Der Begriff des Algorithmus

Wenn wir uns heute, etwas mehr als 60 Jahre nach der Erfindung des Computers, dem Begriff der Programmierung nähern, ist es zweckmäßig, dies in der gleichen Reihenfolge zu tun, in der die verschiedenen Programmierungskonzepte entdeckt bzw. erfunden wurden. Anfänglich war die Programmierung ausschließlich imperativ, und ein Programm war die computerverständliche Formulierung eines Algorithmus. Von imperativer Programmierung sprechen wir, wenn das Programm eine Folge von Anweisungen ist. Ein Algorithmus ist eine Vorschrift, welche Schritte ein Ausführender nacheinander oder teilweise nebenläufig ausführen muss, damit ein gewünschtes Ziel erreicht wird. Als Beispiel denke man an den Algorithmus zur Multiplikation mehrstelliger Dezimalzahlen. Diesen Algorithmus lernt man irgendwann in den frühen Jahren der Schulzeit. Bei der Formulierung eines Algorithmus muss immer vorausgesetzt werden, dass es einen Ausführenden gibt, der bestimmte Schritte ausführen kann, wobei es irrelevant ist, wie er dies tut. Wenn wir also beispielsweise einen Ausführenden hätten, für den die Durchführung der Multiplikation zweier zehnstelliger Zahlen ein einziger Schritt ist, dann besteht der Algorithmus nur aus diesem Schritt. Wenn dagegen die Multiplikation von einem Ausführenden gemacht werden soll, der nur Schritte ausführen kann, bei denen einzelne Ziffern verknüpft werden, dann muss zur Multiplikation ein mehrschrittiger Algorithmus ausgeführt werden. In Abhängigkeit von der Mächtigkeit der Schritte, die der zur Verfügung stehende Ausführende bewältigen kann, muss der Algorithmus zur Erreichung eines bestimmten Ziels mehr oder weniger kompliziert oder trickreich sein.

Es zeigt sich immer wieder, dass das Erfinden von Algorithmen als sehr reizvolle Aufgabe erlebt wird, wobei man manchmal den Eindruck hat, dass sich dies zur Sucht entwickelt. Sie sollten immer daran denken, dass Ihre Aufgabe als Ingenieur der Softwaresystemtechnik nicht vorrangig darin besteht, Algorithmen zu erfinden. In großen Systemen gibt es meistens Tausende von Algorithmen, und der Ingenieur ist dafür zuständig, das Zusammenwirken der Komponenten zu planen und die arbeitsteilige Realisierung zu koordinieren. Man denke hier an die Analogie zum Maschinenbau: Der einzelne Algorithmus entspricht den elementaren Komponenten wie Zahnrädern oder Kugellagern, wogegen das System ein ganzes Auto oder ein Kraftwerk ist, worin es Tausende von Komponenten gibt, die in einer sinnvollen Struktur zusammen gebracht werden müssen. Um als Erfinder von Algorithmen zu arbeiten, braucht man kein Ingenieurstudium.

Petrinetze zur Algorithmendarstellung

Einen Algorithmus darzustellen bedeutet, die gewünschte Schrittfolge darzustellen. Wenn man nicht gleich die Forderung erhebt, dass der Algorithmus in computerverständlicher Form dargestellt sein soll, d.h. wenn man zuerst einmal danach fragt, wie der Algorithmus in menschengeeigneter Darstellung aussehen soll, dann findet man die graphische Form als besonders geeignet. Hier hat sich die Form der sogenannten Petrinetze als besonders kommunikationsfreundlich erwiesen. Ein Petrinetz ist ein bipartiter Graph, der auf eine bestimmte Weise interpretiert wird.

Ein bipartiter Graph ist eine graphische Struktur, in der es zwei symbolisch unterscheidbare Knotensorten gibt, wobei Verbindungen nur zwischen Knoten unterschiedlicher Sorte vorkommen. Wir betrachten das Beispiel in Bild 5. Eigentlich müssten in den Rechtecken, die im Falle der Petrinetze Transitionen genannt werden, verständliche Anweisungen stehen. Da ich aber hier kein konkretes Programm betrachten will, schreibe ich anstelle konkreter Anwei-

sungen einfach die Buchstaben A, B und C in die Transitionen. Die Kreisknoten werden Stellen oder Plätze genannt. Damit man immer weiß, wie weit man mit der Ausführung des Algorithmus schon gekommen ist, führt man eine Marke ein, die je nach Ausführungsstand auf einem der 3 Plätze liegt. Als Anfangsmarkierung ist die Situation gezeigt, dass die Marke auf dem Platz S1 liegt. Im ersten Schritt der Algorithmenausführung muss also die Anweisung A ausgeführt werden; anschließend liegt die Marke auf dem Platz S2. An dieser Stelle nun muss eine Entscheidung gefällt werden, nämlich die Entscheidung, ob als nächstes die Anweisung B oder die Anweisung C ausgeführt werden soll. Diese Entscheidung kann nicht willkürlich gefällt werden, sondern muss anhand einer Überprüfung des Wahrheitswertes der angeschriebenen Bedingungen entschieden werden. Wenn die Bedingung c erfüllt ist, gehen wir nach unten, d.h. wir führen die Anweisung C aus; andernfalls führen wir die Anweisung B aus. Am Ende der Ausführung der Anweisung B liegt die Marke wieder auf dem Platz S2, wo sie vor der Ausführung von B schon lag. Deswegen muss erneut die Bedingung überprüft werden, denn es könnte sein, dass die Anweisung B noch einmal ausgeführt werden muss. Diese möglicherweise wiederholte Ausführung der Anweisung B wird deutlich sichtbar durch die Schleife im Petrinetz.

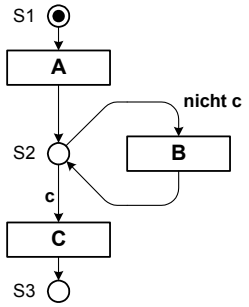

Bild 5 Beispiel einer Algorithmendarstellung in Form eines Petrinetzes

Aktionen, Akteure und Aktionsfelder

Damit man sich vorstellen kann, was bei der Ausführung eines Programms geschieht, muss man nicht nur das den Ablauf veranschaulichende Petrinetz betrachten, sondern man muss auch die Welt vor Augen haben, in der der Programmablauf geschieht. Denn jede Anweisung verlangt eine Operation auf bestimmten Aktionsfeldern. Wenn beispielsweise verlangt wird, dass die Fenster geputzt werden sollen, dann muss es diese Fenster geben, d.h. die Fenster sind die Aktionsfelder für die Ausführung der Anweisung. Neben den Aktionsfeldern muss es jemanden geben, der die Anweisung ausführt, d.h. es muss neben den Aktionsfeldern auch noch Akteure geben. Die Struktur aus Akteuren und Aktionsfeldern bildet den Systemaufbau, also die Welt, worin die Ausführung des Programms möglich ist.

Die Voraussetzung für das Verständnis eines Programms ist immer die Kenntnis der Welt, in der die Programmabwicklung geschieht. Leider wird diese Welt üblicherweise nicht dargestellt. Für uns aber ist die Darstellung dieser Welt, d.h. die Darstellung der zum Programmablauf gehörenden Aufbaustruktur, eine zwingende Selbstverständlichkeit. Auch Aufbaupläne sind bipartite Graphen wie die Petrinetze, nur dass die beiden Knotensorten nun eine völlig andere Interpretation erfahren. Bei den Petrinetzen sind die beiden Knotensorten die Transitionen und die Stellen, wogegen im Aufbau die beiden Knotensorten die Akteure und die Aktionsfelder sind. Das Bild 6 zeigt als Beispiel drei Systemkomponenten, welche die Welt bilden, in der ein Schachspiel ablaufen kann: Es gibt zwei Akteure, den Spieler links und den

Spieler rechts, sowie ein Aktionsfeld, nämlich das Schachbrett mit den darauf stehenden Figuren zwischen den beiden Spielern.

Bild 6 Aufbaustruktur der „Schachspiel-Welt“

Während der Aufbau, worin sich das Schachspiel abspielt, eine natürliche Sichtbarkeit hat und fotografiert werden kann, sind die Welten, in denen das durch die Programme beschriebene Geschehen abläuft, weitgehend gedachte Welten, die nicht fotografierbar sind. Dennoch ist es sinnvoll, auch diese Welten stets als Strukturen aus Akteuren und Aktionsfeldern zu denken.

Dass wir durchaus auch im Alltag außerhalb unseres Umganges mit Computern bereits solche Aufbauwelten denken, erkennt man, indem man sich vorstellt, was sich im Kopf eines telefonierenden Menschen abspielt. Man stelle sich vor, dass sich zuerst die Telefonzentrale meldet, nachdem der Anrufende gewählt hat. Zuerst spricht also der Anrufer mit jemandem in der Zentrale, der ihn anschließend an jemanden anderen weitervermittelt. Im Laufe des Gesprächs kann er noch häufig den Gesprächspartner wechseln, und je nachdem, welchen Gesprächspartner er gerade am anderen Ende hat, wird er sich mit seiner Sprache anpassen müssen, weil nicht unbedingt jeder Gesprächspartner die gleiche Sprache spricht. Was der Telefonierende wirklich sieht, ist nur sein Telefon und sein Telefonhörer, aber was er vor seinem geistigen Auge sieht, ist eine Struktur, in der es mehrere Akteure gibt, die nacheinander mit ihm in Verbindung treten können. Er hat also eine Struktur vor Augen, wie sie das Bild 7 zeigt.

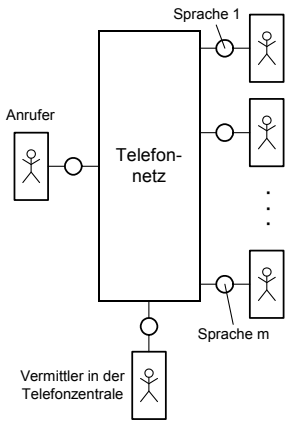

Bild 7 Aufbaustruktur zur Abwicklung eines Telefongesprächs

Der telefonierende Mensch wird nie auf die Idee kommen, er kommuniziere mit seinem Telefon. Das Telefon ist nur Mittel zum Zweck, um mit den zwar gehörten aber nicht gesehenen Partnern zu kommunizieren. In gleicher Weise sollte man nicht denken, man kommuniziere mit dem Computer, wenn man an einem Computerarbeitsplatz sitzt. Nur über einen angemessenen Aufbau, den man vor Augen haben sollte, wenn man an einem Computerarbeitsplatz kommuniziert, wird das System und das Geschehen plausibel.

Wer kein Bedürfnis nach einer Aufbaustruktur hat, die er mit seinem geistigen Auge sehen will, stellt sich auf die Stufe von Kindern, die mit dem Computer kommunizieren, ohne eine Struktur im System sehen zu wollen. Aufgrund ihrer extremen Merkfähigkeit können sie ihren Umgang mit dem Computer ausschließlich phänomenologisch beherrschen, d.h. sie können sich lange Aktions- und Reaktionsketten merken, ohne nach irgendwelchen Begründungen zu fragen. Ingenieure der Softwaresystemtechnik sollten sich aber nicht auf diese Stufe stellen.

Wenn wir nun annehmen, wir müssten das im Bild 5 dargestellte Programm als Text weitergeben, beispielsweise in einem Telefonat, wo wir keine Bilder übermitteln können, dann können wir das sehr leicht tun: Wir werden sagen, dass es in dem Petrinetz die Stellen S1, S2 und S3 sowie die Transitionen A, B und C gibt. Wir werden sagen, dass in der Anfangsmarkierung die Marke auf der Stelle S1 liegt. Und schließlich werden wir mitteilen, welche Knotenverbindungen es gibt. Dazu brauchen wir immer nur zu sagen, von welchem Knoten zu welchem anderen Knoten ein Pfeil führt, beispielsweise von S1 nach A. Wenn an einem Pfeil eine Bedingung steht, werden wir dies auch mitteilen. Auf diese Weise erfährt der Empfänger die logische Struktur des Netzes, aber er erfährt nichts über das Layout. Es könnte sein, dass der Empfänger sich wieder ein Petrinetz aufzeichnet, welches dann aber völlig anders aussieht als das ursprüngliche Netz – beispielsweise wie in Bild 8.

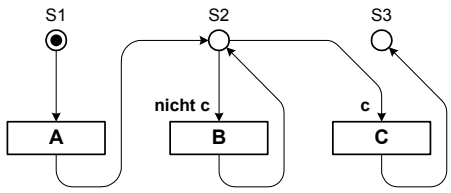

Bild 8 Petrinetz aus Bild 5, in alternativem Layout

Die Textform ist aber für den Computer sehr gut geeignet, denn der Computer interessiert sich sowieso nicht für das Layout, welches nur für die menschliche Anschauung hilfreich ist.

Der Aufbau des Prozedurabwicklers

Nun betrachten wir einen Aufbauplan, den man vor Augen haben sollte, wenn man an die Ausführung eines imperativ formulierten Algorithmus denkt.

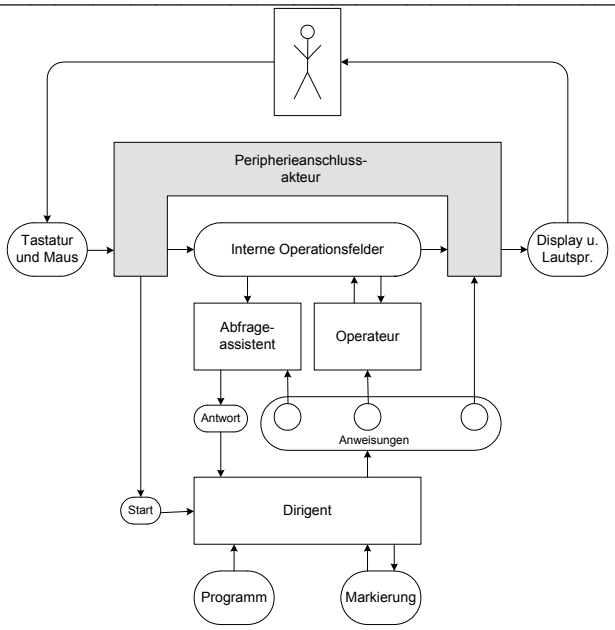

Bild 9 Aufbaustruktur des Systems, worin imperative Programme abgewickelt werden können

In diesem Aufbau gibt es oben den Menschen, der über Tastatur, Maus, Bildschirm und Lautsprecher mit dem programmausführenden System kommuniziert. Der schattiert dargestellte sogenannte Peripherieanschlussakteur ist derjenige, der auf den Bildschirm schreibt und der die vom Menschen produzierten Vorgänge auf der Tastatur und der Maus zur Kenntnis nimmt.

An dieser Stelle muss auf den Unterschied zwischen materiell-energetischer Relevanz und rein informationeller Relevanz von Erscheinungen in Systemen eingegangen werden. Es hängt von der Interessenslage des Systemnutzers ab, ob die Vorgänge auf bestimmten Aktionsfeldern nur informationell oder auch materiell-energetisch relevant sind. Wenn wir eine Kaffeemaschine betrachten, dann ist das Aktionsfeld, auf dem der Kaffee ausgegeben wird, materiell-energetisch relevant, d.h. wir wollen wirklich den realen Kaffee haben und nicht nur einen Zettel mit der Aufschrift *Hier ist der gewünschte Kaffe mit Milch und Zucker; passen Sie auf, dass Sie sich nicht den Mund verbrennen.*
Systeme, in denen es materiell-energetisch relevante Aktionsfelder gibt, können keine universell programmierbaren Systeme sein. Denn universelle Programmierbarkeit liegt nur vor, wenn man einen beliebigen formulierten Willen einem Universalausführer geben kann, der diesen Willen ausführt. Man stelle sich vor, man formuliere einmal den Willen, dass das mitgebrachte Bier gekühlt werden solle, und ein andermal formuliere man den Willen, dass man vom ersten in den fünften Stock gebracht werden solle. Im einen Fall müsste sich das System als Kühlschrank und im anderen Fall als Fahrstuhl verhalten. Solche Universalsysteme lassen sich selbstverständlich nicht bauen.

Die Kanäle zwischen dem Benutzer und dem Computersystem sind in jedem Falle materiell-energetisch relevant, d.h. sie sind nicht beliebig programmierbar, sondern müssen per Konstruktion ganz bestimmte Kommunikationsfähigkeiten des Menschen berücksichtigen. Wenn das System keinen Lautsprecher hat, kann es nicht akustisch mit dem Menschen verkehren. Wenn es keinen Drucker mit roter Farbe hat, kann es keine roten Zeichnungen machen. Bei

der Gestaltung der Tastatur muss die Fingergröße eines Durchschnittsmenschen berücksichtigt werden usw. Nur innerhalb des Systems kommen rein informationelle Aktionsfelder vor. Dort steht es dem Konstrukteur völlig frei, wie er die an diesen Aktionsfeldern vorkommenden Informationen darstellen will. Er kann beliebig miniaturisieren und er kann beliebig kodieren.

Ganz andere periphere Verhältnisse liegen beim sogenannten Prozessrechnen vor, d.h. bei der Rechnersteuerung technischer Anlagen. In Bild 10 ist ein System gezeigt, bei dem es um die Steuerung eines Rangierbahnhofs geht. Der Rechner selbst ist eine verschwindend kleine Komponente im Verhältnis zur Peripherie. Die Bedienerkonsole, also Tastatur, Bildschirm und Maus bilden in diesem Fall nur einen sehr kleinen Bruchteil der Peripherie. Der Hauptteil des Systems besteht aus den peripheren Anschlussgeräten zur Steuerung aller Einrichtungen, die zum Betrieb des Rangierbahnhofs gesteuert werden müssen: Weichen, Signale, Gleisbremsen, Schublokomotiven und ähnliches.

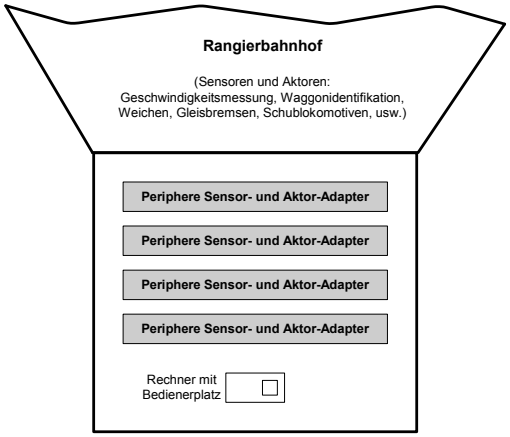

Bild 10 Veranschaulichung der Größenverhältnisse von Rechner zu Peripherie im Falle einer komplexen Prozess-Steuerung

Der im Bild 9 gezeigte Aufbau gehört nicht zu einem ganz bestimmten imperativen Programm, sondern jedes imperative Programm passt zu diesem Aufbau. Im Grunde stellt dieser Aufbau die Struktur dar, die den Erfindern des Computers vor Augen stand.
Neben dem Menschen und dem Peripherieakteur enthält dieser Aufbau drei weitere Akteure. Die Erkenntnis, dass man diese Akteure braucht, kann man sehr leicht selbst finden, wenn man sich fragt, wie man denn einen in Form eines Petrinetzes gegebenen Algorithmus selbst ausführen würde. Eine Idee, die sehr zweckmäßig ist, auf die man aber möglicherweise nicht von alleine kommt, ist die Trennung von Programm und Markierung. Das Programm ist das unmarkierte Petrinetz. Dieses muss man sich an dem mit *Programm* bezeichneten Ort im Aufbau vorstellen. Auf dem mit *Markierung* bezeichneten Aktionsfeld liegt in irgendeiner Formulierung die Information, welche Stelle des Petrinetzes aktuell markiert ist. Wenn die Anfangsmarkierung vorliegt, muss also am Ort *Markierung* die Information S1 liegen.

Es ist eine sehr zweckmäßige und sich immer wieder bewährende Entscheidung, den Akteur, der das Programm lesen kann und der die Markierung verändern kann, nicht auch die Anweisungen selbst ausführen zu lassen. Er kennt nur den Algorithmus und weiß, wie weit die Programmausführung schon fortgeschritten ist. Darüber hinaus kennt er einen oder mehrere Ak-

teure, von denen er die jeweilige Ausführung einer Anweisung verlangen kann. Diese Akteure wiederum können zwar einzelne Anweisungen ausführen, aber sie wissen nichts über die Reihenfolge der Anweisungen, die im Algorithmus formuliert sind. Sie sind darauf angewiesen, dass ihnen der Dirigent jeweils mitteilt, dass sie nun eine Anweisung ausführen sollen. Im Bild 9 sind drei unterschiedliche Akteure zur Ausführung von Anweisungen gezeigt: Es gibt Anweisungen, welche nur der Peripherieakteur ausführen kann, weil diese Anweisungen auf die peripheren Aktionsfelder Bezug nehmen. Dann gibt es Operationsanweisungen, welche Operationen auf dem internen Aktionsfeld verlangen. Und schließlich gibt es Anweisungen, Fragen bezüglich des aktuellen Inhalts des inneren Aktionsfeldes zu beantworten. Diese Antworten braucht der Dirigent, um Verzweigungen im Ablauf entscheiden zu können.