Der alte Code und das neue Virus

Das Urteil der IT-Fachleute ist verheerend: Es handele sich womöglich um den zerstörerischten Softwarefehler aller Zeiten, was die wirtschaftlichen Kosten und die Zahl der verlorenen Leben anbetreffe. Beachten Sie bitte den Nachtrag/ die Aktualisierung am Ende des Textes-

Erinnern Sie sich noch an die Modellrechnung des britischen Wissenschaftlers des Imperial College in London, Neil Ferguson, der die Regierung in Sachen Corona beriet und die wissenschaftliche Grundlage für die strengen Ausgangsbeschränkungen in Großbritannien lieferte? Er prognostizierte 2,2 Millionen Coronatote für die USA und 500.000 für Großbritannien. Auf diesem Modell beruhte letztlich die Corona-Strategie der Abflachung der Kurve „flattening the curve“ durch Social Distancing und Lockdown.

Heute erweist sich dieses Modell als hysterische Überschätzung. Selbst die vielen Coronatoten in Großbritannien (ich möchte stets anfügen, dass diese Coronatoten auch die sehr wahrscheinlich hohe Zahl an Verstorbenen beinhalten, die unnötig intubiert wurden), werden bei weitem nicht die vorderen Ränge der jährlichen Todesursachenstatistik einnehmen, mit oder ohne Lockdown.

Gescheiterte Venus-Raumsonde

David Richards und Konstantin Boudnik, Gründer, CEO sowie Sotfwareleiter von WANdisco bezeichnen diese Modellrechnung nun als den womöglich zerstörerischsten Softwarefehler aller Zeiten*:*

„Die Modellierung nicht-pharmazeutischer Interventionen für Covid-19 durch das Imperial College, die dazu beitrug, Großbritannien und andere Länder zu drakonischen Lockdowns zu bewegen, wird die gescheiterte Venus-Raumsonde ablösen, die als der verheerendste Softwarefehler aller Zeiten in die Geschichte eingehen könnte, was die wirtschaftlichen Kosten und die Zahl der verlorenen Leben betrifft.“

Original auf Englisch heißt es:

„Imperial College’s modelling of non-pharmaceutical interventions for Covid-19 which helped persuade the UK and other countries to bring in draconian lockdowns will supersede the failed Venus space probe could go down in history as the most devastating software mistake of all time, in terms of

economic costs and lives lost."

Im britischen Telegraph fragen die beiden anerkannten Programmier-Spezialisten, warum die Regierung keine zweite Meinung eingeholt hat, bevor sie das Covid-Computer-Modell des Imperial College akzeptiert hat. Weiter schreiben Sie:

„Seit der Veröffentlichung des Mikrosimulationsmodells von Imperial haben diejenigen von uns, die ein berufliches und persönliches Interesse an der Softwareentwicklung haben, den Code studiert, auf dem die politischen Entscheidungsträger ihre schicksalhafte Entscheidung basierten, unsere mehrere Billionen Pfund schwere Wirtschaft einzumotten und Millionen von Menschen in Armut und Not zu stürzen. Und wir waren zutiefst beunruhigt über das, was wir entdeckt haben. Das Modell scheint völlig unzuverlässig zu sein, und Sie würden nicht Ihr Leben darauf verwetten."

Schon vor 20 Jahren Schnee von gestern

Das Modell von Imperial scheine auf einer Programmiersprache namens *Fortran* zu basieren, die schon vor 20 Jahren Schnee von gestern gewesen sei. Ihr Code sei auch für die gescheiterte Raumsonden-Mission verwendet worden, wie für die Raumsonde Mariner 1. Und sie sagen: „In unserer kommerziellen Realität würden wir jeden für die Entwicklung eines solchen Codes feuern, und jedes Unternehmen, das sich bei der Herstellung von Software zum Verkauf darauf verlässt, würde wahrscheinlich pleite gehen".

Weiter schreiben Sie: Die Modelle müssten in der Lage sein, den grundlegenden wissenschaftlichen Test zu bestehen, um bei gleichen Ausgangsparametern die gleichen Ergebnisse zu erzielen. Andernfalls gibt es einfach keine Möglichkeit, zu wissen, ob sie zuverlässig sein werden.

Das Resumee der beiden Kritiker: „Tatsächlich verwenden viele globale Industrien erfolgreich deterministische Modelle, die den Zufallsfaktor berücksichtigen. Kein Chirurg würde einen Herzschrittmacher bei einem Herzpatienten einsetzen, wenn er wüsste, dass er auf einem wohl unvorhersehbaren Ansatz beruht, aus Angst, den hippokratischen Eid zu gefährden. Warum um alles in der Welt würde die Regierung ihr Vertrauen darauf setzen, wenn das gesamte Wohlergehen unserer Nation auf dem Spiel steht?"

Nachtrag vom 20.Mai 2020

Mein letzter Beitrag wurde in den Leserbriefen zur Recht kritisiert, weil ich mich eben fachfremd bin bzgl. Computer Modellen. Leser Daniel Hirsch hat das ganze Problem mit dem Imperial Model viel besser und treffender erklärt. Imperial Model, Computersprache und Schuster bleib bei deinen Leisten. Beim letzten Beitrag, habe ich mein Fachgebiet verlassen, und einige Leser erweiterten daraufhin meinen Horizont. Zu Recht, denn ich werfe es beispielsweise Prof. Christian Drosten vor, dass er sich ausserhalb seiner Fachkompetenz z.B. zur Epidemiologie äussert und genau deswegen scheitert.

Ein Virologe muss vor den Gefahren warnen, aber Epidemiologie bedeutet gerade eben nicht, stets vom worst case auszugehen, sondern sich um echte Daten zu kümmern und diese im Sinne von Wahrscheinlichkeiten zu deuten. Daniel Hirsch ist Computerfachmann und kann die Problematik des Imperial Models deshalb viel treffender beschreiben. Ich möchte Ihnen mit seiner Erlaubnis seine Zuschrift an mich nicht vorenthalten:

Als Mathematiker mit der Spezialisierung numerische Mathematik habe ich während meines Studiums an der Uni Heidelberg zahlreiche Supercomputer entworfen, betrieben und eigene Modelle entwickelt (mit Schwerpunkt in der nichtlinearen Optimierung und im Bereich unstetige Galerkin-Verfahren). Deshalb möchte ich gern die Gelegenheit nutzen und ein bisschen von meiner Expertise beitragen.

1. Programmiersprache

Die Programmiersprache ist letztlich nur eine Form um Maschinensprache (die sprichwörtlichen „bits and bytes“) für den Menschen lesbar bzw. menschliches strukturiertes Denken für den Computer ausführbar zu machen. Die Programmiersprachen sind dabei in ihren Grundfunktionen gleich. In jeder Sprache finden sich einfache Strukturen, wie Schleifen (Zählschleife: „mache irgendwas n mal“; vorprüfende Schleife: „so lange eine Bedingung erfüllt ist, mache irgendwas“; nachprüfende Schleife: „wenn irgendeine Bedingung richtig ist, mache das, was Du vorher gemacht hast nochmal“), Bedingungen („wenn irgendwas richtig ist, mache folgendes, anderenfalls was anderes“), Zugriffe auf Dateisysteme oder auf den Monitor/Tastatur. Zudem gibt es inzwischen in vielen Programmiersprachen auch die Möglichkeit komplexere Strukturen, wie z.B. vererbbare Objekte, Templates, etc. zu bauen. Das im Detail auszuleuchten sprengt den Rahmen.

Fortran ist in der Tat eine recht alte Programmiersprache (von 1957), die ihre Vor- und Nachteile hat. Das zu diskutieren löst in aller Regel – wie so oft in der IT – einen Glaubenskampf aus. Die Unterschiede in den Programmiersprachen liegen neben der offensichtlichen anderen Syntax in der Regel eine Ebene tiefer, also wie performant ein Compiler ist, wie performant das erzeugte binary ist, wie gut der Code auf andere Plattformen übertragen werden kann, etc. Es spielt aber auch eine Rolle, wie stark die Kommerzialisierung einer Programmiersprache ist. Programmiersprachen, die für kommerzielle Software verwendet werden, sind oft stärker unterstützt und entwickeln sich schneller als weniger populäre Sprachen. Die wesentlichen Vorteile von Fortran aus meiner persönlichen Sicht sind, dass die Lernkurve um Fortran programmieren zu können, recht flach und kurz ist und sehr viel frei verfügbarer Code existiert. Es gibt Literatur, wonach gute Fortran Compiler bei Array-basierten Anwendungen schnelleren Code als flexiblere Sprachen wie z.B. C++ produzieren können sollen. Das läßt sich in der Regel aber mit „besserem“ Code kompensieren. Richtig ist in jedem Fall, dass gerade im universitären Bereich Fortran weite Verbreitung hat und es eine lebendige Community gibt, die die vorhandenen Compiler und den Standard (zuletzt 2018) weiterentwickelt. Es ist daher nicht ungewöhnlich Programme in Fortran in diesem Umfeld zu finden.

2. Universitäre Softwareentwicklung

In Mathematik und Physik ist es weit verbreitet Software für verschiedene Zwecke zu entwickeln. Oft wird die Software für eine bestimmte Klasse Großrechner (z.B. Parallelrechner) oder gar nur für einen ganz bestimmten Rechner entwickelt. Im Vordergrund steht dabei immer die Mathematik oder die Physik und weniger die informatische Qualität der Implementierung. Da es ein recht großer Aufwand ist, jedes Einzelproblem, welches man lösen möchte, dediziert zu programmieren, verwendet man den Code in sogenannten Bibliotheken immer wieder. Darin sind mathematische Formen, wie Vektoren, Tensoren, Gitter, etc. ebenso wie die mathematischen Modelle und Algorithmen, die mit diesen Strukturen umgehen müssen, bereits implementiert. Diese Bibliotheken werden dann – meist kostenlos – allen anderen Wissenschaftlern zur Verfügung gestellt. Die Entwicklungsteams sind klein und es wird nicht nach kommerziellen Standards entwickelt. Man kann nicht erwarten, dass die Software einem IT-Qualitätsassessment standhalten wird. Daraus zu folgern, dass die Software falsch rechnet, ist jedoch voreilig.

3. Mathematische Modellierung

Wenn ein Modell aufgestellt wird, bedient man sich für gewöhnlich zunächst der Mathematik und zwar ganz klassisch mit Tafel und Kreide. Die Implementierung erfolgt danach mit Hilfe der vorhandenen Bibliotheken. Wenn das Modell implementiert ist, verwendet man üblicherweise Testdaten und -probleme um zu zeigen, dass die Software funktioniert. Oft werden die Ergebnisse auch publiziert, da neue Modelle / Algorithmen in der Regel nur dann implementiert werden, wenn sie aus wissenschaftlicher Sicht neu sind oder wenn die Implementierung aus irgendeinem Grund besser ist als bisherige Implementierungen (z.B. schnellere Konvergenz, geringerer notwendiger Speicherplatz, höhere Genauigkeit, etc.). Andernfalls macht sich keiner die Mühe vorhandene Modelle nochmal zu implementieren – außer vielleicht im Rahmen einer Übung zu einer Vorlesung. Hier liegt auch der Unterschied zur Venus-Sonde. Die Software für die Venussonde war für eine einmalige Anwendung gedacht (nämlich genau diese Sonde auf dieser Mission) und konnte unter Echtbedingungen nie getestet werden. Die Modelle von Neil Ferguson sind mehrfach getestet und mit anderen Modellen gegen gerechnet worden. Das schließt Programmierfehler freilich nicht aus, aber bisher gibt es keinen Hinweis auf einen solchen. Wahrscheinlicher sind Fehler in den Ausgangsdaten.

4. Numerischer Fehler

Das größte Problem bei mathematischen Modellen ist der sogenannte numerische Fehler. Die meisten Strukturen in der alltäglichen Mathematik basieren auf den reellen Zahlen. Ohne die reellen Zahlen funktioniert unsere Welt nicht mehr. Bestes Beispiel: Der Umfang eines Kreises. Ohne die Zahl pi, ist der Umfang nicht exakt zu errechnen. Dummerweise hat pi unendlich viele Nachkommastellen (die wir logischerweise auch nicht alle kennen). Daher kann man eine solche Zahl nicht auf einem Computer abbilden, der ja nur über endlich viel Speicher verfügt, selbst wenn wir alle Nachkommastellen kennen würden. Man behilft sich durch Runden der Zahl (typischerweise auf 16 Stellen) . Für alltägliche Probleme reicht die Rundungslogik auf normalen

Rechnern völlig aus. Allerdings gibt es noch ein weiteres Problem: Ein mathematisches Modell besteht aus sehr vielen Einzeloperationen. Die Rechenoperationen reagieren dabei unterschiedlich gut auf die Eingangsdaten und die Einzelfehler jeder Operation können sich im gesamten Modell aufschaukeln. Es ist daher sehr wichtig, die Stabilität/Zuverlässigkeit des Algorithmus gegen derartige Effekte zu kennen.

5. Eingangsfehler

Bei allen Algorithmen gilt das „shit-in-shit-out"-Prinzip. Arbeitet man mit Daten, die einen großen Fehler beinhalten, dann kommt am Ende ein noch viel größerer Fehler raus (wie viel schlimmer sagt uns die Stabilität eines Modells). Die Programmiersprache ist nicht entscheidend, denn es kommt überhaupt nicht darauf an, ob ein mathematisches Modell in Fortran, C++, Basic oder von mir aus auf Lochkarten implementiert wird. Genauso wenig wie es darauf ankommen sollte, ob man eine Addition mit dem Rechenschieber, dem Abakus, einem Taschenrechner oder im Kopf durchführt. Ich bin davon überzeugt, dass das Problem des Modells von Neil Ferguson in den Eingangsdaten liegt. Das von ihm verwendete Modell basiert im Wesentlichen auf folgenden Daten:

- Todesrate (CFR) in Hubei – hierzu hat sich Neil Ferguson 39 Todesfälle von diversen Webseiten zusammen gesucht und davon 26 verwendet (leider ohne zu sagen, welche 26)
- CFR auf den Evakuierungsflügen von China nach z.B. Deutschland, Japan und Malaysia (insgesamt 290 Fälle)
- Aufwuchsrate der Epidemie, also der tägliche Zuwachs an neuen Fällen – diese Aufwuchsrate wurde auf Basis der offiziellen Meldungen abgeleitet
- Zeitraum zwischen Infektion und Tod / Erholung

Damit gibt es folgende Probleme:

- Zunächst ist die CFR nicht zuverlässig. Sie hatten das so schön formuliert – „viele Patienten sterben nicht **an** COVID-19 sondern **mit** COVID-19". Diese Erkenntnis, die ja auch im Wesentlichen auf der Arbeit von Prof. Schirmacher und seinen Kollegen beruht, hatte man zu diesem Zeitpunkt (Anfang Februar) noch nicht, sondern hat platt angenommen, wer positiv auf COVID getestet wurde und starb, ist an COVID-19 gestorben. Die Todesfälle in Hubei betreffen allesamt ältere Menschen (nur sechs Opfer sind unter 60 Jahre, der Jüngste ist 36, wobei nicht klar ist, welche 26 Personen aus dem Datensatz mit 39 Opfern in die Analyse eingeflossen sind). Bereits an diesen frühen Daten war absehbar, dass ältere Menschen deutlich stärker betroffen sind als jüngere. Das Modell lässt das vollkommen unberücksichtigt. Prof. Ioannidis hat bereits früh auf die mangelnde statistische Güte der vorhandenen Daten hingewiesen.
- Es fehlten überall Testkapazitäten, daher wurde nicht repräsentativ getestet, sondern in der Regel nur beim Vorliegen von (teils schweren) Symptomen und auch erstmal nur in Wuhan, also mit sehr hoher Wahrscheinlichkeit Menschen mit Vorerkrankungen und mit sehr geringer

Wahrscheinlichkeit junge, gesunde Menschen ohne Symptome, deren CFR weit geringer ist. Menschen ohne oder mit nur milden Symptomen sind in der Stichprobe aus Hubei vollkommen unterrepräsentiert.
- Umgekehrt wurden die ausgewählten Heimkehrerflüge größtenteils vollständig getestet. Deshalb überwiegen in diesem Datenset die Patienten ohne Symptome (Gesamt 10 Fälle, davon 7 asymptomatisch). Die Daten sind daher keineswegs vergleichbar.
- Bei den Heimkehrerflügen war in 50% der Fälle kein Infektionsdatum bekannt. Neil Ferguson hat daher einfach das Datum des ersten Kontakts mit den Gesundheitsbehörden angenommen. Dazu schreibt er wörtlich: „We note this is the latest possbile onset date and may therefore increase our estimates of CFR".
- Zudem wurde mit Ausbreitung der Pandemie auch die Testkapazität erhöht, so dass sich die Wahrscheinlichkeit, dass eine bestimmte Bevölkerungsgruppe in der Gruppe aller getesteten repräsentiert ist, sich mit der Zeit – quasi täglich – verändert hat. Die Hypothese, die täglich publizierte Zahl der Infizierten, würde jeden Tag im gleichen linearen Verhältnis zur tatsächlichen Zahl der Infizierten stehen, ist nicht richtig. Neil Ferguson hat das zwar in der Studie erwähnt und auch richtigerweise auf die unterschiedlichen Testkapazitäten, -qualität und -praxis in verschiedenen Ländern hingewiesen, aber die Relation zwischen Testaufwuchs und gemessenen Infektionen konnte niemand zu diesem Zeitpunkt richtig abschätzen.
- Die Aufwuchsrate der Epidemie wurde deutlich überschätzt, denn es wurden ja nicht jeden Tag gleich viele Menschen getestet, sondern jeden Tag mehr, so dass man auch immer mehr Infizierte fand. Der Aufwuchs setzt sich zusammen aus dem Aufwuchs der Testkapazitäten und dem Aufwuchs der tatsächlichen Infektion, ersteres wurde aber zu gering geschätzt.
- Für die Wahrscheinlichkeitsverteilung für den Verlauf zwischen Infektion und Tod / Genesung wurde die Wahrscheinlichkeitsverteilung der SARS-Epidemie von 2003 in Hong Kong herangezogen und verallgemeinert, so dass der Zeitraum mit 22 Tagen analog zur damaligen SARS-Epidemie angenommen wurde. Auf den ersten Blick erscheint das fragwürdig, denn zum Zeitpunkt der Studie lagen noch keine gesicherten Erkenntnisse vor, dass der Verlauf von COVID-19 dem von SARS entspricht.

Mit dieser recht wackeligen Datenlage und noch wackligeren Annahmen wurde dann mit Hilfe von verschiedenen statistischen Methoden (Bayes Statistik, Maximum-Likelihood, Kaplin-Meier) die CFR berechnet.

Die hohe Unsicherheit beschreibt Neil Ferguson selbst, denn die ermittelte CFR schwankt in den Gruppen Hubei und den Rückflügen zwischen 1,2% und 18%. Am Ende werden eigentlich nicht zusammen passende Daten miteinander vermischt und eine CFR von ca. 1% mit einem Unsicherheitsintervall von 0,5% bis 4% geschätzt. Zieht man die unbeachteten Fehler (Überschätzung der Aufwuchsrate, Dunkelziffer bei den Infektionen, Differenzierung bei der Todesursache, etc.) in die Kalkulation mit ein, reduziert sich das weiter.

Aus handwerklicher Sicht hätte das eigentlich niemals publiziert werden dürfen. Die Schwankungen in den Ergebnissen gehen über eine ganze Größenordnung und die Unsicherheit in den Eingangsdaten ist

offensichtlich. Es zeigt aber sehr schön, das Dilemma. Die Wissenschaft konnte im Februar keine klaren Ergebnisse liefern. Es wäre möglich, dass die CFR bei 18% lag. Genauso konnte sie bei 0,8% oder darunter oder bei jedem anderen Wert liegen. Die Entscheidung war daher eine rein politische. Die Politik wiederrum hat einfach die „aus der Hüfte geschossene“ Todesrate von 1% genommen und durchmultipliziert. Dabei kamen ebenso logisch wie falsch Millionen Tote heraus.

Man kann an diesem Beispiel wunderbar erkennen, wie gefährlich die mathematische Modellierung sein kann, wenn man die Annahmen nicht gründlich betrachtet. Ein ehemaliger Chef von mir (alter Controller) hat mal zu mir gesagt: „Businesscases schaue ich mir nie an. Ich glaube schon, dass die Leute richtig rechnen können. Ich schaue mir immer die Annahmen an. Da lügen sich alle in die Tasche.“ Um in solchen schwerwiegenden und unklaren Situationen eine gute Entscheidung zu treffen, reicht es nicht auf oft fragwürdige Zahlen zu schauen. Es braucht vor allem Führungsqualitäten, wozu auch ein Bauchgefühl und Risikobewusstsein gehören.

Der Beitrag erschien zuerst bei ACHGUT hier