15. Jan. 23.02.41 RES202_Automatisiertes_Softwaretesten

Willkommen zum Forschungspodcast der Helmholtz-Gemeinschaft.

Ich bin Holger Klein.

In Saarbrücken, da sitzt das CISPA, eine Abkürzung, die ich bisher immer noch nicht auswendig gelernt habe.

Das ist das Helmholtz-Zentrum für Informationssicherheit.

Und am CISPA, da arbeitet Andreas Zeller, der ist Professor für Softwaretechnik.

Und was Andreas Zeller da arbeitet und forscht, das muss derart wichtig sein, dass er 2023 schon zum zweiten Mal die höchstmögliche Forschungsförderung in Europa bekommen hat.

Kann aber auch sein, dass er seine Forschung einfach bloß super verkauft hat.

Mal gucken, ich frage nach.

Hallo Herr Zeller.

Hallo Herr Klein, einen schönen guten Morgen.

Würde es überhaupt gehen, dem Europäischen Forschungsrat, dem ECR, einen Bären aufzubinden, indem man den mit Buzzwords füttert?

Da muss ich Sie leider enttäuschen, Herr Klein.

Es wäre sonst ja, es wäre etwas, das vielleicht, sagen wir mal, in der Finanzwelt

funktioniert.

Dann könnten Sie mit fantastischen Buzzwords ankommen, "Fake it till you make it".

Aber auch da gibt es nicht allzu viele leichtgläubige Leute.

Es geht um große Summen.

Und wenn man große Summen vergibt, im öffentlichen wie auch privaten Bereich, möchte Sie, dass dort einige Prüfungen stattfinden, einige sehr wichtige Prüfungen, damit Sie diese wichtigen Entscheidungen gut treffen können.

Im Fall vom IOC ist das so, Sie haben da ein ganzes Panel von Wissenschaftlerinnen und Wissenschaftlern, die Sie sowohl von der Papierform her als auch von der, als auch in einer persönlichen Vorstellung überzeugen müssen.

Und ja, das ist nicht so einfach.

Ist das wie so ein ganz normaler Forschungsantrag stellen oder ist das umfangreicher?

Es ist zunächst einmal ein Forschungsantrag.

Das heißt, Sie haben eine Forschungsidee.

Und für diese Forschungsidee brauchen Sie Geld, damit Sie davon Leute bezahlen können, die die Forschung machen.

Die Dimensionen sind allerdings ungewöhnlich groß.

Es fängt bei anderthalb Millionen an.

Das sind die relativ kleinen IOC-Grants für Leute, die frisch in der Forschung sind und hört bei zweieinhalb Millionen auf.

Das sind die Advanced-Grants für Leute, die schon etabliert sind.

Also zweieinhalb Millionen für einen einzelnen Forscher, eine einzelne Forscherin, das ist schon ziemlich gewaltig.

Und in Deutschland können Sie durchaus eine Truppe von, sagen wir, sechs bis acht Leuten über bis zu fünf Jahre beschäftigen.

Was ist denn Ihre Forschungsidee für diese fünf Jahre, acht Leute fünf Jahre?

Ich bin unterwegs in der Softwaretechnik.

Das heißt, ich beschäftige mich damit, wie man funktionierende Software bauen kann, auch wie man existierende Software gut prüfen kann, ob sie das tut, was sie soll.

Und dort bin ich insbesondere im Bereich des Testens und der Fehlersuche unterwegs.

Und was ich erforsche, sind Techniken, die in der Lage sind, Programme automatisch zu testen, ob sie das Richtige tun.

Wenn Sie es ganz einfach haben wollen, ich baue eine schwarze Kiste, da werfen Sie dann ein Programm Ihrer Wahl rein und hinterher kommt entweder ein grünes Häkchen raus nach dem Motto "hat funktioniert" oder ein rotes blinkendes Etwas.

Das sagt Ihnen, es hat nicht funktioniert.

Aber Moment mal, das wird doch programmiert, damit es funktioniert.

Wie kann denn das nicht funktionieren?

Da müssen Sie sich anschauen, wie Programme in der Regel getestet werden.

Die werden von Hand getestet und natürlich im Einsatz.

Das bedeutet, der Programmierer selbst prüft nach, ob das Programm das Richtige tut.

Es gibt auch Tester, die ebenfalls nochmal prüfen, ob das Programm das Richtige tut.

Das sind aber überwiegend händische Tests.

Das bedeutet, wenn Sie denken, das ist eine Webseite, Sie setzen also jemanden dran, der klickt sich da durch und prüft nach, ob das Richtige auf dem Bildschirm erscheint, um das sehr einfach auszudrücken.

Genau das Richtige, für mich sehr einfach, wie in einem Zehner.

Sehr einfach, genau.

Diese händischen Tests sind allerdings aufwendig, weil sie von Menschen gemacht werden.

Und es ist auch so, dass Software sich häufig ändert.

Also wenn Sie auch eine Software in Betrieb haben, dann wollen Sie Verbesserungen vornehmen.

Und jedes Mal, wenn Sie so eine Verbesserung vornehmen, müssen Sie eigentlich hinterher prüfen, ob immer noch genau das Ganze so funktioniert, wie es vorher funktioniert hat.

Und weil das alles sehr teuer ist, ist ein Ziel der Forschung, solche Tests weitgehend zu automatisieren.

Das bedeutet, dass statt eines Menschen ein Computer diese Prüfungsarbeit übernimmt.

Das ist allerdings nicht ganz so einfach.

Und weil es eben noch viel Forschung zu tun gibt, habe ich mir überlegt, wie man das sehr gut machen kann und habe einen entsprechenden Antrag geschrieben.

Das klingt so, als würde irgendjemand beim Bau der Software rumpatzen und dann müsste Zeller kommen und hinterher aufräumen.

Ist das so?

Könnte man das Problem schon am Anfang lösen?

Naja, Sie kennen den Spruch "Errare humanum est", das heißt, Fehler sind menschlich, wo Menschen arbeiten.

Machen Sie Fehler, das ist unvermeidlich.

Das passiert mir genauso.

Auch wenn ich etwas programmiere, denke ich möglicherweise nicht an alle möglichen Grenzfälle, die passieren können.

Ich bin nicht kreativ genug, darüber nachzudenken, wie jemand mein Programm möglicherweise missbrauchen könnte.

Und weil solche Fehler eben passieren, ist es unvermeidlich, dass man hinterher testet, ob das Ganze auch richtig funktioniert.

Da kommen Sie nicht drum rum.

Das machen Sie bei allen Ingenieursleistungen, beim Autobau, beim Flugzeugbau.

Sie würden sich auch nicht in ein Flugzeug reinsetzen, wo Sie sagen, die Ingenieure sind sich ganz sicher, dass das fliegen wird.

Wir haben es noch nicht getestet.

Aber wir sind uns 100 Prozent sicher.

Setzen Sie sich rein.

Ingenieure sagen auch, wir brauchen keine Fallschirme dafür.

Also ich kann Ihnen versichern, das wird nicht so wahnsinnigen Anklang finden, weil wir eben aus der menschlichen Erfahrung heraus wissen, wir können eine Perfektion erreichen, aber wir wissen nie wirklich, ob wir diese Perfektion auch wirklich haben.

Und deswegen gehen wir davon aus, dass auch wenn Leute das Tollste programmieren, dass das immer noch mal nachgeprüft werden muss.

So, und das automatisch nachzuprüfen ist nicht ganz einfach, so haben Sie es formuliert.

Wenn Softwareleute "nicht ganz einfach" sagen, heißt das in der Regel "absolut unmöglich", zumindest aus meiner Perspektive.

Was ist das Problem da?

Also weil im Grunde müssen Sie doch eigentlich nur eine Klapperatur bauen, die die richtigen Knöpfe drückt.

Ja, das sind zwei Sachen.

Also zum einen, wenn Sie sagen, Sie müssen die richtigen Knöpfe drücken, dann ist das allein schon mal ein großes Forschungsproblem.

Weil wie bringen Sie, nehmen Sie eine beliebige Webseite oder was ähnliches, wie bringen Sie dem Computer bei, ja zum Beispiel gehen Sie auf einen Webshop oder sowas, wie bringen Sie dem Computer bei, er möge jetzt mal hergehen und sagen wir irgendwo eine Kaffeetasse kaufen.

Das wäre jetzt so ein typischer Testfall für einen Webshop.

Sehen Sie, da müssen Sie erstmal herausfinden, wie suche ich denn jetzt ein Produkt raus, wie tue ich das Produkt in meinen Warenkorb rein.

Wenn ich das Ganze in meinen Warenkorb reingetan habe, dann müssen Sie auch für den richtigen Knopf drücken, dass Sie zu den Bezahloptionen kommen.

Und dann wird es erst richtig schwierig, weil wenn der Computer dann herausfinden muss, wie gebe ich jetzt eine gültige Kreditkartennummer ein zum Beispiel.

Eine Kreditkartennummer, die hat eine Prüfsumme, Sie können nicht eine beliebige Nummer da eingeben.

Möglicherweise prüft das Ding auch sogar nach, ob zum Beispiel die Postleitzahl zum Ort passt, den Sie eingeben, korrigiert Ihren Straßennamen.

Und dann sehen Sie, dass Sie da jede Menge Kontextwissen eigentlich bräuchten, um sowas vernünftig auszufüllen.

Also von sich selbst wissen Sie, was Ihre Adresse ist, Sie wissen, was Ihre Telefonnummer ist, Ihre E-Mail-Adresse ist und was Ihre Kreditkartennummer ist.

Das finden Sie raus, wenn Sie die Kreditkarte oder die Kreditkarte aus Ihrem Portmonee holen.

Aber der Computer weiß das alles nicht.

Und das ist das eine.

Also Sie müssen auf irgendeine Weise dafür sorgen, dass Sie den Computer mit dem Wissen versehen können, dass er solche Eingaben für das Programm, was Sie testen wollen, dass er weiß, wie diese Eingaben aussehen müssen, dass man das Kontextwissen dafür auch definieren kann und dass der Computer sowas möglicherweise sogar lernen kann, indem er andere Interaktionen betrachtet.

Dann haben Sie aber als nächstes das Problem, zum einen haben Sie das Problem, diese Eingaben zu finden, aber das nächste Problem ist dann, der Computer muss auch prüfen, ob das, was am Ende rauskommt, richtig ist.

Das bedeutet, Sie müssen dem Computer beibringen, er muss in der Lage sein, Fehlermeldungen zu erkennen, sowas wie "Die benutzte Kreditkarte ist ungültig" oder aber auch den Erfolg zu erkennen.

"Halleluja, das Produkt wird jetzt versandt" oder was auch immer.

Und wenn Sie sich selber jetzt schon mal vorstellen, bei manchen Interaktionen, also ein Einkauf in einem Webshop, das ist etwas, was wir in der Regel heute alle ganz gut beherrschen.

Aber jetzt stellen Sie sich vor, Sie füllen irgendeinen Bauantrag aus oder irgendwas noch viel Komplexeres.

Was sind Sie da froh, wenn Sie zum Schluss sagen, "Boah, ich hab's endlich geschafft, ich hab alles richtig eingegeben, der Computer hat's akzeptiert, Halleluja, ich bin der Held!"

Dann haben Sie ein richtiges Erfolgserlebnis.

Und jetzt denken Sie an die ganzen Schwierigkeiten, die Sie da schon als Mensch haben, um das alles richtig auszufüllen.

Ja, und das sind die Schwierigkeiten, die ein Computer noch viel mehr meistern muss, der ja zum Beispiel eben keine Ahnung hat, was ein Bauantrag ist oder was einen Erfolg ausmacht.

Jetzt könnte ich aber doch den Computer auf diesem Webshop trainieren.

Also ich könnte doch einmal hingehen und sagen, so, das ist jetzt mein Prüfprogramm Webshop.

Da setze ich, keine Ahnung, 30 Menschen hin, die alles durchklicken, also jeden Spezialfall einmal durchnehmen.

Und dann habe ich meine Software für den Webshop.

Das ist aber nicht das, was Sie haben wollen, ne?

Was Sie machen können, ist, Sie können aufzeichnen zum Beispiel.

Sie können aufzeichnen zum Beispiel, wie sich ein Mensch da durchklickt.

Das ist schön.

Dann haben Sie aber einen Testfall, genau einen Testfall, mit dem Sie eine konkrete Kaffeetasse bei einem konkreten Webshop kaufen können.

Und dann haben Sie diesen einen Testfall.

Das ist gut.

Den können Sie immer und immer wiederholen, bis irgendjemand bei Ihnen auf die Idee kommt, die Kaffeetasse aus dem Sortiment zu nehmen und was anderes reinzusetzen.

Aber könnte man das dann nicht ersetzen durch eine Variable, die einfach nur Produkt heißt?

Ja, dann könnte man dann noch eine Suchfunktion einbauen, die sucht, welche Produkte überhaupt da sind.

Aber dann haben Sie immer nur noch genau einen Testfall.

Und der eine Testfall, mit dem Sie den erfolgreichen Pfad da abdecken, nämlich Kunde schafft es, das ganze Ding zu kaufen hinterher, das ist nur eine Sache.

Sie möchten ja auch prüfen zum Beispiel, dass Sie nicht etwa Sachen kaufen können, die nicht im Sortiment sind.

Sie möchten auch prüfen können, dass wenn der Kunde tatsächlich mit einer ungültigen Kreditkarte einkauft, dann möchten Sie sicher gehen, dass der Kauf nicht abgeschlossen wird.

Zum Beispiel sind wir neulich auf einen Weinhändler in Frankreich gestoßen.

Das war ganz lustig.

Dieser Weinhändler hatte auf seinem Webshop eine versteckte Variable zum Beispiel.

Und wenn man die gesetzt hat von Hand, konnte man kostenlos einkaufen.

Warum hat der sowas in seinem Webshop und wie ist die Adresse?

Ich verrate Ihnen das jetzt nicht.

Aber es wurde uns tatsächlich ein Paket mit Wein geliefert.

Und diese Variable hatte der Programmierer des Webshops drin gelassen, damit er selbst diese Webseite testen kann, ohne dass er jedes Mal da eine Bestellung aufgeben muss.

Und das hat er vergessen.

Wie haben Sie diese Variable gefunden?

Das gehört dann zu unserem Testverfahren dazu.

Wir suchen eben nicht nur nach dem nach dem einem Pfad, der zum Erfolg führt, sondern wir versuchen auch die vielen, vielen Möglichkeiten abzudecken, die nicht zum Erfolg führen.

Das bedeutet, wir schauen auch nach Variationen, insbesondere nach den vielen Möglichkeiten des Misserfolgs, die allesamt auch nicht zum Erfolg führen dürfen.

Wie zum Beispiel, da ist eine Variable und wenn ich die setze, ja, dann darf sowas nicht passieren.

Und da sehen Sie auch schon, wenn Sie da immer tiefer rein, wenn Sie tiefer Sie da reingehen in das Problem, desto mehr stoßen Sie auf solche Details, die allesamt irgendwo geprüft werden müssen.

Das auch bedeutet, dass wenn Sie tatsächlich einen Webshop betreiben, dann reicht Ihnen der eine Testpfad nicht aus, sondern Sie brauchen eine Vielzahl von Tests eigentlich.

Und ja, diese Tests, wenn man die allerhändisch macht, das kann man machen, das ist teuer.

Das geht aber auch dann wiederum nur so lange, bis, ja, was soll ich sagen, bis Ihr Webshop irgendwann nach Island expandiert.

Und dann haben Sie in Island, haben die Knöpfe allesamt unterschiedliche Bezeichnungen.

Dann heißt es nicht immer bestellen, dann heißt es bestellen auf isländisch, wie auch immer das jetzt heißt.

Oder aber Sie gehen, Sie expandieren nach Israel oder in ein arabisches Land, da sind die Knöpfe nicht mehr nur von links nach rechts, sondern von rechts nach links angeordnet.

Die haben möglicherweise andere Beschriftungen, sind größer, kleiner, können nicht mehr so einfach getroffen werden.

Ja, und...

Oder der Programmierer stirbt und irgendjemand findet die Variable und räumt den Webshop leer.

Ja, das sollten wir eigentlich mal machen.

Also ich sollte einfach mal eine Bestellung über 1000 Weinflaschen aufgeben.

Vielleicht fällt es ja dann irgendjemandem auf.

Ach, der weiß das gar nicht, der Betreiber?

Sie haben dem das gar nicht gesagt?

Doch, doch, natürlich haben wir das dem Betreiber gesagt.

Wir haben das dem schon vor drei Jahren gesagt, aber der hat es immer noch nicht korrigiert, dass das der Bemerkenswert ist.

Also, wie...

Also, ich gerne weiß, bitte, Herr Zeller.

Ja, also, ich glaube, Bordeaux ist die Spezialität dort vor Ort.

Wie sind Sie da vorgegangen?

Also haben Sie da einfach irgendwie eine Software ins Internet geschickt und gesagt, spuck mal, also, lauf mal über alle möglichen Webshops drüber?

Nein, das machen die nicht.

Nein.

Wir testen nicht auf echten Systemen, also auf Produktivsystemen.

Das wäre ethisch höchst verantwortungslos, weil wir würden ja tatsächlich diese Webseiten angreifen und schädigen.

Also in dem Fall den Weinhändler in Frankreich.

Das machen wir nicht, sondern wir machen uns gegebenenfalls Kopien von den Systemen oder wir arbeiten auf Open Source Systemen oder wir arbeiten in Kooperation mit den Anbietern.

Das heißt, wir geben denen unsere Testprogramme oder unsere Testdaten und testen dann damit und die berichten uns dann, was bei ihnen in aller Regel schiefgegangen ist.

Aber auf realen Systemen testen wir nicht, dann würden wir die ja regelrecht

angreifen.

Das machen wir nicht.

Aber wie sind Sie denn dann auf den Weinshop gekommen?

Das ist ganz einfach.

Wir haben Verfahren entwickelt, die eben in der Lage waren, systematische solche Variationen durchzuspielen.

Und dann haben wir Verfahren entwickelt, die in der Lage sind, solche Variationen zu erkennen.

Und da haben wir irgendwann festgestellt, dass es, wir haben auch Webseiten durchsucht von ihrem Programmcode her nach solchen Möglichkeiten.

Ja, und dann ist uns dieser Webshop ins Auge gefallen.

Das war eigentlich sehr einfach.

Und den gibt es nur einmal oder ist das so eine Standardsoftware, die alle verwenden?

Das gibt es glücklicherweise nur einmal.

Sie merken, wo ich hin will.

Also wenn Sie jetzt denken, Sie könnten jetzt, sagen wir mal, zur Bank Ihres Vertrauens gehen und hoffen, dass es da irgendeinen geheimen Parameter gibt, durch den Sie zum Millionär werden.

Das würde mich doch etwas überraschen, dass das geht.

Aber es sind natürlich auch Leute unterwegs, die genau nach solchen Schwachstellen suchen.

Und wir als Forscher versuchen da immer ein paar Schritte voraus zu sein.

Das bedeutet, wir suchen nach Möglichkeiten, Systeme zu testen, die stärker sind, die auch stärker automatisiert sind als das, was Angreifer derzeit tun.

Die auch davon ausgehen, dass das mit Kooperation der Entwickler funktioniert, sodass die Entwickler dann unsere Techniken einsetzen können, um Fehler in ihren Systemen zu finden.

Aber am Ende werden sie von den Angreifern immer wieder eingeholt, oder?

Ja, wir haben da ein klassisches Dual-Use-Problem.

Also die Techniken, die wir entwickeln, die sind sehr, die werden natürlich auch immer leistungsfähiger.

Und wenn Leute unsere Techniken nehmen und damit auf Fehlersuche tatsächlich dann in fremde Systeme gehen, dann passiert genau das.

Dann finden sie möglicherweise Schwachstellen und werden dann für die nächsten Wochen kostenlos mit Wein beliefert oder Schlemmerung.

Wenn Sie sagen, die Techniken, die Sie entwickeln – jetzt, ich bin kein Programmierer, ich bin noch nicht mal Wissenschaftler genug, um irgendwann mal ein Seminar programmieren gelernt zu haben.

Wie sehen diese Techniken aus?

Sind das auch wieder Programme, die dann irgendwo laufen?

Also ich kenne halt so Basic, Print, Go to und so.

Ja, ja, alles gut.

Also im Prinzip könnte man hergehen und kann natürlich, was weiß ich, Mausbewegungen, Tastatureingaben, Netzwerkeingaben, das kann man alles ausprogrammieren.

Und man kann tatsächlich dann für jedes Programm, was man testen will, kann man ein geeignetes Testprogramm schreiben.

Das wird auch so gemacht.

Und solche Testprogramme laufen dann automatisch ab und suchen dann sozusagen die Knöpfe, um dann draufzudrücken.

Und einem solchen Testprogramm kann man dann auch beibringen, dass ein Knopf in einem Websystem zum Beispiel verschiedene Beschreibungen oder verschiedene Positionierungen haben kann.

So was geht alles.

Das wird auch gemacht.

Solche Beschreibungen sind aber dann immer noch recht eng an das zu testende Programm geknüpft.

Und was wir suchen, sind Möglichkeiten, sogenannte Eingabesprachen zu definieren.

Das bedeutet, wir entwickeln formale Systeme, sogenannte Grammatiken, die beschreiben, in welcher Reihenfolge man welche Eingaben machen kann, um damit bestimmte Funktionen zu erreichen.

Und eine solche Grammatik sieht ähnlich aus wie, sagen wir mal, eine deutsche

Grammatik.

Die drückt Ihnen dann aus, hier kommt jetzt Subjekt, Prädikat, Objekt oder hier kommen Pronomen, Prädikat, Objekt.

Und dann können Sie für Subjekt und Prädikat und Objekt jeweils entsprechende Einsetzungen machen.

Als Kinder haben wir so ein Spiel gespielt.

Das hieß, wie hieß das, Onkel Otto sitzt lachend in der Badewanne.

Und dann durften wir für Onkel und Onkel verschiedene Verwandtschaftsbezeichnungen einsetzen, Otto verschiedene Namen.

Und für das Verb durften wir verschiedene Sachen einsetzen.

Und da haben wir das dann immer durcheinandergewürfelt mit neuen Kombinationen.

Da kamen sehr lustige Sätze bei raus.

Aber nach einem ähnlichen Verfahren funktionieren auch unsere Testgeneratoren.

Das heißt, die haben so einen groben Aufbau der Eingabe, haben die definiert.

Und dann setzen die immer und immer wieder neue Variationen ein.

Das heißt, die suchen dann nicht nur nach einer Kaffeetasse, die suchen dann auch eben nach einem ganzen Set.

Die suchen dann nach Büromöbeln, nach Weinflaschen, was auch immer.

Und die kaufen dann auch auf alle möglichen Variationen ein.

Die brechen ab, die gehen zurück, die fangen wieder von vorne an.

Und die nutzen auch gleichzeitig solche Grammatiken, um damit zu prüfen, ob die Eingabe korrekt ist.

Das heißt, die prüfen dann zum Schluss nach, ob das Ganze auch erfolgreich war.

Und die sind, wenn sie so wollen, beschreiben diese Sprachmodelle sowohl eine Eingabesprache, das heißt, drückt dies, drückt das, tippt das und das ein, als auch die Ausgabe.

Das heißt, jetzt muss das und das auf dem Bildschirm erscheinen, jetzt muss das und das im Netzwerk als Befehl, also als Ergebnis zurückkommen.

Und da sind wir tatsächlich die weltweit führenden Gruppen, weil unsere Sprachmodelle, die wir da haben, ausdrucksmächtiger sind als alles andere, was es jemals gegeben hat.

Aber Sie sind jetzt immer noch im Webshop.

Ja, das Webshop ist ein schönes Beispiel, weil wir das eben auch als Mensch bedienen.

Aber tatsächlich lassen sich unsere Techniken auch sehr schön zum Beispiel auf elektronischen Diensten anwenden.

Also wenn Computer miteinander reden zum Beispiel, dann tauschen die auch Daten aus.

Das sind dann Daten, die sind für Menschen vielleicht möglicherweise noch lesbar, aber das ist nichts, was sie noch selbst eingeben.

Da unterhalten sich die Computer miteinander.

Und insbesondere in dieser Domäne, wo Computer miteinander Daten austauschen, haben wir sehr starke Verfahren entwickelt, um in diesem Datenaustausch zu testen und zu prüfen.

Und dann geht es auch möglicherweise nicht mehr um einzelne Weinflaschen, sondern da kann es schon um Millionen Transaktionen gehen oder um auch mögliche Schäden, die sehr, sehr groß sein können.

Und da ist unsere Arbeit ganz besonders wichtig.

Mir fällt gerade auf, dass ich nicht die leiseste Ahnung habe, wie Software eigentlich funktioniert.

Ich helfe Ihnen gerne weiter.

Können Sie das mal für doofe erklären?

Also für ganz doofe?

Naja, wenn Sie es ganz einfach haben wollen, dann stellen Sie sich eine Software vor wie eine schwarze, aus unserer Perspektive ist eine Software eine schwarze Kiste.

Da kommt eine Eingabe rein, also irgendein Text oder eine Folge von Zahlen und am anderen Ende kommt wieder eine Folge von Zahlen raus.

Das ist die ganz einfache Fassung.

Wenn Sie mit einem Webshop interagieren, dann haben diese Zeichen und Zahlen, dann werden die glücklicherweise als Eingaben, sind das Eingaben von Ihnen, die Sie jetzt gerade auf einer Webseite gemacht haben.

Und das, was am anderen Ende rauskommt, ist das Ergebnis.

Das heißt, das ist die Webseite, die Sie sehen.

Und diese Zeichen und Zahlen werden von der Software intern so verarbeitet, dass das passiert, was Sie haben möchten.

Wobei diese Zeichen und Zahlen von einem einzelnen Programm entweder selbst berechnet werden können, aber das Programm kann sich dafür auch mit anderen Programmen austauschen, auch möglicherweise übers Internet.

Dann haben Sie eine Vielzahl von einzelnen Programmen, die allesamt miteinander kommunizieren und die allesamt zusammenarbeiten, um Ihre Aufgabe zu erfüllen.

Dieses Erfüllen der Aufgabe durch die Software, das sind ja dann die Code-Zeilen, die werden dann so abgearbeitet und da wird hin und her gesprungen und so weiter.

Wo docken Sie denn da jetzt Ihre Software, die das dann überprüft, wo docken Sie die denn dann an?

Also setzen Sie die irgendwo an eine Kommunikationsschnittstelle sozusagen und sagen, schnüffel mal alles ab hier?

Genau so ist es.

Wir nutzen die Kommunikationsschnittstelle, die die Software ohnehin benutzt.

Das heißt, wir docken dort an, wo die Software zum Beispiel übers Netz sich mit anderen Programmen unterhält.

Dort docken wir an und nutzen diese Kommunikation.

Für den Webshop ist das so, da haben Sie Ihren Webbrowser.

Das ist eine Komponente, die kommuniziert mit einem Web-Server irgendwo am anderen Ende.

Und da nutzen wir, docken wir auch genau an diesen Kommunikationsschnittstellen an.

Und unsere Techniken sind in der Lage, diese Kommunikationssprachen zwischen einem Web, zwischen Ihrem Browser und dem Server, diese Kommunikationssprachen zu erfassen.

Das heißt, wir können damit lernen, wie diese Sprachen aussehen.

Wir können diese Sprachen auch selbst sprechen.

Also unsere Programme können diese Sprache selbst sprechen und sind deswegen in der Lage, eine Vielzahl von Interaktionen zwischen Ihrem Webbrowser und dem Server am anderen Ende zu automatisieren.

Und auf diese Art und Weise können wir kräftig und sehr umfassend und rund um die Uhr beide Seiten des Ganzen testen.

Ihre Software lernt also sozusagen erst mal Deutsch, indem sie zwei Deutschen dabei zuguckt, wie sie möglichst viel Deutsch reden.

Und danach guckt sie sich eine andere Kommunikation an und kann dann beurteilen, das ist Deutsch oder das ist nicht Deutsch.

Wenn Sie so wollen.

Das ist eine sehr grobe Vereinfachung.

Anders verstehe ich es leider nicht.

Aber tatsächlich ist das so, dass viel der Kommunikation zwischen einzelnen Computern auch ein bisschen sich so anfühlt wie eine Unterhaltung zwischen Menschen.

Das heißt, die sagen dann, ich möchte gerne so und so viel Geld von A nach B überweisen.

Und dann sagt der andere Computer, das ist ja sehr schön, aber wer bist du überhaupt?

Und dann sagt der erste Computer, ich bin der und der.

Und dann sagt der andere Computer, kannst du das auch beweisen, dass du ein der und der bist?

Und dann sage ich, ja, ich kann das beweisen, weil ich habe hier mein Geburtsdatum und andere geheime Informationen.

Sagt der andere Computer, okay, das ist schön, wunderbar, dann nehme ich das jetzt mal an.

Aber kannst du mir auch beweisen, dass du nicht nur der und der bist, sondern dass du noch ein weiteres Geheimnis hast?

Und dann geht es so hin und her, bis dann irgendwann die Überweisung tatsächlich genehmigt ist, dass der Bankcomputer das auch annimmt.

Und so unterhalten sich die beiden Computer miteinander.

Das sieht nicht viel anders aus, als wenn sie sich mit einem Bankmitarbeiter persönlich unterhalten.

Außer dass Dinge wie, was weiß ich, Diskussionen über das Wetter oder wie es der Familie so geht, dass das wegfällt.

Ihre Software muss aber immer noch wissen, dass es sich um eine Bank handelt.

Oder sind Sie so variabel, dass Sie einfach sagen können, hier, guck mal, ob das, was da passiert, sinnvoll ist?

Unsere Software kann einerseits aus existierenden, wenn wir beobachten, was zwischen zwei Rechnern passiert, dann können wir daraus lernen, wie solche Transaktionen typischerweise aussehen.

Und dann können wir daraus ableiten, wie die Sprache tatsächlich aussieht.

Wir haben auch ein sehr schönes Verfahren entwickelt, die Programme auf beiden Seiten zu analysieren, dass wir herausfinden können, welche Sprachen dort jeweils gesprochen werden.

Das ist hilfreich, weil damit kriegen wir auch die, sagen wir mal, ungewöhnlicheren Sprachelemente raus, die nur selten beobachtet werden oder nie beobachtet werden können.

Und das können wir dann nutzen, um uns dann selbst gegenüber der Software als zum Beispiel ein Kunde oder uns gegenüber einem Kunden als eine Bank auszugeben.

Und dann können wir so tun, als wären wir die Bank und können dann sehen, ob das Kundenprogramm korrekt funktioniert.

Können Sie auch erkennen, ob es eine Bank oder ein Kaffeetassenhändler ist?

Ja, die entsprechenden Transaktionen sehen unterschiedlich aus.

In einem Fall kaufen Sie eine Kaffeetasse für 3,50 Euro und lassen das an Ihre

Tante Elfriede schicken.

Und im anderen Fall überweisen Sie 5 Millionen an Ihr Nummernkonto in Dubai.

Ja, da sind schon andere Daten und andere Dimensionen jeweils im Spiel.

Warum muss daran überhaupt noch geforscht werden?

Das klingt so, als wüssten Sie längst, wie es geht.

Und jetzt gibt man es ein paar Codern und die programmieren das dann vor sich hin und dann ist gut.

Ja, das ist eine sehr gute Frage.

Danke.

Das ist eine sehr gute Frage.

Wir würden uns hier alle wünschen, dass Software von vornherein so geschrieben wird, dass sie eben keine Fehler mehr hat, dass alles richtig funktioniert.

Und in dem Moment, wo, sagen wir, große Vermögenswerte oder gar Menschenleben mit im Spiel sind, wird natürlich auch ein sehr viel höherer Aufwand getrieben.

Wenn es zum Beispiel um Flugzeuge, Züge, Autos geht, wo Menschenleben dranhängen können.

Da wird natürlich ein immenser Aufwand auch für das Testen betrieben, das ist klar.

Aber die Automatisierung lässt immer noch zu wünschen übrig.

Das liegt daran, dass es bisher zu wenig Möglichkeiten gab, das Ganze zu automatisieren in einem Maße, dass man es immer und immer wieder neu anwenden kann.

Ich kann immer ein Programm schreiben, was ein anderes Programm auf Herz und Nieren prüft.

Das ist immer möglich, aber das ist sehr teuer und sehr aufwendig.

Und wenn ich eine Vielzahl von Programmen habe, die allesamt getestet werden müssen, dann muss ich diesen Aufwand immer und immer wieder betreiben.

Und bei jeder Änderung des entsprechenden Programmes muss ich mein Testprogramm wieder ändern.

Das ist alles teuer.

Und ab einem gewissen Punkt sagen die Anbieter, das lohnt sich jetzt hier nicht mehr, weil der Aufwand für das Testen in einer schlechten Relation zum möglichen Schaden, der dabei entstehen kann.

Das heißt ganz konkret, im Zweifelsfall ist es billiger, die Angehörigen dafür zu entschädigen, dass das selbst fahrende Auto einen Unfall verursacht hat, als die Software überprüfen zu lassen.

Wenn es um Menschenleben geht, dann ist natürlich jeder Aufwand gerechtfertigt.

Aber offensichtlich scheint es diesen Weinhändler in Frankreich, den wir erwähnt haben, für den ist es offensichtlich billiger, ab und zu offensichtlich Weinflaschen an Leute zu bestellen, die nicht bezahlen, als den Fehler zu reparieren.

So etwas kann durchaus passieren.

Jede Software, die unterwegs ist, hat irgendwo Fehler.

Es gibt entsprechende Verzeichnisse, wo drinsteht, welche Fehler in der Software vorhanden sind.

Und wo die Entwickler entweder nicht hinterherkommen, diese Fehler zu reparieren, oder aber auch einfach sagen, das lohnt sich für uns nicht, diesen Fehler zu reparieren, weil der Fall, der darin beschrieben ist, der ist einfach zu selten.

Oder der ist einfach zu unbekannt, oder es ist zu unwahrscheinlich, dass das ausgenutzt wird.

Der ist noch zu unbekannt.

Jeder und jede, die diesen Podcast hier gehört hat, wird doch jetzt sämtliche Weinshops Frankreichs durchstöbern und gucken, wo da ein Fehler sein könnte, dass man billig bestellen kann.

Aber kann ich Ihnen auch wieder sagen, der Aufwand, den Sie betreiben können, um sämtliche Weinshops Frankreichs zu besuchen, steht auch wiederum nicht in der Relation zu dem Gewinn, den Sie dabei bezahlen.

Stimmt auch.

Stellen Sie einen Programmierer ein, der Ihnen das macht, der setzt sich da acht Stunden hin und hinterher haben Sie eine Weinflasche, die hat sich 800 Euro gekostet.

Vielleicht müssten Sie ein paar Weinflaschen mehr bestellen.

Aber das sind dann die ökonomischen Dinge, die dort zum Einsatz kommen.

Der Haken ist, und da sind wir an einem interessanten Punkt unserer Arbeit, dass wenn wir es einfacher machen, den Test zu automatisieren, geben wir gleichzeitig auch in gewisser Weise eine Bauanleitung für mögliche Angreifer mit.

Das heißt, wir sind da in einem Dilemma drin.

Je besser unsere Techniken dafür sorgen, dass Entwickler ihre Software testen können, desto mehr weisen wir auch mögliche Angreifer in den Weg.

Passt mal auf, hier ist möglicherweise ein Weg, wie man in Frankreich Weinflaschen kostenlos bestellen kann.

Und wir müssen deswegen auch, was zum Beispiel unsere Forschungsergebnisse angeht, müssen wir auch aufpassen, dass wir die so gestalten, dass unser Wissen, das wir weitergeben, Entwicklern einen Vorsprung bereitet gegenüber möglichen Angreifern.

Dass also Entwickler einen Vorsprung haben, ihre Software sicher zu machen, bevor Angreifer das nutzen, um die Fehler auszunutzen.

Ist das überhaupt möglich?

Das ist durchaus möglich, zum Beispiel wenn unsere Techniken grundsätzlich voraussetzen, dass sie Zugang zum Quellcode des Programms haben müssen.

Wenn unsere Techniken voraussetzen, dass Domänenwissen vorhanden ist, dass also der Entwickler da irgendwas reinstecken muss, was ein Angreifer typischerweise nicht hat, dann schaffen wir das, den Entwicklern diesen Vorsprung zu geben.

Wie was reinstecken muss, was der Angreifer nicht hat?

Das ist sehr einfach.

Wenn ich zum Beispiel eine Bankensoftware teste, und die Bankensoftware fragt mich, was ist denn das geheime Passwort?

Und das kenne ich jetzt nicht.

Dann sagt die Bankensoftware, Dankeschön, das war's.

Und dann komme ich ein paar Sekunden später wieder an, Hallo, ich bin wieder da.

Und die Bankensoftware fragt mich wieder nach dem Passwort, ich habe das wieder nicht.

Nach einer bestimmten Anzahl von Versuchen, dann sperrt mich die Bankensoftware einfach aus.

Sie sagt dann, tut uns leid, die kommen hier nicht weiter.

So, das war's dann soweit.

Und dann ist auch für den Angreifer, egal wie smart er ist, wenn er dieses Wissen nicht hat, dann ist er im Feierabend.

Und der Entwickler hingegen, der das Passwort kennt, oder der eine andere Zugangsmöglichkeit hat, der kann natürlich sagen, hier ist das Passwort, und jetzt teste ich mal ordentlich, und jetzt kann ich da tausende und abertausende Tests durchfahren, nach allen Regeln der Kunst, und kann auf diese Art und Weise seine Software sehr viel besser gegen Angreifer stellen, als dass ein Angreifer dieses Wissen nutzen könnte.

Das, was Sie da zu bauen versuchen, ist das vergleichbar mit einem Universalschlüssel?

Also eine Software, die ich auf ein beliebiges System werfen kann, und das dann

automatisch testet?

Ja, das ist die Idealvorstellung.

Dass wir letzten Endes eine Blackbox haben, da stecke ich mein Programm rein, und unser Testprogramm findet von selbst heraus, wie das Programm getestet werden muss, und erzeugt dann jede Menge verschiedener Eingaben, um dann wirklich alle möglichen Verhaltensweisen abzudecken, und am anderen Ende kommt dann ein Bericht raus und sagt, pass auf, in aller Regel ist dieses Verhalten so und so, und wir denken, dass das auch das erwartete Verhalten ist, aber dann haben wir noch ein paar Sonderfälle gefunden, wo sich das Programm ganz anders verhält, und es wäre vielleicht ganz gut, da nochmal reinzuschauen.

Unser Programm kann nicht entscheiden, ob das Verhalten korrekt ist oder nicht, das muss immer noch ein Mensch entscheiden.

Ja, aber es ist halt Statistik, ne?

100 Mal hat es so ausgesehen, einmal sah es so aus, warum hat es einmal so ausgesehen?

Guckt ihr das mal an?

Genau, und diese Ausnahmen sind dann, weil an dem Entwickler natürlich auch besondere Interessen haben.

Also andersrum wiederum beim Weinhändler in Frankreich, 99 Prozent der Kunden bezahlen ihre Weinflasche, ein Prozent zahlt nicht, das eine Prozent ist das Interessante.

Wie weit sind Sie davon entfernt?

Weil das klingt jetzt für so einen Laien wie mich, klingt das gar nicht so kompliziert.

Es klingt aufwendig, aber nicht kompliziert.

Ja, es ist leider, blöderweise ist es genau umgekehrt.

(Lachen) Treffer.

Blöderweise ist es genau umgekehrt, und unsere Aufgabe ist es, dieses komplizierte Problem zu lösen, sodass wir eine einheitliche und immer wieder verwendbare Möglichkeit haben, diese Sprachen, mit denen das Programm kommuniziert, auszudrücken und diese Sprachen auch immer wieder zu nutzen.

Und wenn wir das einmal gelöst haben, dass wir das hinbekommen, dann ist es eben gar nicht mehr aufwendig, das Ganze an ein neues Programm anzupassen.

Das ist momentan, es ist umgekehrt momentan, es gibt einen nicht komplizierten, aber sehr aufwendigen Weg zu testen.

Deswegen wird nicht genug getestet, oder na ja, manchmal auch kriminell zu wenig getestet.

Und wir wollen das genau umdrehen.

Wir möchten automatisches Testen sehr viel vereinfachen, sodass die Hürde, es einzusetzen, sehr viel niedriger ist.

Aber Sie müssen Ihrer Software doch eigentlich nur sagen, hier, mach alle Eingaben und protokolliere.

Ja, das wäre schön.

Blöderweise haben Sie dann auch eines dieser hübschen Detailprobleme.

Wir nennen das eine kombinatorische Explosion.

Es gibt einfach zu viele Möglichkeiten, mit der eine Software interagieren kann.

Denken Sie darüber nach.

Nehmen wir einfach mal an, Ihre Software nimmt einfach nur Buchstaben entgegen.

So, von A bis Z.

So, und jetzt haben Sie 26 verschiedene Buchstaben von A bis Z.

Gut, machen Sie 26 Tests draus.

Die bestehen aber nur aus einem Buchstaben.

Jetzt versuchen Sie mal, ein ganzes Wort zu bilden.

Und jetzt müssen Sie alle Wörter durchgehen.

Sagen wir alle Wörter mit vier Buchstaben.

Das sind dann 26 mal 26 mal 26 mal 26. 26 hoch 4.

Ich kann die Berechnung gerade mal für Sie machen.

Sehen Sie, das hat man davon, wenn man Computer ist.

Aber wenn ich hier 26 nehme und das hoch 4 mache, dann komme ich hier bequem raus.

456.000 Möglichkeiten.

Ja, das sind alle Buchstaben mit nur vier Zeichen.

Aber jetzt gibt es da Befehle, die haben 20 Zeichen oder IBAN-Nummern oder sowas in der Art.

Die haben in Deutschland, wie viele haben die, 18 Zeichen oder sowas.

Ja, und dann ist das nur ein einziges Datum, das Ganze.

Und Sie haben dann in Wirklichkeit tausende von solchen Dingern mit drin.

Und die auch wieder untereinander kombinierbar sind.

Ja, untereinander kombinierbar sind sie.

Ja, ich verstehe.

Und dann sehen Sie, dass die Zahl der Tests leider beschränkt sein muss.

Weil natürlich kann ich, und das ist ganz einfach, wir nennen das Monkey Testing.

Ich kann einfach alle Kombinationen durchgehen der Reihe nach.

Und das heißt deswegen Monkey Testing, weil das basiert auf dem Gleichnis, dass wenn ich einen Affen nur lang genug an einer Tastatur sitzen lasse, dann kann er mir sämtliche Werke von Shakespeare abtippen.

Einfach durch zufälliges Kombinieren.

Und dann habe ich sämtliche Werke von Shakespeare, habe ich auch sämtliche Werkwerke, die Shakespeare eigentlich noch schreiben wollte, wäre er nicht so früh geschroben.

Ich habe auch sämtliche Kritiken von allen Shakespeare-Aufführungen.

Nur blöderweise können Sie dann ganz schön ausrechnen, dass Sie da mit dem erwarteten Lebensalter des Universums das nicht mehr hinbekommen.

Ja, man braucht halt einfach mehr Affen.

Das habe ich auch schon mal ausgerechnet. (lacht) Das habe ich auch schon mal ausgerechnet.

Und Sie müssten die, also schon für einfache Probleme, bräuchten Sie für diese Affen mehr Wasser, als in allen Ozeanen der Welt vorhanden ist.

Also rein technisch ist das eine ganz andere Herausforderung.

Okay, ich habe es.

Quantenaffen.

Ja, vielleicht.

Es ist auch so, dass die Affen, die ganz unten sitzen, die kommen nicht mehr zum Tippen, weil die haben nämlich mehrere hundert Meter weitere Affen über sich drüber.

Also wenn Sie die ganzen Ozeane in Affen verwandeln, auch dann kommen Sie nicht über wenige Zeichen hinaus.

Tut mir leid, ich mache das.

Wir kriegen auch Ärger mit Tierschützern, wenn wir das machen, ganz abgesehen davon, dass wir kein Wasser mehr zum Trinken haben.

Dann simulieren die die Affen.

Das ist ja, was Sie am CISPA machen.

Sie simulieren diese Affen ja im Grunde.

Ja, das nützt mir aber auch nichts, auch wenn ich jetzt einen wahnsinnig schnellen Computer habe.

Haben wir natürlich die hunderttausend von Eingaben pro Sekunde hinbekommen?

Haben wir auch.

Das nützt Ihnen auch noch nichts, weil mit jeder weiteren Entscheidung, die dazukommt, mit jedem weiteren Zeichen, was dazukommt, multipliziert sich die Anzahl.

Sie haben da ein exponentielles Wachstum.

Und ja, irgendwann habe ich alle Computer der Welt, die versuchen, jetzt eine Kaffeetasse zu bestellen.

Und ja, dann kommt der Nächste an und sagt, ich brauche jetzt aber noch die estnische Kaffeetasse.

Und dann brauche ich nochmal 26 Mal so viele Computer und die Erde ist schon voll.

Und ja, so funktioniert es einfach nicht.

Tut mir leid.

Da setzen Sie die Mathematik einfach hin.

Da setzen Sie die Mathematik in die Grenzen.

Wie lösen Sie das Problem?

Weil das ist ja eigentlich das, was Sie machen wollen, oder?

Ja, wir lösen das Problem dadurch, indem wir versuchen, das Ganze zu abstrahieren.

Das bedeutet, statt dass wir jetzt zum Beispiel der Reihe nach erstmal eine Kaffeetasse und dann eine blaue Kaffeetasse und dann eine orange Kaffeetasse und dann eine rote Kaffeetasse bestellen.

Das wird ja so der Reihe nach durchspielen.

Gott, wenn ich mir vorstelle, wir sind da bei Amazon und rufen einmal nach und bestellen jedes Produkt einmal.

Dann habe ich auch eine endlose Kette von Lieferfahrzeugen, die sich in die Intimäe nach Hause wollen.

Vielleicht sollte ich zusehen, dass mir das CISPA eine spezifische Kreditkarte gibt.

Genau.

Und den Scheiß muss ich dann alles zurückschicken.

Das wäre ja sehr unterhaltsam.

Nein, nein.

Was wir da machen, ist, wir abstrahieren das Ganze und wir sagen dann, also gut, wir nehmen einfach an, hier gibt es ein Ding, das nennt sich Produkt und das ist irgendwo in der App schon vorhanden.

Und ich gehe davon aus, dass wenn es für die eine Kaffeetasse funktioniert, dass es auch für die anderen Kaffeeposten funktioniert.

Und dann kann ich dann mit diesem einen Produkt, was ich jetzt symbolisch immer mit einem Repräsentanten dieses Produktes, also der Kaffeetasse, die wir jetzt da haben, das spiele ich dann immer und immer wieder durch, auch unter verschiedenen Variationen.

Und dann habe ich noch weitere solcher Kategorien.

Also ich habe ein Produkt, ich habe eine Zahlungsart zum Beispiel.

Möglicherweise habe ich verschiedene Produktkategorien.

Was weiß ich, es gibt Produkte, die werden mit der Spedition verschickt, weil sie so groß sind.

Andere Produkte werden im Brief verschickt, weil sie so klein sind.

Das will ich auch abdecken.

Aber ich kann dann letzten Endes mich auf eine Zahl, auf eine handhabbare Zahl von möglichen Repräsentanten für jede Kategorie einschießen.

Und dann, das sage ich auch, ich muss jetzt nicht alle Kombinationen der Reihe nach durchprüfen, sondern es reichen mir aus, wenn ich zum Beispiel jedes Paar von Möglichkeiten durchspiele.

Das heißt, ich bezahle eine Kaffeetasse per Lastschrift, ich bezahle sie per Kreditkarte, ich bezahle sie per Vorkasse, so weit, so gut.

Und dann lasse ich mir einmal ein anderes Produkt, lasse ich mir ein Klavier per Spedition liefern und einmal versuche ich, es in den Brief zu stecken.

Und dann kann ich dann hergehen und kann dann sagen, wenn ich eine gewisse Anzahl von Kombinationen auf diese Art und Weise ausprobiert habe, dann hoffe ich, dass es dann eine ausreichend große Stichprobe ist, mit der ich dann alle möglichen Variationen abgedeckt habe.

Aber genau da ist doch dann wieder die Sollbruchstelle, und zwar in der Hoffnung, dass es eine ausreichend große Stichprobe ist.

Ja, das ist eine Hoffnung, dass es eine ausreichend große Stichprobe ist.

Und deswegen kann Ihnen auch die beste Anzahl von Tests nicht garantieren, dass nichts schief geht, weil Sie eben nur eine beschränkte Menge von Tests machen können.

Das ist dann auch wiederum wie beim Flugzeug.

Sie machen eine definierte Anzahl von Flügen mit definierten Parametern, wo Sie alle Systeme der Reihe nach durchprobieren.

Und wenn Sie dann Ihr neues Flugzeug ein Jahr lang unter allen möglichen Bedingungen getestet haben, dann sagen Sie, gut, jetzt haben wir alles wenigstens einmal gesehen und wir haben auch alle Kombinationen von Dingen jetzt wenigstens einmal gesehen.

Dann ist die Restwahrscheinlichkeit, dass etwas schief geht, entsprechend gering.

Und dann kann das Ding noch irgendwann tatsächlich in die Produktion gehen.

Ist das dann letztlich Ihre Forschung, der Versuch, den Hoffnungskorridor zu verschmälern?

Ja, vor allen Dingen ist meine Forschung dahinter überhaupt erstmal die Situation zu bekommen, dass ja solche Tausende von Tests überhaupt mal fahren können.

Wenn Sie sich anschauen, wie Software momentan getestet wird, dass es in sehr vielen Fällen nicht annähernd so weit, dass da alle Situationen durchgeprüft werden.

Also auch, sagen wir mal, nach den Regeln der Technik, dass man sagt, ich will jetzt alles einmal ausprobieren, das ist hinten und vorne nicht so weit.

Nehmen Sie ein beliebiges Software-System, was, sagen wir, in Ihrer Kommune eingesetzt wird, wo zum Beispiel Passdaten oder Führerscheindaten oder was auch immer verarbeitet werden und geben Sie dem Ding mal, sagen wir, einen arabischen Namen in arabischer Schrift oder geben Sie dem Ding ein negatives Geburtsdatum oder ähnliches, werden Sie feststellen, wie schnell diese Systeme anfangen zu stottern und zu spucken, wenn sie überhaupt noch weiter funktionieren.

Das ist bisher nicht so das riesengroße Problem, solange diese Systeme eben nur von den Mitarbeitern in der Kommune bedient werden.

Warum sollten die ein negatives Geburtsdatum eingeben?

Aber wenn Sie solche Systeme dann eben vernetzen mit anderen Systemen und dann auf einmal über das Internet darauf zugegriffen werden kann, dann kann das ein großes Problem werden und diese Systeme sind in aller Regel nicht darauf vorbereitet.

Ein Problem werden im Sinne von, wenn da jetzt jemand ein negatives Geburtsdatum eingibt, könnte das möglicherweise dazu führen, dass irgendwo eine Tür aufgeht, durch die ich eigentlich nicht gehen sollte?

Naja, fangen Sie an mit einem einfachen Beispiel.

Sie machen eine Banküberweisung und ich gehe jetzt her und überweise Ihnen ein Honorar von minus 100 Euro.

Dann würde ich jetzt erwarten, dass von mir 100 Euro auf Ihr Konto überwiesen werden, obwohl das eigentlich... Ja, das wäre wunderschön.

Da bin ich jetzt mal optimistisch, dass die Banksoftware das erkennt.

Also, ich... Probieren können wir es mal, oder?

Wir können das mal ausprobieren, aber ich bin jetzt einfach mal optimistisch und hoffe, dass die Banken ihr Geschäft zumindest insofern absichern und gut verstehen.

Aber wer weiß, ob nicht irgendein Praktikant ein System baut, um solche Überweisungen auszuwerten, die jetzt da hin und her geschickt werden, oder die vorab zu prüfen, bevor sie an den Hauptrechner der Bank geschickt werden.

Und der hat dann eben mal so ein kleines Programmchen geschrieben, was einfach mal aufsummiert oder was auch immer.

Und wer sagt mir, dass dieser Praktikant daran gedacht hat, dass diese Nummer möglicherweise negativ sein kann?

Und dann können durchaus subtile oder auch nicht so subtile Probleme entstehen.

Und da so etwas passiert sehr leicht.

Da haben Sie auf einmal ein Feld, wo ein Name drinsteht und dann schicke ich Ihnen einen Namen, Andreas Zeller, Freiherr, eh noch zu Gutenberg, was weiß ich was, mit 300 Vornamen und 20 Fürstentiteln und sonst irgendwas.

Und dann stellt sich heraus, dass die Software am anderen Ende möglicherweise aber nur, sagen wir, auf 64 Zeichen oder so was vorgesehen ist.

Weil irgendjemand mal festgelegt hat, dass es überhaupt keine Namen gibt, die länger als 64 Zeichen sein können.

Niemand hat so was.

Und dann komme ich dann doch an und sage, ich bin hier der sonst was.

Ja, und was passiert dann?

Dann können Sie Glück haben.

Und der Programmierer hat daran gedacht und hat dann dafür gesorgt, dass die Software das korrekt ablehnt und sagt, hier, tut mir leid, kommen wir bitte mit weniger Vornamen an.

Oder aber Sie haben Pech und der Programmierer hat das nicht gedacht.

Und dann kann es zum Beispiel zu einem Phänomen kommen, was wir einen Pufferüberlauf nennen.

Das heißt, die Zeichen landen dann in den ersten 32 Zeichen des Speichers, wo der Name eigentlich hingehört.

Und die weiteren Zeichen füllen dann beliebige weitere Speicheradressen im Computer.

Was weiß ich, Passwörter, die füllen Geburtsdaten.

Was überschreiben das alles mit meinen vielen, vielen Fürstentiteln?

Und dann kann die Software beliebige Dinge tun.

Sie könnte einfach abstürzen, das wäre schon mal was.

Sie kann sich auch aufhängen, sie kann überhaupt nicht mehr funktionieren.

Sie kann mir möglicherweise aber auch über, wenn ich einen sehr, sehr originellen Computerbefehl zum Beispiel als Vornamen eingebe, dann kann sie mir möglicherweise Zugriff gewähren.

Und auf solche Fehler sind viele Softwareprogramme überhaupt nicht vorbereitet.

Da gibt es ja auch Untersuchungen darüber, wie viele Sicherheitslücken es gerade in öffentlichem Bereich gibt oder in medizinischer Software gibt.

Das ist alles Software, die ist dafür geschrieben, ihre eine Aufgabe zu erfüllen und das macht sie gut.

Aber die Programmierer waren nicht kreativ genug, an alle Möglichkeiten des Missbrauchs zu denken.

Und das sind aber Dinge, die wir dann mit unseren Tests sehr schnell finden und dementsprechend dann den Entwicklern die Gelegenheit geben, diese Fehler zu beseitigen, bevor jemand anders auf die Idee kommt, diese Fehler auszunutzen.

Ich habe gerade zum ersten Mal in meinem Leben verstanden, was ein Buffer Overflow ist.

Habe ich das Wort Buffer Overflow benutzt?

Nein, haben Sie nicht.

Sie haben Speicherüberlauf.

Aber so ein bisschen Englisch konnte ich dann doch noch.

Ich dachte, zumindest habe ich gerade gedacht, das ist ein Buffer Overflow, was

er da macht.

Naja, das ist ganz klassisch.

Jetzt, was mich wirklich, also warum hat das alles, was Sie da erfinden oder zu erfinden versuchen, warum hat das nicht längst schon mal jemand erfunden?

Das Problem ist doch nicht neu.

Das Problem ist neu insofern, als dass wir heute sehr viele Systeme haben, die ursprünglich in einer rein, in einer kontrollierten Umgebung genutzt wurden, wo also die, wo also es klar war, wer die Leute sind, die die Software bedienen, wo klar war, wo die Daten herkommen und wo man sich überall gegenseitig vertraut hat.

Also wenn eine Bank der anderen Banken Überweisung schickt und die Banken kennen sich, so, dann ist auch klar, ist auch klar, an wen ich mich wenden kann, wenn dabei irgendwas schief geht.

Durch die, das Gleiche gilt auch für Maschinen zum Beispiel.

Wir haben Druckmaschinen, alles Mögliche, die bisher in irgendwelchen geschlossenen Netzen operiert haben.

Und da gab es halt irgendwo, gab es halt eine Fernbedienung für diese Druckmaschine oder eine Konsole für diese Druckmaschine.

Und in dieser Konsole konnte ich alles Mögliche eingeben.

Und dann wurde die Druckmaschine dementsprechend bedient.

Und die Annahme, und dann kann man natürlich auch davon ausgehen, dass es da einen Angreifer gibt.

Nur ein Angreifer, der müsste sich ja dann physikalischen Zugang zu der Maschine verschaffen.

Und wenn er schon, wenn er schon einmal im Gebäude ist, dann kann er sowieso alles Mögliche machen.

Und dann ist das Ganze nicht mehr extra abzusichern.

Weil wenn er, dann kann er sich direkt an die Konsole setzen und loslegen, was auch immer unser Angreifer jetzt gerade vorhat.

Nur ist es aber so, dass immer mehr diese Systeme über Umwege oder auch direkt ans Internet angeschlossen werden.

Ich mache meine Bankgeschäfte selber.

Ich kann meine Visitenkarten online bestellen.

So ist es.

Und je mehr, und das bedeutet, dass sie ihre Kunden eben, dass sie ihr Gegenstück möglicherweise nicht mehr so gut kennen.

Aber die Software, die dort auf den entsprechenden Rechnern läuft, ist niemals mit dem Ziel geschrieben worden, dass sie sich auch gegen Angreifer verteidigen muss oder dass sie ungültige oder ungewöhnliche Eingaben ablehnen muss.

Und das ist durchaus, das ist durchaus ein, ja, ich will nicht sagen, eine Zeitbombe ist, aber es ist ein Problem, was wir halt immer und immer wieder sehen.

In aller Regel ist das so, dass wenn wir für ein, wenn wir für irgendein System zum Beispiel Testdaten erzeugen und die Firma kommt zu uns und sagt, wir würden gern mal, Herr Zeller, wir wissen, dass Sie daran forschen, das ist schön,

wir würden gern mal unser System in der Hinsicht testen.

Und dann frage ich sie, ja, wie sieht denn die Sprache aus?

Und dann geben sie mir eine Beschreibung der Sprache.

Und dann setzen wir uns hin und werfen unser System an und dann erzeugen wir einen Schwung Testdaten dafür.

Und dann schicken wir diese Testdaten an die Firma und die Firma sagt uns dann, ja, das ist uns reihenweise abgestürzt, vielen Dank auch.

[Lachen] Reden wir über bestehende Systeme oder auch über neue Entwicklungen?

Also sind neue Entwicklungen genauso wenig resilient oder begreifen moderne Entwickler inzwischen, was das Problem ist?

Bestehende Systeme, die von vornherein mit Anschluss ans Internet und so weiter gebaut wurden, sind meistens etwas resilienter.

Aber auch hier sehen wir, dass die Programmierer bisher immer ganz überwiegend vom, wir nennen das den Happy Path ausgegangen sind.

Hier kommen die Daten an und die Software tut das Richtige, wenn sie die richtigen Daten erhält.

Aber die Programmierer gehen nicht von den vielen, vielen Möglichkeiten aus, wie man diese Daten so gestalten und manipulieren kann, dass man da auch tatsächlich Fehler drin findet.

Und die Daten, die wir erzeugen, allein schon von der schieren Menge her, decken halt eine Vielzahl von Möglichkeiten, von Konfigurationen, von Optionen ab.

Wir können dort ja auch zum Beispiel bekannte Angriffsmuster einschleusen, wie zum Beispiel diese wahnsinnig langen Namen oder besondere Zeichensätze oder Gott, wenn ich ein Emoji, es reicht manchmal ja schon aus, wenn ich ein Emoji, also so ein grinsiges Gesicht in meinen Namen einsetze, dann stürzt die Software schon ab.

Also das geht dann ganz schnell.

Und dann ist natürlich die Frage, was bedeutet es, wenn die Software einen Fehler zeigt?

Wie schlimm ist dieser Fehler?

Also stürzt die Software einfach ab oder sagt sie, ich mache keine Funktion?

Hängt die sich auf oder fängt die möglicherweise an, dann die Daten noch an weitere Software weiterzuschicken?

Letztes Jahr im September hatten wir einen Fall, da war ich nicht daran beteiligt, ist auch gut so.

Da ist die Produktion von Volkswagen für einen Tag lahmgelegt worden, weil zwei Rechner im Rechnernetz von Volkswagen ein Datenpaket hatten und die nicht wussten, was sie damit machen sollten.

Und die haben sich das Fortwährend hin und her zugeschickt und waren deswegen für weitere Anfragen nicht mehr zu erreichen.

Da hat die Ping-Pong gespielt mit dem Datenpaket und waren dann beschäftigt.

Und da ist die Produktion von ganz Volkswagen für einen Tag stillgestanden.

Das ist ein, ich weiß nicht, ist das ein Millionenschaden?

Vermutlich.

Viele Millionen sind da reingegangen.

Das sind natürlich Sachen, hinterher kann ich sagen, pass mal auf Leute, ihr hättet vielleicht die Systeme im Hinblick auf diese Möglichkeit besser testen können.

Hättet ihr das getestet, hättet ihr mit einem Aufwand, hättet ihr das gekostet, ja, das hätte wahrscheinlich auch 100.000 gekostet, aber hättet damit einen Schaden von Dutzenden Millionen am Ende vermeiden können.

Und das ist halt immer so eine Risikoanalyse.

Sie sind da als Entwickler, sie sind da im Druck.

Die Firma will Geld verdienen, die Software muss raus.

Und der Kunde legt ja auch nur Wert darauf, dass die Software das tut, was sie soll.

Der Kunde kommt ja nicht her und sagt, habt ihr das Ding auch gegen alle möglichen Angriffe gut abgesichert?

Gut, vielleicht fragt der Kunde das und dann sagt der Entwickler, ja, wir haben ein bisschen was ausprobiert, das Gängige scheint zu funktionieren.

Aber der Aufwand, den man eigentlich dort betreiben müsste, der ist in Wirklichkeit nur automatisiert zu machen.

Und deswegen erforschen wir eben, wie wir das besser automatisiert machen können.

Sie sagten eben so beiläufig, ist der Fehler schlimm, ist er nicht so schlimm?

Ist nicht jeder Fehler schlimm?

Weil letztlich weiß ich ja gar nicht, wozu selbst der kleinste Fehler dann irgendwann noch mal führt, oder?

Ja, also wir unterscheiden da schon verschiedene Kategorien.

Es gibt zum Beispiel kosmetische Fehler.

Das ist irgendein Tippfehler auf ihrer Webseite oder so was.

Also ein Tippfehler im Text, der angezeigt wird.

Das ist ein bisschen peinlich für das entsprechende Unternehmen, wenn die die Regeln der Rechtschreibung nicht beherrschen.

Aber meinetwegen.

Und dann gibt es schwerere Fehler, zum Beispiel, wenn ihre Webseiten zum Beispiel von Leuten mit Sehbehinderung nicht gut erkannt werden.

Dann ist das nicht ein Fehler in dem Sinne, dass es inkorrekt ist, aber es ist etwas, das sie beheben müssen.

Auch weil sie zum Beispiel die entsprechenden Standards erfüllen müssen, gerade wenn sie im öffentlichen Dienst unterwegs sind.

Oder wenn sie zum Beispiel nicht etwa sehbehinderte Menschen als ihre Kunden verlieren möchten.

Ja, und dann gibt es natürlich echte Fehler, die tatsächlich dann dazu führen, dass das Geschäft nicht funktioniert.

Das heißt, jemand möchte gerne mit Ihnen bei Ihnen was kaufen, und das funktioniert dann nicht.

Und dann geht er natürlich woanders hin.

Oder aber, das kann natürlich auch passieren, dass Sie einen Fehler haben, der es Dritten ermöglicht, Zugang zu ihren Rechnern zu bekommen.

Und dann steht den Angreifern das ganze Repertoire an Schadensmöglichkeiten offen.

Zu unserer Zeit häufig ist das dann sogenannte Ransomware.

Das bedeutet, der Angreifer übernimmt Ihren Computer, blockiert den Zugang und sagt, Sie kriegen nur noch Zugang zu Ihrem Computer und den darauf gespeicherten Daten, wenn Sie an folgende Adresse folgende Summe in Bitcoins schicken.

Und erst dann gebe ich Ihnen den Geheimschlüssel, mit dem Sie wieder Zugang bekommen.

Und das ist natürlich dann das Schwerste, was passieren kann, weil dann, abhängig davon, wie gut Sie in der Lage sind, Ihre Systeme von einem Backup wiederherzustellen, kann es dann natürlich schon zu Produktionsausfällen oder möglicherweise auch praktisch zum kompletten Verlust Ihrer IT führen.

Und dann reden wir über Firmenpleiten, die daraus entstehen können.

Ist das, was Sie da machen, eigentlich Grundlagenforschung oder angewandte Forschung?

Das ist eine interessante Kombination aus einerseits Grundlagenforschung, weil diese Frage, wie man die Sprache beschreiben kann, mit der Computer

miteinander reden und wie man die entsprechenden Eigenschaften beschreiben kann und wie man auch zum Beispiel miteinander verknüpfen kann, dass auf diese Eingabe-Eigenschaften, diese Ausgabe-Eigenschaften folgen sollen.

Das ist ein etabliertes Gebiet der theoretischen Informatik, also ganz tief in der Grundlagenforschung.

Und wir bemühen uns auch, das ganz, ganz nah an den Verfahren der theoretischen Informatik auszurichten, damit wir da das Rad nicht neu erfinden und auch davon profitieren können, von den vielen Erkenntnissen, die theoretische Informatik dort schon eben theoretisch gebildet hat.

Dort werden nicht – theoretische Informatik – werden nicht irgendwo Kaffeetassen ausgetauscht, sondern einzelne Buchstaben, A, B, C oder sowas.

Weil die interessieren Kaffeetassen nicht, die interessieren auch Benchmarks nicht.

Aber wir ersetzen dann diese A, B, Cs durch Kaffeetassen, Lastschrift und Zustellungen mit Post oder was auch immer.

Und in dem Moment wird es dann eben angewandt.

Und dann richten wir wiederum unsere Testverfahren an den Anforderungen aus, die tatsächlich Entwickler in realen Programmen haben.

Bedeutet, wir bemühen uns natürlich schon damit, dass die Verfahren, die wir bauen, dass die nicht nur auf dem Papier funktionieren, sondern dass sie sich auch tatsächlich auf realen Systemen ohne viel Aufwand einsetzen lassen.

Weil wenn wir da eine große Hürde bauen, so nach dem Motto, wir haben hier dieses tolle Verfahren, aber um dieses Verfahren in deiner Firma einzusetzen, musst du erst mal Leute mit einem Doktorabschluss einstellen, die sich damit auskennen.

Die sind rar, das wäre schön.

Das wollen wir nach Möglichkeit vermeiden, sondern wir möchten, dass das auch von gewöhnlichen Entwicklern ohne viel Aufwand eingesetzt werden kann.

[Siebert] Ach, das heißt, ich hätte jetzt gedacht, vielleicht verkaufen Sie ja sogar auch die Dienstleistung.

Das wäre ja auch fürs CISPA super, wenn da noch ein bisschen Kohle reinkäme.

So kommt mal mit euren Systemen vorbei, wir checken die euch durch, kostet 100.000.

[Kehlmann] CISPA selbst ist nicht als Dienstleister in der Hinsicht unterwegs.

Die ist auch eine Institution der Grundlagenforschung, das heißt, wir machen die Forschung.

Aber CISPA ist auch daran interessiert, dass Forschung nicht nur auf dem Papier stattfindet, sondern auch hinterher zum Beispiel seinen Weg in entsprechende Firmen und Start-ups findet.

Und da sind wir in der Tat gerade dabei, ein entsprechendes Start-up aufzubauen, was dann Dienstleistungen um diese Forschung anbieten kann.

Wenn ein Kunde kommt und sagt, könnt ihr auch mein System mal testen, wenn wir dann sagen, also forschungsmäßig ist das jetzt das gleiche, was wir schon ein paar Mal gesehen haben, da lernen wir jetzt nichts Neues draus.

Aber wenn der Kunde dann eben interessiert ist und wenn der Kunde da Geld für bezahlen möchte, dann können wir das dann über die Mauer werfen und können das dann unserem Start-up anbieten.

Sie wollen die Automatisierung dieser Testverfahren einfacher machen.

Das klingt so, als wäre das ein Fass ohne Boden.

Gibt es irgendein Ziel, auf das Sie hinarbeiten?

Oder ist es eigentlich ein Prozess, in dem Sie bis zum Sankt-Nimmerleins-Tag stecken werden?

Aus der theoretischen Informatik heraus gibt es eine Erkenntnis, das ist das sogenannte Halteproblem.

Und das sagt, es ist nicht möglich, ein Programm zu bauen, was für ein anderes Programm vorhersagt, was es tun wird.

Unter allen Bedingungen.

Das heißt, es ist nicht möglich und es wird nie möglich sein, zum Beispiel ein vollautomatisiertes Programm zu bauen, was für jedes andere Programm sagt, das ist korrekt.

Das wird funktionieren.

Wir haben auch vorhin schon über die kombinatorische Explosion beim Testen gesprochen.

Das ist auch etwas, das kriegen Sie nicht weg.

Das ist ein grundsätzliches Problem.

Das heißt, es wird niemals möglich sein, ein Programm vollständig umfassend zu testen, automatisch.

Und es wird auch nicht möglich sein, den Akt zum Beispiel zu beweisen, dass ein

Programm immer und immer wieder das Richtige liefert.

Es wird auch nicht möglich sein, diesen Prozess so vollständig zu automatisieren, dass man es immer und immer wieder automatisch machen kann.

Und das bedeutet, dass es immer notwendig sein wird, dass sich Menschen oder wer auch immer da smart genug ist, mit dem Programm beschäftigen und zum Schluss sagen können, das ist jetzt gut genug, das Programm ist richtig und das Programm erfüllt seinen Zweck.

Und das Programm, neben dem, dass es seinen Zweck erfüllt, hat auch keine sonstigen unerwünschten Seiteneffekte.

Und diese Unvollständigkeit ist aus Forschungssicht einerseits natürlich sehr unbefriedigend, weil wir werden also niemals fertig damit.

Es kann immer wieder kompliziertere Programme geben, für die wir uns neue Verfahren ausdenken müssen, die dann einen Teil dieser Komplexität abdecken können.

Auf der anderen Seite bedeutet das auch, dass man als Forscher niemals aufhört.

Man kann in einem Programm relativ leicht, wie wir sagen, die niedrig hängenden Früchte finden.

Das heißt, Sie können relativ leicht die Fehler finden, die sofort ins Auge stechen.

Sie können schnell nachweisen, dass ein Programm im Hinblick auf bestimmte Eigenschaften Fehler zeigt.

Also es gibt immer eine Reihe von einfachen Fehlern, die Sie schnell finden können.

Die sind auch in Programmen heute weitgehend abgearbeitet.

Dann gibt es kompliziertere Fehler, wo Sie einen immer größeren Aufwand treiben müssen, um diese Fehler zu finden.

Und da versuchen wir mit unserer Arbeit, sehr viel tiefer in diese Domäne vorzudringen.

Ja, und dann gibt es Fehler, wie soll ich sagen, da gibt es Fehler, von denen wir nur vermuten können, dass sie wahrscheinlich irgendwo sind, aber wir wissen nicht, wie sie aussehen.

Wir wissen nicht, was sie sind.

Das sind dann die "unknown unknowns".

Ich glaube, Donald Rumsfeld hat es mal so konnte.

Also wir haben die Sachen, die sind bekannt, die sind unknowns.

Das ist bestimmt herrlich.

Das ist geregelt.

Wir haben die "known unknowns", das heißt, wir wissen, hier gibt es möglicherweise Fehler, aber wir haben eine Idee, wie wir darauf stoßen können.

Und dann gibt es die "unknown unknowns".

Das sind Fehler, die wahrscheinlich irgendwo sitzen, aber wo wir bis jetzt noch überhaupt keine Idee haben, wie die eigentlich aussehen.

Die einzige gute Nachricht ist, die Angreifer wissen das hoffentlich auch nicht.

Und solange es weder die Angreifer noch die Verteidiger wissen, dass da möglicherweise eine Schwachstelle ist, dann ist es kein Problem in der Praxis.

Die Frage ist, wer findet zuerst diese "unknown unknowns"?

Wenn es die Angreifer zuerst finden, dann müssen wir uns schnell überlegen, wie wir diese Fehler beseitigen können.

Ein Ende ist nicht in Sicht, aber es wird ja irgendeinen Anfang gegeben.

Aber erinnern Sie sich daran, wie Sie in genau diesen Forschungsbereich gekommen sind?

Gibt es so diesen Moment, wie diese scheiß Büroklammer immer wieder aufpoppt, das muss da weg.

Und dann sind Sie losgelaufen, um es zu fixen?

Ich bin eigentlich über Umwege dahin gekommen.

Ich habe angefangen mit meiner Forscherkarriere eigentlich mit Verfahren, was der automatischen Fehlersuche bedeutet.

Mein Programm tut nicht das, was es soll.

Wie kann ich das möglichst gut herausfinden?

Das ist ein ziemlich unbeleuchtetes Gebiet in der Informatik.

Leute reden nicht gerne über ihre Fehler.

Und wenn Programmierer eine lange Nachtsitzung machen, um an ihrem Programm zu arbeiten, ist es meistens, weil sie irgendeinem Fehler auf der Spur sind und versuchen herauszufinden, was genau die Ursache ist.

Als ich in den Zwanzigern war, habe ich auch häufiger solche Nachtsitzungen gemacht und habe mich immer geärgert, wieso muss ich hier die ganze Nacht sitzen?

Das lässt sich doch irgendwie automatisieren lassen.

Und habe ich dann auch Verfahren entwickelt, mit denen man wirklich ganz wunderschön solche Fehlermengen eingrenzen kann, dass man herausfinden kann, der Fehler tritt genau auf, wenn folgende Bedingung eintrifft.

Also der Fehler tritt immer dann genau auf, wenn man eine Kaffeetasse erst kauft, dann aus dem Warenkorb wieder rausnimmt und dann per Lastschrift bezahlt, aus irgendwelchen Gründen.

Und der tritt auch nur genau unter diesen Bedingungen auf.

Und sowas kommt voll automatisch raus.

Das ist schön.

Aber die Voraussetzung dafür war, um sowas automatisch herauszufinden, war, dass mein System selbst Experimente durchführen konnte.

Das heißt, das System ist zum Schluss einfach selbstständig hergegangen und hat gesagt, dann probiere ich es mal mit etwas anderem als einer Kaffeetasse.

Aha, mit einem Klavier, interessant.

Okay, dann tritt der Fehler auch auf.

Dann probiere ich es hiermit, dann probiere ich es damit.

Aha, damit tritt der Fehler nicht auf.

Offensichtlich scheint Lastschrift wichtig zu sein.

Also ähnliche Experimente, wie sie auch einem Spurensucher, der dann deduktiv vorgeht, mit einem Experiment nach dem anderen herangeht.

Und dann konnte ich solche Fehlerbedingungen gut eingrenzen.

Das war schön.

Aber die Voraussetzung dafür war, dass der Test automatisch ablaufen konnte.

Erst dann konnte man auch automatisch Fehlersuche machen.

Und ja, dann bin ich auf die Frage gestoßen, wie kann ich den Test besser automatisieren?

Und dann hatte ich noch ein sehr viel größeres Problem an der Backe.

Weil wenn Sie ein Programm haben, das einen Fehler zeigt, dann muss es irgendwo eine Ursache für diesen Fehler geben.

Das ist unvermeidlich.

Und dann ist es eigentlich nur eine Frage der Zeit, dass Sie diese Ursache finden.

Und dann haben Sie die Ursache gefunden und Sie sind glücklich.

Wenn Sie aber ein Programm testen wollen und zeigen wollen, dass es frei von Fehlern ist, ja, dann hört das eigentlich nie auf.

Und Sie können immer noch besser werden und immer noch besser werden und immer smarter werden.

Und Sie schieben da die Qualität Ihres Programms, Sie verbessern die Qualität Ihres Programms immer weiter.

Aber diese Erlösung, dass Sie den Moment haben, jetzt weiß ich, es kann nichts mehr schiefgehen.

Die haben Sie in Wirklichkeit nie.

Sie machen das Risiko immer kleiner, aber Sie kommen niemals bei null an.

Und das ist etwas, ja, das finde ich leider, das ist etwas unbefriedigend.

Auch von der Forschung her.

Auf der anderen Seite, wie schon gesagt, es sorgt dafür, dass Sie immer weiter forschen können und forschen müssen.

Und das alles, was Sie mir jetzt erzählt haben, wie haben Sie das dann in den Antrag geschrieben, wo Sie die zweieinhalb Millionen für gekriegt haben?

Sie haben fünf Seiten und zehn Minuten.

Wow.

Das heißt, Sie müssen in fünf Seiten erklären, was Sie vorhaben.

Das ist die Kurzfassung des Antrags.

Müssen Sie auch erklären, was das Problem ist?

Ja, selbstverständlich.

Oder die Leute, die da sitzen, verstehen auch das Problem?

Selbstverständlich.

Im Grunde sitzen da so Typen wie ich.

Genau.

Wow.

Ein bisschen mehr vom Fach, aber auch die Informatik hat natürlich eine große Bandbreite an Themen.

Und dann müssen sich dann auch Leute aus der Robotik zum Beispiel mit Fragen der Visualisierung beschäftigen oder mit effizienten Algorithmen.

Das sind alles Gebiete, die relativ weit voneinander entfernt sind.

Das ist ein bisschen so, wie wenn Sie, sagen wir jetzt, englische Literatur studiert haben, aber jetzt kommt jemand aus der vergleichenden Kunstwissenschaft oder was auch immer.

Ja, dem müssen Sie dann erst mal erklären, worin das Problem eigentlich besteht.

Und die besondere Kunst besteht nicht darin, es Ihren Fachkollegen zu erklären.

Die verstehen, was Sie sagen.

Die verstehen, was das Problem ist.

Und da gibt es auch einen zweiten Teil des Antrags, der geht an die Fachexperten.

Und da können Sie dann sozusagen mit Ihresgleichen reden.

Das ist vergleichsweise einfach, aber die Herausforderung besteht in diesen fünf Seiten und in diesen zehn Minuten, wo Sie versuchen müssen, das Problem und Ihre Lösung und warum das cool ist und warum das gefordert werden muss, tatsächlich einem relativ breiten Publikum zu erklären, um die zu motivieren, damit sie nicht gleich in der ersten Runde herausfliegen.

Und das ist etwas, da habe ich dann sehr viel Zeit reingesteckt, das zu perfektionieren, indem ich mit vielen Kollegen, auch die weit außerhalb meines Gebiets waren, geredet habe und habe denen versucht zu erklären, wie es funktioniert.

Und habe dabei auch gelernt, das haben wir jetzt ja auch schon früher in dem Gespräch festgestellt, Sie können da gar nicht einfach genug sein.

Sie müssen da wirklich hergehen und sagen, ich erkläre das jetzt auf die einfachstmögliche Art und Weise, sodass es wirklich jeder versteht, damit, wenn es zum Schluss dann zum Schwur kommt, dann ist es völlig in Ordnung, wenn Ihr Antrag abgelehnt wird, weil es bessere Anträge gab.

Das ist in Ordnung.

Das ist sportlich so, das gehört so.

Sie haben die Auslese der Besten.

Wenn es einen besseren Antrag gibt, dann muss der genommen werden, dann müssen Sie zurückstehen.

Das ist ganz normal, das gehört zu den Spielregeln.

Aber wenn Ihr Antrag abgelehnt wird, weil ein Gutachter oder das Panel, was Ihr Antrag anschaut, weil die nicht verstanden haben, worum es eigentlich geht, weil die nicht verstanden haben, wo das Problem ist oder wenn die nicht verstanden haben, worin die Lösung besteht, das ist sehr ärgerlich.

Und deswegen habe ich da viel Arbeit reingesteckt, das so zu erklären, dass es einigermaßen, zumindest für Fachleute, leicht verständlich ist.

Aber wie haben Sie denn die Lösung formuliert, weil es gibt doch gar keine?

Die Lösung habe ich so formuliert.

Ich habe mir gedacht, hier ist ein Beispiel und wenn man das alles so macht, wie ich mir das so vorstelle, dann wäre die Lösung zum Schluss diese hier.

Und diese Lösung und dieses Beispiel war einfach genug gehalten, um das tatsächlich in wenigen Minuten nachvollziehen und verstehen zu können.

Und dann habe ich gesagt, gut, jetzt habe ich das an diesem Beispiel demonstriert.

Und wenn wir das jetzt ganz groß machen, dann habe ich all diese großen Probleme vor mir.

Und wie ich diese großen Probleme zu löse gedenke, das führe ich jetzt hier nicht weiter aus.

Das steht im zweiten Teil des Antrags.

Aber Sie haben ja gesehen, dass es auf dem Beispiel funktioniert.

Also bitte geben Sie mir jetzt das Geld.

So ungefähr war die Argumentation.

Jetzt hat das aber doch ein offenes Ende.

Das heißt, Sie haben die Lösung aus diesem Antrag, also praktisch die

Zwischenlösung, wenn man so will, das Problem haben Sie jetzt gelöst.

Und dann müssen Sie aber weiter forschen, weil das Problem nie aufhört zu bestehen.

Das heißt, in fünf Jahren gehen Sie wieder und sagen, hier, gib noch mal zweieinhalb.

Dann fühlen Sie sich doch verarscht.

Es war ja sowieso schon mein zweiter Antrag und deswegen wünsche ich sowieso schon damit Leben, dass möglicherweise die schöne Frage kommt, Sie haben doch schon mal so viel Geld bekommen.

Wie soll es damals nicht funktionieren?

Und, haben die gefragt?

Nein, haben sie glücklicherweise nicht gefragt, aber ich hatte da natürlich auch eine Antwort dazu.

Und dann hat man gesagt, ja, wir sind es damals zwar schon gut angegangen, aber noch zu naiv.

Und jetzt sind wir sehr viel besser.

Jetzt haben wir es sehr viel besser verstanden.

Aber was Sie dann machen, ist, Sie sagen dann, hier, ich habe dieses Beispiel gelöst.

Und dann schwelgen Sie darin, was daraus alles möglich wird.

Aber dann, bevor Sie dann zu sehr schwelgen, müssen Sie dann wieder so

rhetorisch runtergreifen und sagen, machen Sie ein Erwartungsmanagement und sagen, ja, jetzt haben wir dieses kleine Beispiel gelöst.

Aber es gibt doch folgende Riefen, wenn wir das ins Große gehen.

Dann gibt es jede Menge Risiken und jede Menge Probleme, die alle noch auftreten sollen.

Die führen Sie dann noch auf.

Aber ich bin zuversichtlich, ich, ich bin ja hier der Antragsteller, ich bin ja der Supermann.

Ich bin zuversichtlich, dass wir das lösen können, weil ich bin ja ich und ich bin der Beste, zumindest in Europa, der das machen kann.

Das schreiben Sie, so etwas Ähnliches schreiben Sie da rein.

Also wenn jemand das lösen kann, dann höchstwahrscheinlich nur der Antragsteller selbst.

Und auf diese Art und Weise können Sie dann die Erwartungen einigermaßen in den Griff bekommen und auf Ihre Person und auf die Annahme des Antrags lenken.

Wissen Sie jetzt schon, dass Sie nach diesen fünf Jahren noch mal einen Antrag stellen werden?

Es ist schon frech genug, eigentlich, diesen, sich zum zweiten Mal zu bewerben.

Ich glaube nicht, dass ich einen dritten Antrag, einen dritten Antrag stellen werde.

Ich würde es auch so sehen, wenn ich einen dritten Antrag stelle, dann sollte ich

vorher wirklich gut nachweisen, dass der zweite Antrag erfolgreich war und dass daraus ganz viele ganz tolle neue Möglichkeiten entstehen.

Aber ich denke, wenn der zweite Antrag erfolgreich ist, dann sollte sich das nicht in einem großen weiteren öffentlichen Forschungsprogramm ausdrücken, sondern tatsächlich in Firmen, die das Ganze einsetzen und die das Ganze auch dann mit Erfolg einsetzen und die auf diese Art und Weise auch der Gesellschaft dann das zurückgeben, was die Gesellschaft vorher in diese Forschung investiert hat.

Andreas Zeller, vielen Dank.

War mir ein Vergnügen.

[Musik] [Ende]