angeführten Maßstäbe zu funktionieren, sowie zwischen ihnen und über sie hinaus; sie würden ihre konstituierenden Elemente in bisher unvorstellbare Konjunktionen mit Elementen versetzen, die andere Realitäten durchqueren und so eine rasche Kondensation der Datenverarbeitung bewirken. Auf sich selbst nicht reduzierbar und somit unmöglich als reine Software konfigurierbar, kann man diese affirmativen und konvulsiven Elemente und Konjunktionen vielleicht dabei ertappen, wie sie geschäftigem Treiben nachgehen oder in der eigenwilligen Engmaschigkeit zwischen einzelnen Realitätsschichten auf der Lauer liegen. Funktionierende Beziehungen von Kapazitäten und Energien im Zeitverlauf, das strukturelle und konstitutive Wechselspiel der Einschränkungen, Erfordernisse und Erfindungen von Software – all das bedeutet, dass die Freiheitsmetaphern, die wir brauchen werden, um durch die Langeweile zu navigieren, sowohl verfügt als auch kodiert sein müssen.

Aus dem Englischen von Susanne Steinacher

Akrich, Madeleine, „The De-scription of Technical Objects", in: Bijker, Wiebe und Law, John (Hrsg.): *Shaping Technology Building Society,* Cambridge, Mass., 1992
Dijkstra, Edsger Wybe, *A Discipline of Programming,* Prentice-Hall, 1976
Lighthill, James, „Artificial Intelligence: A General Survey", in *Artificial Intelligence: a paper symposium,* Science Research Council, 1973
Rose, Steven, *Darwins gefährliche Erben: Biologie jenseits der egoistischen Gene,* München 2000