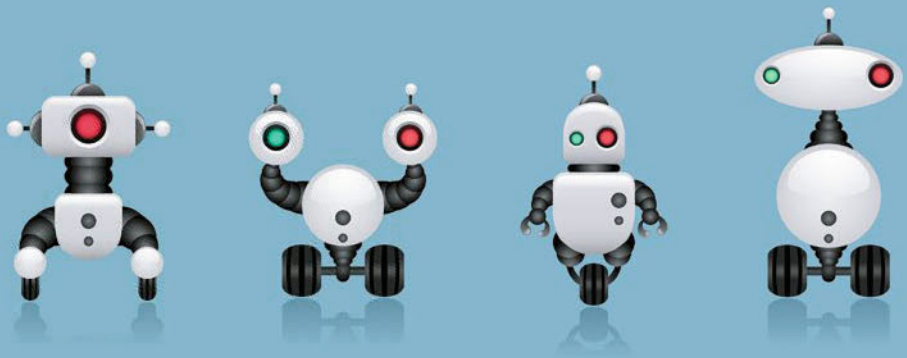
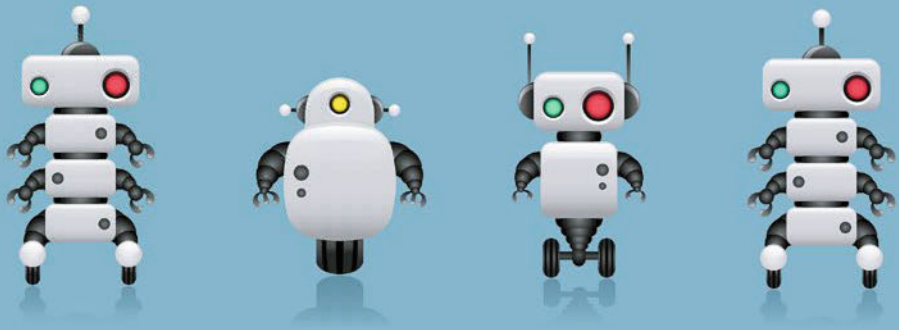
Paul Curzon · Peter W. McOwan

Computational Thinking



Die Welt des algorithmischen Denkens – in Spielen, Zaubertricks und Rätseln



EBOOK INSIDE

Springer

Computational Thinking

Paul Curzon · Peter W. McOwan

Computational Thinking

Die Welt des algorithmischen Denkens – in Spielen, Zaubertricks und Rätseln

Aus dem Englischen übersetzt von Bernhard Gerl

Springer

Paul Curzon
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, Großbritannien

Peter W. McOwan
School of Electronic Engineering and Computer Science
Queen Mary University of London
London, Großbritannien

Übersetzt von Bernhard Gerl

ISBN 978-3-662-56773-9 ISBN 978-3-662-56774-6 (eBook)
https://doi.org/10.1007/978-3-662-56774-6

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Übersetzung der englischen Ausgabe: The Power of Computational Thinking von Paul Curzon und Peter W. McOwan, © World Scientific Publishing Europe Ltd. 2017. Alle Rechte vorbehalten

Einbandabbildung: © capon/stock.adobe.com
Verantwortlich im Verlag: Margit Maly

Gedruckt auf säurefreiem und chlorfrei gebleichtem Papier

Springer ist ein Imprint der eingetragenen Gesellschaft Springer-Verlag GmbH, DE und ist ein Teil von Springer Nature
Die Anschrift der Gesellschaft ist: Heidelberger Platz 3, 14197 Berlin, Germany

Geleitwort

„Informatik hat ungefähr so viel mit Computern zu tun wie Astronomie mit Teleskopen." Dieses Zitat wird dem Turing-Preisträger Edsger W. Dijkstra zugeschrieben und beschreibt recht gut, warum *Computational Thinking* so wichtig ist: Die Informatik hat mit dem Klischee zu kämpfen, dass die Maschine das Untersuchungsobjekt ist. Im englischen Sprachraum wird das durch den Begriff „computer science", also Computerwissenschaft, auch noch verstärkt.

Computer sind aber von Menschen gestaltete Systeme! Insofern kann es kaum darum gehen, sie zu analysieren wie ein Naturphänomen. Vielmehr sind der Mensch und seine kreative Gestaltungskraft in den Mittelpunkt zu stellen. Nur auf diese Weise ist unsere zunehmend von menschlicher Gestaltung geprägte Lebenswelt zu verstehen.

Die aus diesen Überlegungen hervorgehenden Denkwerkzeuge bereichern nicht nur die Informatik, sondern sind in vielen Disziplinen sinnvoll anwendbar. Vor allem Jeanette Wing hat seit 2006 dafür den Begriff *Computational Thinking* geprägt und kämpft darum, dass informatisches Denken einen festen Platz in den Curricula der Schulen erhält. *Computational Thinking* ist heute fester Bestandteil fast aller angelsächsischer Lehrpläne und in den USA, Großbritannien, Australien und Neuseeland stark verbreitet.

Zunehmend spielt *Computational Thinking* auch eine Rolle in unserem allgemeinbildenden Schulsystem: im Sachunterricht der Primarstufe ebenso wie im Rahmen des Informatikunterrichts der Sekundarstufen in unterschiedlichen Bundesländern. Viele der Kompetenzen von Computational

Thinking entsprechen denen der Bildungsstandards Informatik oder ergänzen diese.

Ein weiteres Indiz dafür, dass die Kompetenzen von *Computational Thinking* mehr mit Menschen als mit Computer zu tun haben, ist ihr Alter von teilweise über tausend Jahren. So wurden auch schon früher Verfahren in festen Abfolgen einfacher Basisoperationen beschrieben, den Algorithmen. Der Euklidische Algorithmus geht ungefähr auf das Jahr 300 v. Chr. zurück. Sicher ist er somit keine Erfindung der Informatik. Die Anforderungen an die Formulierung – etwa in puncto Eindeutigkeit und Verständlichkeit – haben sich seither immer weiterentwickelt, sodass das Verfahren von Computern heute zwar immer noch mehr oder weniger im Sinne von Euklid eingesetzt wird, aber in einer komplett anderen Notation. Diese hilft nicht nur Computern, sondern auch Schülerinnen und Schülern, die das Verfahren verstehen und nachvollziehen möchten.

Tauchen Sie daher ein in die Welt des *Computational Thinking* und entwickeln Sie anhand der vielen anschaulichen Anwendungsbeispiele dieses Buchs selber Ideen, in welcher Hinsicht es Ihnen etwa bei der Lösung konkreter Aufgabenstellungen, für das bessere Verständnis von Erkenntnissen oder bei der eindeutigeren Formulierung Ihrer Ergebnisse helfen kann. Dabei wird gleichzeitig „kulturelle Kohärenz" hergestellt, die Brücke zwischen verschiedenen (Fach-)Kulturen geschlagen.

Prof. Dr.-Ing. Jens Gallenbacher
Leiter der Didaktik der Informatik an der
Technischen Universität Darmstadt

Vorwort

In nur wenigen Jahrzehnten hat das „Computational Thinking“, also das analytische, von Algorithmen geprägte Denken unser Leben, Arbeiten und Spielen von Grund auf verändert. Sogar Wissenschaft wird heute anders betrieben. Es hat geholfen, Kriege zu gewinnen, vollkommen neue Industrien entstehen lassen und viele Leben gerettet. Computerwissenschaftler haben eine ganz eigene leistungsfähige Art und Weise entwickelt, wie sie an die Lösung von Problemen, z. B. bei der Programmierung, herangehen. Die Methode ist aber auch unabhängig davon einsetzbar und so wichtig geworden, dass in vielen Ländern inzwischen gefordert wird, schon Grundschülern damit vertraut zu machen.

In diesem Buch erklären wir das algorithmische Denken auf leicht verständliche Weise. Wir werden Zaubertricks, Spiele und Rätsel genauso vorstellen wie anspruchsvolle Probleme des realen Lebensalltags, an denen Computerwissenschaftler arbeiten. Wir werden Grundbausteine bei dieser Art der Problemlösung behandeln wie Denken mit Algorithmen, Zerlegung, Abstraktion, Verallgemeinerung, logische Schlussfolgerungen und Mustererkennung. Dabei wird sich auch zeigen, dass es für erfolgreiches Computational Thinking entscheidend ist, den Menschen zu verstehen. Wir werden zudem die Verbindungen zwischen Computational Thinking und dem wissenschaftlichen und kreativen Denken – und wie daraus Innovationen entstehen – erkunden.

Ob Sie nun einfach neugierig sind, um, was es beim Computational Thinking geht, ob Sie neue Möglichkeiten finden wollen, selbst effektiver zu werden, ob Sie im Bereich der Computerwissenschaften zu arbeiten beginnen oder aber einfach Spaß an Spielen und Rätseln haben: Dieses Buch ist

genau für Sie geschrieben. Sie werden dadurch auf jeden Fall einen Vorsprung erlangen, ob nun beim Programmieren-Lernen oder ganz allgemein der Entwicklung neuer Technologien, aber auch bei der Lösung persönlicher Probleme im echten Leben. Sie werden sowohl für Ihr eigenes Gehirn als auch für die digitale Welt ein tieferes Verständnis entwickeln und sogar erfahren, wie man ein digitales Gehirn entwickeln kann.

Wir hoffen, dass unser Buch Ihnen zeigen wird: Zu denken wie ein Computerwissenschaftler ist faszinierend und macht großen Spaß. Das gilt natürlich unabhängig davon, ob Sie eine Frau oder ein Mann sind, doch aus Gründen der besseren Lesbarkeit verwenden wir in diesem Buch überwiegend das generische Maskulinum. Dies impliziert immer beide Formen, schließt also die weibliche Form mit ein.

Paul Curzon
Peter W. McOwan

Danksagung

Dieses Buch enthält neues Material und überarbeitete Artikel, die für unsere Website *Computer Science for Fun* (www.cs4fn.org), die zugehörigen Zeitschriften und unsere Seite *Teaching London Computing* (www.teaching-londoncomputing.org), mit der wir EDV-Lehrer unterstützen, entstanden sind.

Wir sind der Queen Mary University in London sehr dankbar dafür, dass man uns dort immer bei unserem Engagement für die allgemeine Öffentlichkeit unterstützt hat. Unsere Arbeit an der Entwicklung von Informatikmaterialien, die Spaß machen, wurde im Laufe der Jahre von vielen verschiedenen Organisationen unterstützt, so neben der Queen Mary University in London von dem EPSRC, von Google, dem Bürgermeister von London, dem Erziehungsministerium, der BCS, RCUK, Microsoft und ARM.

Lehrer im ganzen Land und darüber hinaus haben uns immer wieder in unserer Arbeit bestärkt. Vor allem die Pädagogen, Akademiker, Industrievertreter und andere Mitglieder der Gruppe *UK Computing at School* haben unsere Arbeit unterstützt, immer wieder die ausgezeichnete Ideen geliefert und viele unserer Angebote ausprobiert. Auch den zahlreichen Studenten und Lehrkräften sind wir dankbar, die in den letzten zehn Jahren unsere Aktivitäten mit Enthusiasmus begleitet haben und bereit waren, sich darauf einzulassen und neue Ideen beizutragen. Simon Peyton-Jones von Microsoft Research, Peter Dickman von Google und Bill Mitchell von der British Computer Society haben uns sehr intensiv unterstützt. Auch Tim Bell und Quintin Cutts sowie den Teams, die an unseren Aktivitäten ohne PC an den Universitäten Canterbury und Glasgow beteiligt waren,

verdanken wir neben vielem anderem eine Unzahl von Ideen. Die *Knights Tour*-Veranstaltung wurde insbesondere durch eine Idee von Maciej M. Syslo und Anna Beata Kwiatkowska von der Nicolaus Copernicus University inspiriert.

Auf vielfältige Weise wurden wir auch von Mitarbeitern der Queen Mary University in London unterstützt, darunter Ursula Martin, Edmund Robinson und Sue White, die uns geholfen haben, die Website cs4fn zum Laufen zu bringen. Gabriella Kazai und Jonathan Black waren hoch engagierte Mitarbeiter der ersten Stunde, William Marsh, Jo Brodie, Nicola Plant, Jane Waite und Trevor Bragg waren in letzter Zeit beteiligt. Und noch viele weitere haben uns unterstützt.

Als Teenager wurden wir vor allem von Martin Gardners Unterhaltungsmathematikbüchern inspiriert, auch wenn wir erst später erkannten, dass das meiste der wirklich interessanten Dinge in Wirklichkeit Informatik war, die sich als Mathematik tarnte. Wir hoffen, dass dieses Buch in ähnlicher Weise rasch erkennen lässt, dass Informatik das wirklich Unterhaltsame ist. Lassen Sie sich nicht davon beirren, wenn manche es Mathematik nennen.

Wir haben mit Matt Parker, Jason Davison und Richard Garriott viele nützliche Diskussionen über Mathematik und Zauberei geführt. Dabei möchten wir auch den genialen Magiern der Vergangenheit und Gegenwart danken, die sich jene intelligenten mathematischen Tricks ausgedacht haben, die Computeralgorithmen nutzen, mit denen wir heute spielen und unterrichten können. Namentlich seien folgende Zauberer genannt, die uns inspiriert haben: Alex Elmsley, Karl Fulves, Nick Trost, J. K. Hartman, Paul Gordon, Brent Morris, Colm Mulcahy, Arthur Benjamin, Max Maven, Aldo Colombini, Persi Diaconus, John Bannon und natürlich der große, 2010 verstorbene Martin Gardner. Wir empfehlen Ihnen, sich deren Werke einmal anzusehen und so mehr über die von selbst ablaufenden Zaubertricks zu erfahren. Sie werden einige erstaunliche Techniken, Algorithmen und Unterhaltungsmöglichkeiten entdecken und vielleicht hinter das größte Geheimnis überhaupt kommen ... nämlich dass Zauberei (abgesehen von Informatik natürlich) ein großartiges Hobby ist.

Wir wären niemals zu dem Spiel gekommen, das wir heute betreiben, ohne die zahlreichen inspirierenden Lehrer, die uns nicht nur für Mathematik und Wissenschaften begeisterten, sondern – genauso wichtig – für Englisch und uns damit halfen, das Schreiben zu verstehen und zu lieben.

Am meisten Dank gilt unseren Familien und ihrer unglaublichen Unterstützung und Geduld.

Inhaltsverzeichnis

Über die Autoren

Paul Curzon ist Informatikprofessor an der Queen Mary University in London. Seine Forschungen beschäftigen sich mit der Informatikausbildung, mit Mensch-Maschine-Kommunikation und formalen Methoden. Er erhielt neben mehreren Preisen für gute Lehre 2010 die National Teaching Fellowship der Higher Education Academy und wurde 2007 von der EPSRC als nichtprofessioneller Informatikautor ausgezeichnet. Er war an der Gründung von *Teaching London Computing* (www.teachinglondon-computing.org) beteiligt und leitet Weiterbildungsmaßnahmen für Lehrer. Paul brachte sich seine ersten Programmierkenntnisse an einem Strand in Südfrankreich bei.

Peter W. McOwan ist Informatikprofessor an der Queen Mary University in London. Seine Forschungsschwerpunkte liegen in den Bereichen Computervision, künstliche Intelligenz und Robotik. Er erhielt 2008 die National Teaching Fellowship der Higher Education Academy und 2011 die IET-Mountbatten-Medaille für seine Bemühungen, die Informatik für verschiedenen Zielgruppen aufzubereiten. Peter ist Amateurzauberer mit einem gesunden Interesse an Science-Fiction.

Paul und Peter schufen gemeinsam das international bekannte Projekt *Computer Science for fun* (www.cs4fn.org) und waren Gründungsmitglieder des britischen Netzwerks *Computing At School* (CAS). Heute ist Paul im Vorstand des CAS.

1

Die Zukunft des Denkens

Computational Thinking ist eine zentrale Fähigkeit, die Informatiker erlernen und dann verwenden, um Probleme zu lösen. Sie ist derart wichtig, dass in vielen Ländern erwogen wird, bereits Schülerinnen und Schüler darin zu unterrichten. Doch worum handelt es sich dabei eigentlich? Wie hat dieses besondere Denken alles verändert, was wir tun? Und warum ist es die Grundlage für so viele unterhaltsame Dinge?

Was wollen Sie tun?

Möglicherweise sind Sie ein Wissenschaftler und versuchen, das Verhalten von Vögeln zu untersuchen, die am Boden fressen. Und Sie beobachten, wie die einen nach Futter suchen, während andere den Himmel wegen der Raubvögel im Blick behalten. Doch wie entscheiden die Tiere jeweils, wer gerade was macht? Auch andere Wissenschaftler verbringen ihre Zeit damit, die Vögel zu beobachten, doch Sie gehen einen anderen Weg: Denn Sie denken über den Algorithmus – also die Abfolge von Schritten – nach, an die sich die Vögel halten müssen, wenn sie beschließen, sich so oder anders zu verhalten. Dazu entwerfen Sie ein Computermodell und simulieren verschiedene Szenarien auf der hypothetischen Grundlage, dass jeder Vogel seinen Nachbarn beobachtet. Das Modell bildet Ihre Beobachtungen über das Verhalten der Vögel ab und erstellt darüber hinaus Vorhersagen, die Sie dann wiederum in freier Wildbahn überprüfen können.

Vielleicht sind Sie ein Zauberkünstler. Sie haben eine Idee für einen neuen Trick, der auf einer mathematischen Eigenschaft der Zahlen beruht. Sie entwickeln die einzelnen Schritte und die Darbietung, aber wird der

P. Curzon und P. W. McOwan, *Computational Thinking,*
https://doi.org/10.1007/978-3-662-56774-6_1

Trick wirklich immer funktionieren? Statt ihn nur auszuprobieren, können Sie logisch darüber nachdenken und beweisen, dass er immer funktioniert. Es sei denn, Sie finden heraus, dass dem nicht so ist und es eine Situation gibt, in der er schiefgehen könnte. Indem Sie dann die Darbietung ein klein wenig ändern, können Sie sicherstellen, dass diese Situation nie auftreten wird.

Sie sind Schüler, und der Lehrer erklärt Ihnen, wie das Gehirn funktioniert. Er hat das Bild eines Neurons an die Tafel gemalt und dessen einzelnen Teile beschriftet, die Sie lernen sollen. Über Nacht schreiben Sie ein Programm, das wie ein Neuron funktioniert. Wenn Sie mehrere davon verbinden, sehen Sie, wie eine Gruppe von Neuronen tatsächlich Aufgaben erledigen kann. Sie verwenden dieses Programm, um am nächsten Tag diese Funktionsweise Ihren Mitschülern zu erklären.

Oder vielleicht sind Sie ein Arzt, der frustriert ist, weil Ihre medizinischen Mitarbeiter ein Gerät fehlerhaft bedienen. Die Verwaltung macht den Mitarbeitern Vorwürfe und eine Krankenschwester wird aufgrund ihres Fehlers einfach gefeuert. Sie erkennen, dass das Problem im Design des Gerätes begründet ist: Es begünstigt, dass die Mitarbeiter einen Schritt überspringen können. Sie sprechen mit dem Hersteller darüber, wie eine geringfügige Veränderung im Design dafür sorgen könnte, dass dieses Problem nie wieder auftritt.

Oder sind Sie Lehrer mit einem gigantischen Stapel an ungeordneten Klausuren. Die müssen noch sortiert werden, um die richtige Klausur am Elternabend schnell finden und darüber sprechen zu können. Doch das ist kein Problem, denn Sie kennen eine wirklich schnelle Methode, wie sich die Papiere ordnen lassen.

Vielleicht haben Sie einen Ferienjob in einem Imbiss. Ihnen fällt auf, dass sich immer lange Schlangen bilden und die Kunden gereizt sind. Sie weisen Ihren Vorgesetzten darauf hin, dass dies teilweise daran liegt, dass die Person an der Kasse viel Zeit mit Nichtstun verbringt. Mit einer kleinen Veränderung in der Art und Weise, wie das Team zusammenarbeitet, könnte alles beschleunigt werden.

Vielleicht ist Ihr Kopf voller Ideen für Spiele, die Sie und Ihre Freunde gerne spielen möchten. Doch im Gegensatz zu anderen reden Sie nicht nur über Ihre großartigen Ideen, sondern schreiben dazu kurzerhand Programme … und fangen nach wenigen Tagen an zu spielen.

In der Informatik geht es nicht nur um Computer, sondern die Arbeitsweise von Computern kann überall zur Anwendung kommen. Denken Sie wie ein Computerwissenschaftler und es werden sich zahlreiche

Situationen ergeben, in denen Sie etwas verbessern können – und überall gibt es Gelegenheiten, um Ideen Wirklichkeit werden zu lassen.

Kompetenzen für das 21. Jahrhundert

Wer Informatik studiert, lernt auch eine vollkommen neue Art zu denken und Probleme zu lösen. Und diese Denkweise ist entscheidend in unserer modernen Welt voller Technologie. Dieses sogenannte Computational Thinking[1] bringt Ihnen viele Vorteile, wo auch immer Sie später arbeiten werden … und die Idee dahinter hat bereits international für Furore gesorgt. In vielen Ländern wird sie für so wichtig gehalten, dass man sie zu einer Kernkompetenz erklärt hat, die sich schon Grundschüler neben Lesen, Schreiben und Rechnen aneignen sollten. Es handelt sich um eine Art zu denken, die bereits dazu führte, dass Computer große Teile unseres Lebens übernommen und dadurch alles verändert haben, was wir tun: vom Musikhören über den Handel mit Aktien und Wertpapieren und der Art und Weise, wie wir einkaufen, bis zum Betreiben von Wissenschaft. Es verleiht uns die Fähigkeit, nicht nur großartige Ideen zu haben, sondern diese auch in die Wirklichkeit umzusetzen.

Der Erziehungswissenschaftler und Mathematiker Seymour Papert (1928–2016) war es, der den Begriff „Computational Thinking" als Erster verwendet hat. Er vertrat die Ansicht, dass Mathematik ganz anders unterrichtet werden solle, nämlich so, wie Computer arbeiten. Doch nicht nur die Mathematik hat sich durch die Informatik verändert, sondern alle Wissenschaften. Die Computerwissenschaftlerin Jeannette Wing (*1956) glaubt, dass diese Denkweise zu lernen das Wichtigste im Informatikstudium ist, sich aber auch weit darüber hinaus als nützlich erweist. So war es diese Informatikprofessorin, die dem Begriff zu Popularität verhalf. Microsoft zeigte sich von ihren Argumenten und der Wichtigkeit des Themas so überzeugt, dass das Unternehmen ihrer Universität, der Carnegie Mellon University in Pittsburgh, Pennsylvania, Drittmittel von mehreren Millionen Dollar bewilligte, damit dort ein Zentrum entstehen konnte, in dem dieser Aspekt der Informatik sowie die Art, wie er andere Wissenschaften verändert, untersucht werden konnte.

[1] Anm. des Übersetzers: Für „Computational Thinking" gibt es keinen deutschen Terminus. In diesem Buch wird deshalb der englische Ausdruck verwendet.

Was also ist Computational Thinking? Es bedeutet nicht, dass wir anfangen sollen, wie Computer zu „denken", obwohl wir diese immer öfter so programmieren, dass sie selbst Computational Thinking verwenden. Es handelt sich um eine Reihe von Kompetenzen, die sich Menschen angeeignet haben, um Probleme zu lösen, und die aus Untersuchungen darüber hervorgegangen sind, wie Computer arbeiten. Computational Thinking beinhaltet so mache offensichtlich wichtige Qualifikation, die für die Entwicklung vieler Dinge entscheidend ist wie Kreativität oder die Fähigkeiten, etwas erklären zu können und im Team zu arbeiten. Es führt zu Denkweisen, die auf anderen Gebieten geprägt wurden, etwa mathematisch und wissenschaftlich zu folgern. Im Kern handelt sich es sich jedoch um einige sehr spezifische Problemlösefähigkeiten wie jener, logisch und in Algorithmen zu denken mit einem Fokus auf dem letzten Detail, oder jener, sich neue und effiziente Wege zu überlegen, wie etwas getan werden kann. Und schließlich geht es auch darum, Menschen zu verstehen. Es ist einzigartig, wie die Computerwissenschaft diese unterschiedlichen Fähigkeiten zusammenbringt. Vereint ergeben sie eine leistungsfähige Art des Denkens, das die Welt verändert. Und das beeinflusst, wie wir Wissenschaft betreiben, einkaufen, Unternehmen führen, Musik hören, Spiele spielen, ja eigentlich betrifft es alle Aspekte unseres Lebens.

Algorithmisches Denken

Das Denken in Algorithmen ist ein Herzstück des Computational Thinking. Dazu gehört, dass man an die Lösung von Problemen vollkommen anders herangeht. Für einen Informatiker ist die Lösung eines Problems nicht nur eine Antwort wie „42". Es ist auch keine Leistung wie: „Ich habe gerade das Sudoku von heute gelöst." Lösungen sind Algorithmen. Ein *Algorithmus* ist eine Reihe von Anweisungen, denen man folgen muss. Wenn man diesen Anweisungen im Algorithmus genau folgt, dann sollte man tatsächlich eine Antwort auf das Problem finden (wie 42) oder das erreichen, was man beabsichtigt hat (wie ein Sudoku zu lösen). Wenn erst einmal eine algorithmische Lösung vorliegt, findet man die Antwort für eine Problemstellung, indem man den Anweisungen einfach „blind" folgt. Mit einem Algorithmus, der ein Problem löst, kann jeder dieses Problem lösen, ohne nachzudenken. Er muss den Algorithmus nicht einmal kennen oder verstehen, was dieser letztendlich macht. Vielleicht weiß er nicht einmal, dass er tatsächlich ein Sudoku löst (oder sogar, was das ist). Daraus folgt auch, dass selbst eine „dumme" Maschine – ein Computer – jede Variation

des Problems lösen kann, indem sie einfach mechanisch den Anweisungen folgt. Denn das ist alles, was Computer tun: Sie folgen Algorithmen, die von Menschen geschrieben wurden.

Was diese Idee so leistungsfähig macht, ist, dass die Durchführung eines Algorithmus Lösungen für eine ganze Gruppe von Problemen ergibt. Ein Algorithmus, der Kreuzworträtsel löst, könnte viele Kreuzworträtsel lösen. Ein Algorithmus zum Rechnen sollte zu jeder Rechnung in der Lage sein. Wenn man in dieser Weise über Probleme und Lösungen nachdenkt, spricht man von *algorithmischem Denken*.

Es reicht zum Beispiel nicht, dass man weiß, dass $20 + 22 = 42$ ist. Ein Informatiker will einen Algorithmus, der alle Zahlen addieren kann. Eigentlich lernt jeder in der Grundschule einen Algorithmus dafür, deshalb kann jeder Schüler rechnen, ohne sich diesen selbst neu überlegen zu müssen. Genauso haben alle Computer die Instruktionen eingebaut, wie man addiert, weil das so wichtig ist. Computer können nur deshalb als Taschenrechner verwendet werden, weil sie Anweisungen kennen, die ihnen sagen, wie sie Rechnungen durchführen sollen. Ein Computerprogramm ist also ein Algorithmus, der in einer Sprache geschrieben wurde, die ein Computer versteht, einer *Programmiersprache.*

Verändern Sie die Welt!

Doch es geht nicht nur um Berechnungen, Algorithmen können für alles Mögliche verwendet werden. Wer in Algorithmen denkt, kann damit die Welt verändern. Wenn Sie einen Algorithmus als Programm aufschreiben, können Sie ihn dazu bringen, blind alle möglichen Dinge zu tun. Banken verwenden Algorithmen statt Menschen, um mit Aktien zu handeln, um sie möglichst schnell zu kaufen oder zu verkaufen und damit Millionen Profit machen. Die NASA setzt sie ein, um Raumschiffe zum Mars zu schicken. Sie selbst nutzen Algorithmen, um Musik oder Videos abzuspielen. Algorithmen steuern Flugzeuge, helfen Chirurgen beim Operieren und uns beim Einkaufen, während wir im Wohnzimmer oder in einem Zug sitzen. Sie sind sogar inzwischen in der Lage, ein Auto zu steuern. Selbst Kunstwerke und Musik können sie erzeugen. Heutzutage spielen sie in allen Aspekten unseres Lebens eine Rolle. Algorithmen haben bereits unsere Art zu leben verändert und werden das noch stärker tun. Deshalb ist es für jeden wichtig, das Denken in Algorithmen zu verstehen. Genau wie wir Physik und Biologie lernen, um die stoffliche Welt und das Leben um uns herum

zu verstehen, müssen wir etwas Informatik kennen, um die virtuelle Welt zu begreifen, die in aller Stille unser Leben übernommen hat.

Wissenschaftliches Denken

Doch das Denken in Algorithmen ist mehr als eine Möglichkeit, Probleme zu lösen. Es bietet eine neue Methode, die Welt zu verstehen. Die traditionelle Wissenschaft basiert auf Experimenten: Biologen experimentieren mit Ratten, Affen oder Zellkulturen, Mediziner führen Arzneimitteltests durch, und Physiker führen Versuche in der realen Welt durch. Doch wenn sie in Algorithmen denken können, gibt es eine Alternative. Angenommen Sie haben eine Theorie darüber, wie etwas funktioniert, sei es nun bezüglich der Einflüsse von Strahlung auf einen Planeten, wie ein Ökosystem funktioniert oder wie Krebs Zellen angreift, dann können Sie Algorithmen entwerfen, die genauso arbeiten. Sie können ein *Computermodell* schaffen: ein Programm, das dazu geschaffen ist, das Phänomen, das Sie interessiert, zu simulieren. Dann können Sie Ihre Experimente an dem Modell durchführen statt in der realen Welt. Wenn Ihr Ansatz richtig ist, wird sich das Programm so verhalten wie die Sache, die es modellieren soll. Anderenfalls stimmt mit der Theorie etwas nicht. Indem Sie darüber nachdenken, was schiefgelaufen ist, können Sie Wege finden, die Theorie zu verändern und so Ihren Ansatz verbessern. Das Modell kann darüber hinaus auch neue Vorhersagen liefern, die sich dann draußen in der echten Welt überprüfen lassen.

Computational Thinking

Also ist Computational Thinking weit mehr als nur Lösungen in Form von Algorithmen zu finden. Es handelt sich vielmehr um eine ganze Reihe von Techniken, die uns leistungsfähige Werkzeuge liefern, mit denen sich Dinge verbessern lassen und tiefer über die Welt nachdenken lässt. Doch statt nun weiter über Begriffe und Definitionen zu sinnieren, werden wir das Computational Thinking nun ganz praktisch mithilfe von Beispielen einführen – und zwar sowohl mit ernsten (etwa wie man Menschen mit Behinderungen helfen kann) als auch mit unterhaltsamen (etwa mit Spielen, Rätseln und Zaubertricks).

2

Sprechen aus der Taucherglocke

Eine der schlimmsten Krankheiten, die man sich vorstellen kann, ist das Locked-in-Syndrom. Man ist vollständig gelähmt und kann höchstens mit den Augen blinzeln. Der wache Verstand ist in einem nutzlosen Körper eingesperrt, der zwar in der Lage ist, alles wahrzunehmen, aber nicht kommunizieren kann. Das kann jedem ganz unerwartet als Folge eines Schlaganfalls passieren. Wenn man Menschen mit dem Locked-in-Syndrom unterstützen will, muss man Krankenschwester oder Arzt werden. Oder kann auch ein Computerwissenschaftler helfen?

Locked-in-Syndrom

Beim Locked-in-Syndrom ist man infolge eines Schlaganfalls vollständig gelähmt. Man kann weiterhin denken, sehen, hören und ist so intelligent wie zuvor. Das kann jedem passieren und es gibt keine Heilungschance. Deshalb kann die Medizin wenig mehr tun, als es dem Patienten so angenehm wie möglich zu machen. Sich anderen mitzuteilen, ist eines der größten Probleme, vor denen Menschen mit Locked-in-Syndrom stehen. Wie können sie mit ihren Ärzten, ihrer Familie und Freunden kommunizieren? Und wie kann man ihnen dabei helfen? Was ein Computerwissenschaftler in diesem Fall versuchen würde, ist offensichtlich: Er könnte eine neue Technologie entwickeln, die bei der Kommunikation hilft. Doch mit ein wenig Computational Thinking kann man zu einer wesentlichen besseren Problemlösung gelangen, als technische Lösungen sie bieten.

P. Curzon und P. W. McOwan, *Computational Thinking,*
https://doi.org/10.1007/978-3-662-56774-6_2

Schmetterling und Taucherglocke ist ein unglaublich ergreifendes Buch. Es handelt sich um die Autobiografie von Jean-Dominique Bauby, die er schrieb, nachdem er vollständig gelähmt in einem Krankenbett aufgewacht war. Darin beschreibt er sein Leben mit dem Locked-in-Syndrom. Er hatte also eine Möglichkeit zu kommunizieren gefunden – nicht nur mit Ärzten, Freunden und seiner Familie, sondern er schrieb sogar ein ganzes Buch, und zwar ganz ohne Technologie. Aber wie?

Versetzen Sie sich in seine Lage. Sie wachen in einem Krankenhausbett auf. Wie könnten Sie nun kommunizieren? Wie ein Buch schreiben? Sie haben nur eine Person, die Ihnen mit Stift und Papier gegenübersitzt, um Ihre „Wörter" aufzuschreiben. Sie können sich zwar nicht bewegen und sprechen, aber sehen und hören … und sind einer der wenigen Glücklichen mit Locked-in-Syndrom, die immerhin mit den Augen blinzeln können.

Und nun stellen Sie sich vor, Sie sind der Arzt dieser Person und müssen einen Weg finden, um mit ihr zu kommunizieren.

Einfach wie das Abc

Was Sie brauchen, ist eine Möglichkeit, das Blinzeln des Patienten (das ist alles, was er kann) in Buchstaben umzuwandeln. Vielleicht sagen Sie ihm, dass einmal Blinzeln „A" bedeutet, zweimal „B" usw. Der Helfer muss dann nur noch zählen, wie oft Sie geblinzelt haben, um die entsprechende Nachricht zu schreiben.

Mit dieser Methode denken wir schon wie ein Computerwissenschaftler. Und was wir dabei tun, ist das Herzstück des Computational Thinking: *algorithmisches Denken*. Wir sind auf eine Reihe von Schritten gekommen, mit deren Hilfe wir und der Helfer sicherstellen können, dass die Buchstaben, die der Patient denkt, übermittelt werden. Ein Computerwissenschaftler nennt diese abgesprochene Methode der Kommunikation einen *Algorithmus*, eine Reihe von Schritten in einer vorgegebenen Reihenfolge, der man folgen muss und mit der man ein Ziel erreicht (hier das Ziel, Buchstaben und Wörter zu kommunizieren). Durch algorithmisches Denken erschafft man Algorithmen, um Probleme zu lösen.

Das Schöne an Algorithmen ist, dass die einzelnen Schritte von den Beteiligten abgearbeitet werden können, ohne dass diese verstehen müssen, was sie da eigentlich tun. Bei unserem Algorithmus wüssten die Helfer vermutlich, was und warum sie es machten, doch das Buch hätte auch sonst geschrieben werden können. Denn es muss nichts weiter getan werden als zu zählen, wie oft geblinzelt wird, und entsprechend den Anweisungen, die zuvor

gegeben wurden, die Buchstaben zu notieren. Es ließe sich sogar eine Tabelle zur Verfügung stellen, in der die Buchstaben nachgeschlagen werden können, sodass gar nicht mehr nachgedacht werden muss. Das Schöne an Algorithmen ist also, dass Menschen Dinge „mechanisch" ausführen können – und das ist das Entscheidende, denn es bedeutet, dass auch Computer solchen Anweisungen blind folgen können.

Unser Algorithmus für die Kommunikation besteht eigentlich aus zwei Teilen: einem, dem Bauby folgen muss (in der richtigen Anzahl blinzeln), und einem für den Helfer (zählen, wie oft geblinzelt wurde, und die Anzahl in einen Buchstaben übersetzen, der notiert wird). Computerwissenschaftler haben sogar einen eigenen Namen für solche Algorithmen, die Informationen zwischen zwei Personen oder Computern weiterleiten – *Protokoll*. Wenn beide Seiten ihrem Teil des Protokolls folgen, enden die Wörter, die Bauby denkt, als geschriebene Buchstaben auf Papier. Wenn einer von beiden einen Fehler macht – sich zum Beispiel verzählt und damit nicht dem Protokoll folgt –, kommt die Nachricht nicht durch. Das Großartige an Computern ist, dass ihnen derartige Fehler nicht passieren: Sie folgen ihren Anweisungen immer ganz genau. Und wenn die Anweisungen richtig sind, dann wird der Computer diese auch richtig befolgen.

Denken in Algorithmen ist eine spezielle Methode der Problemlösung – eine, bei der man am Ende nicht nur eine einzige Lösung erhält – wie im Beispiel das, was Bauby sagen will, wenn er zum ersten Mal aufwacht. Sondern man findet eine Lösung in Form von Schritten, denen andere (inklusive Computer) folgen können, um Antworten zu erhalten. Also eine Lösung wie im Fall Baubys, durch die wir nicht nur erfahren, was er gerade jetzt sagen möchte, sondern mit der wir (und jeder andere) herausfinden können, was er zukünftig sagen wird. Zugegebenermaßen erscheint diese Lösung ziemlich langsam. Und vielleicht gibt es noch eine bessere Möglichkeit. Über bessere und effizientere Lösungen nachzudenken, ist ebenso Teil des algorithmischen Denkens.

Baubys Methode

Bauby fand eine bessere Möglichkeit, einen effektiveren Algorithmus … und den beschreibt er in seinem Buch. Vergessen Sie nicht, dass der Helfer sprechen kann, sodass wir das nutzen können. Bei Baubys Algorithmus hat der Helfer das Alphabet laut vorgelesen: „A … B … C …". Wenn der Buchstabe, an den er gedacht hatte, ausgesprochen wurde, blinzelte er.

Der Helfer schrieb den Buchstaben auf und begann das Alphabet wieder von vorne. Probieren Sie das mit einem Freund aus. Kommunizieren Sie zuerst einmal Ihre Initialen auf diese Weise. Dann stellen Sie sich vor, dass dies Ihre einzige Möglichkeit ist, mit irgendjemanden zu kommunizieren (Ich hoffe, Ihr Name ist nicht Zacharias Zog oder Yvette Zoltan …).

Stellen Sie sich jetzt einmal vor, dass Sie immer auf diese Weise mit Ihren Freunden oder Ihrer Familie reden müssen, und auch mit Ärzten und Krankenschwestern. Dies ist Ihre einzige Möglichkeit, um die kleinste Kleinigkeit des Alltags zu regeln: etwa wenn Sie möchten, dass jemand die Vorhänge öffnet oder das Fernsehprogramm wechselt.

Sobald Sie es einmal ausprobiert haben, werden Sie feststellen, dass sich einige weitere Probleme ergeben, die gelöst werden müssen, bevor es wirklich funktioniert. Nach ein paar Versuchen werden Ihnen vielleicht selbst einige Möglichkeiten einfallen, wie man den Algorithmus verbessern könnte. Was könnten Sie machen?

Prüfen der Details

Womöglich ist Ihnen schon aufgefallen, dass wir zur schnelleren Kommunikation mehr als die 26 Buchstaben brauchen – nämlich auch Zwischenräume, Ziffern, Punkte, Kommata usw. Diese müssen wir also mit auf die Liste für den Helfer setzen. Vielleicht gibt es auch eine bessere Möglichkeit als eine lange Liste. Vielleicht sollte die erste Frage sein: „Ist es ein Buchstabe?“ Wenn ja, machen wir weiter wie vorher, und wenn nicht, können wir die anderen Symbole abarbeiten. Kommt Ihnen das bekannt vor? Es ist die Art und Weise, wie Textverarbeitungssysteme mit unterschiedlichen Zeichensätzen umgehen.

Was wir ebenfalls noch lösen müssen, ist die Frage, was passiert, wenn die Person irrtümlich blinzelt. Wir benötigen eine Möglichkeit, mit der wir zum Ausdruck bringen können, dass das letzte Blinzeln ignoriert und die Buchstaben noch einmal abgefragt werden sollen. Sie würden das sicher nicht Buchstabe für Buchstabe aussprechen wollen. Dementsprechend brauchen wir, wenn wir gerade einen Fehler gemacht haben, eine Möglichkeit zurückzugehen. Dazu wird ein Code benötigt, der „Rückgängig machen“ bedeutet. Eine Methode, Dinge rückgängig zu machen, ist ein wichtiger Teil jedes Algorithmus, bei dem Menschen beteiligt sind, denn Menschen machen Fehler. In unserem Fall könnten wir uns darauf einigen, dass zweimaliges schnelles Blinzeln diese Bedeutung hat. Vielleicht fällt Ihnen aber auch etwas Besseres ein. Und vielleicht haben Sie auch noch andere

Probleme bemerkt, die ebenfalls gelöst werden müssen. Die Überprüfung eines Algorithmus funktioniert sowohl in der Theorie als auch in der Praxis so. Diese *Evaluation* ist ein wichtiger Bestandteil des Computational Thinking.

Immer wenn wir uns einen neuen Algorithmus ausdenken, müssen wir sehr sogfältig prüfen, ob er auch wirklich funktioniert. Programmierer verbringen mehr Zeit damit, ihre Programme (die ja nichts weiter als Algorithmen für Computer sind) zu überprüfen, als damit, diese zu schreiben. Es passiert sehr schnell, im Detail einen Fehler zu machen oder eine Situation zu vergessen, die gelegentlich auftreten könnte, sodass der Algorithmus auch damit umgehen können muss. Das Entscheidende an einem Algorithmus ist, dass er jedes Mal zuverlässig funktionieren muss, ganz egal, was auch passiert.

Denken in Algorithmen bedeutet, an alle Details zu denken und Lösungen für potenziell auftretende Probleme zu finden. Es geht darum zu erkennen, dass es viele Möglichkeiten geben kann, etwas zu tun, und dann die beste davon in einer bestimmten Situation auszuwählen. Und vergessen Sie nicht, dass es bei einem der Probleme, über die wir uns oben bereits Gedanken gemacht haben, darum ging, dass Fehler zu machen menschlich ist. Theoretisch funktioniert unsere Lösung: Blinzle einfach zum richtigen Zeitpunkt. Wir könnten uns auf den arroganten Standpunkt stellen, dass die Leute einfach alles richtig zu machen hätten … und es sei dann schließlich ihre Schuld, wenn wegen eines Fehlers etwas schiefgeht. Doch in der Praxis würden auch Sie und ich manchmal unabsichtlich zum falschen Zeitpunkt blinzeln. Es ist also besser, wenn wir das Problem so lösen, dass der Lösungsweg für alle praktikabel ist. Immerhin geht es um einen Menschen, dem wir helfen wollen, und nicht um eine Maschine! Beim Computational Thinking geht es also auch darum, Menschen zu verstehen.

Optimierungen vornehmen

Ist ein Muster erkennbar?

Für unseren Locked-in-Patienten lässt sich die Kommunikation beschleunigen, wenn sich bereits bei der Hälfte eines Wortes erraten lässt, um welches es sich handelt. Wenn wir bei „E-l-e-f“ angekommen sind, können wir fast darauf wetten, dass es sich um Wort „Elefant“ handeln wird. Wir können also die Regeln so abändern, dass es dem Helfer möglich ist,