Softaware

Ein Plädoyer für Bewusstseinserweiterung bei der Zusammenarbeit von Mensch und Maschine

Dr. Gero Scholz

Nein, der Titel dieses Beitrags enthält keinen Schreibfehler. Wenn Sie ahnen, was das eingeschobene „a“ bedeuten soll, dann sind wir bereits mitten im Thema, denn Sie haben eine Leistung erbracht, die einem Menschen gelingen kann, für eine Maschine jedoch eine erhebliche Hürde darstellt: Sie haben erkannt, dass es sich um ein Wortspiel handeln könnte, um eine Kombination aus **Soft**ware und **Aware**ness (Aufmerksamkeit, Wahrnehmung, Selbstreflexion). Im Untertitel ist von *Bewusstseinserweiterung* die Rede. Das passt auch zu dieser Interpretation des Titels.

Softwaresysteme unterstützen uns in vielfältiger Weise. Sie steuern unsere Heizung, helfen uns bei der Suche im Internet oder beim Einparken des Autos. Neuerdings dienen sie sich uns als Junior-Partner an: „Alexa, spiel meine Lieblingsmusik!“. Dabei kommt ihnen unsere angeborene Neigung zur Vermenschlichung von Technik entgegen, zur Vermutung von planvollem Handeln und zur Unterstellung von Absichten oder Wertvorstellungen.

Wir müssen Prinzipien für die Überschneidungsbereiche der Handlungsfelder von Mensch und Maschine festlegen. Die zentrale These dieses Beitrags ist, dass die weitere Annäherung von Mensch und Maschine am besten gelingen wird, wenn beide Seiten mehr voneinander wissen, also ein *Bewusstsein für ihr Gegenüber* entwickeln. Als *Softaware* bezeichnen wir im Folgenden ein Computerprogramm, das sich genau darum bemüht. Das Gegenstück dazu ist der Mensch, der einen bewussten Kontrakt mit der Technologie schließt, klar umrissene Erwartungen an die Kompetenzverteilung hat und deren Einhaltung überwacht.

Spätestens seit der Erfindung der Fließbandfertigung im Automobilbau wissen wir, dass die Sicht der Menschen auf die Maschine nicht einheitlich ist: Henry Ford persönlich hat sich ganz bestimmt als Schöpfer einer mächtigen Produktionsmaschine erlebt, aber die Arbeiter am Band waren eben nicht die Schöpfer, sondern die Räder im Getriebe. Es kann eine riesige Kluft liegen zwischen den wenigen, die ein System entwerfen oder betreiben und den vielen, die ihm ausgeliefert sind. Dies umso mehr, als allgegenwärtige Software längst nicht so einfach zu erkennen ist wie ein Automobilwerk. Oder wissen Sie, von wie vielen Überwachungskameras sie heute schon erfasst wurden? Haben Sie getankt, waren Sie im Einkaufs-Center?

Die drei Eingangsbeispiele (*Heizungssteuerung, Suchmaschine, Einparkhilfe*) stehen stellvertretend für drei Situationen und die entsprechende *Haltung des Menschen gegenüber der Maschine*.

Maschinen, die uns bei kleineren alltäglichen Regelungsaufgaben entlasten (Backofen, Raumheizung), nehmen wir als *Underdog* wahr. Suchmaschinen, soziale Systeme und Überwachungseinrichtungen erleben wir anders: Je nach Veranlagung und (Leidens-)Erfahrung

sehen wir sie als Spielwiese an, als notwendiges Übel oder erleben sie als Bedrohung, als *Big Brother.* Die dritte Kategorie bilden Softwaresysteme mit dem Anspruch, Helfer auf Augenhöhe zu sein, die *Companions*. Sie werden uns in Zukunft immer öfter begegnen.

Underdogs

Eine Heizungssteuerung fällt in die Kategorie „nützliche Technik, die eine klar definierte Aufgabe erfüllt". Als „Fachidioten" sind diese Systeme für uns emotional uninteressant und auch nicht bedrohlich, weil wir eine klare Vorstellung von ihren *Kompetenzgrenzen* haben. Wir müssen nicht genau verstehen, wie sie funktionieren, um sie zu benutzen und richtig einordnen zu können.

Wir müssen allerdings ihren Zweck und ihren Wirkungsbereich verstehen, insbesondere auch ihr Schadenspotential. Eine Ölheizungssteuerung ohne Flammenwächter könnte dazu führen, dass die Einspritzpumpe den Inhalt des Öltanks genüsslich in alle Kellerräume entleert. Das Einlassventil für einen Gasbrenner sollte „eigensicher" sein, also im Fall eines Stromausfalls bitteschön durch mechanischen Federdruck sofort schließen. Wenn ein Temperatursensor defekt ist, dann kann es vielleicht passieren, dass eine Anlage dauernd mit voller Kraft heizt. Das ist nicht gefährlich, aber ein deutliches Versagen wäre es doch. Gute *Softaware* sollte nicht blind den aktuellen Sensorwert in Schaltbefehle für den Brenner umrechnen. Sie sollte sich ihrer Sensoren und Aktoren bewusst sein und davon ausgehen, dass sie fehlerhaft sein können. Eine einfache Gegenmaßnahme besteht darin, mehrere Sensoren zu benutzen und deren Werte zu vergleichen. Hat die Anlage dennoch nur einen einzigen Außentemperaturfühler und stellt fest, dass dieser acht Stunden lang exakt 6,3° Celsius anzeigt, dann wird sie misstrauisch und informiert den Benutzer. Gute *Softaware* ist zu dieser Art von Selbstreflexion in der Lage. Sie beobachtet sich selbst, führt ein Logbuch über Auffälligkeiten und wechselt im Zweifelsfall in einen Störmodus.

Ist die Bedienung eines *Underdog* Systems kompliziert, so entwickeln wir Ärger oder Fatalismus. Schlecht konzipierte Systeme konfrontieren uns mit einem Funktionsbaukasten und einer Fülle von Einstellknöpfen. Etwa für Vorlauftemperatur oder „Steilheit der Regelungskurve". Sie verlangen uns ab, präzise Zeitfenster für die Dauer der „Nachtabsenkung" an jedem Wochentag festzulegen, wissen aber nicht einmal, was Feiertage sind. Schlechte Software ist aus der Sicht des Konstrukteurs der Anlage konzipiert, der es am liebsten hat, wenn der Benutzer das Fachvokabular eines Heizungsbauers beherrscht.

Gute *Underdog Softaware* weiß, dass sie ein Gebäude in einem Temperaturbereich halten soll, der den Bedürfnissen seiner menschlichen Bewohner entspricht. Sie erkundigt sich, ob man gern kühl schläft, abends baden will, ob die Kinder noch in einem Alter sind, wo sie sonntags früh um sieben putzmunter sind, ob man im Schichtdienst arbeitet. Sie findet selbst heraus, auf welchem Breitengrad des Globus das Gebäude steht, ob es Zeitumstellungen gibt und kann Kontakt mit lokalen Wetterprognosen aufnehmen. Bei aller Empathie macht sie ihre Annahmen transparent und legt vor allem die Schlussfolgerungen offen, die sie daraus zieht, etwa, wenn sie einen avisierten Wetterwechsel berücksichtigt. Sie verbessert ihre Regler-Charakteristik, wenn wir ihr mit qualitativen Hinweisen helfen („Gestern Nachmittag war es im Wohnzimmer zu warm"), d.h. sie kann sogar mit unscharfen Aussagen etwas anfangen. Sie kann feststellen, ob und wann sich in bestimmten Räumen normalerweise Menschen aufhalten und das erworbene Wissen nutzen, um einerseits Energie zu sparen, andererseits aber auch Räume rechtzeitig im Voraus aufzuheizen. Gute *Softaware* versucht mit der Art und Qualität von Informationen zurecht zukommen, die Menschen als problemadäquat ansehen und die sie auch einem menschlichen Butler gegenüber verwenden würden, der sich für sie um die Heizung ihrer Wohnräume kümmert.

Gute *Softaware* bietet für absehbare Sonderfälle unterschiedliche Eskalationsebenen an und vereinbart vorab mit dem Benutzer das gewünschte Verhalten.

> Was soll passieren, falls es zu einer technischen Störung an der Anlage kommt, die durch einen automatischen Neustart nicht behoben werden kann?
> Bitte wählen Sie aus (Mehrfachauswahl möglich):
>
> > Störmeldung im Display zeigen, Warnton ausgeben
> > Email / SMS an den Benutzer versenden
> > Email an den Wartungsbetrieb versenden (allgemeiner Hinweis)
> > Email an den Wartungsbetrieb versenden (mit Autodiagnose-Daten)
> > Dem Wartungsbetrieb unmittelbar Online-Zugriff auf die Anlage geben

Dem Status einer *Underdog Softaware* ist eine sachlich-neutrale Sprache angemessen. Sie bezeichnet sich im Benutzerdialog selbst als „Die Anlage“, abstrahiert dabei auch von ihrem Markennamen, nennt sich also nicht „Ihre neue XYZ-thermatic“.
Ein Benutzerdialog mit der Frage „Darf ich gleich einen Terminvorschlag beim Wartungsbetrieb einholen, wenn ich merke, dass ich kaputt bin?“ wäre unangemessen. Anbiederung ist für *Underdogs* nicht zulässig.

Gute *Underdog Softaware* legt Wert darauf, den menschlichen Benutzer stets informiert zu halten, wenn sie auf Basis einer getroffenen Vereinbarung autonom agiert. Sie wird also eine Kopie der Störfall-Nachricht an den Benutzer senden, die automatisch an den Wartungsbetrieb hinausgegangen ist.

Sie darf ein klein wenig Werbung in eigener Sache anbieten, etwa, indem sie einen Link zu einer (sicheren!) Webpage anbietet, auf der man den Temperaturverlauf und den Energieträgerverbrauch im Zeitverlauf nachvollziehen kann. Sie kann sogar einen anonymisierten Vergleich mit anderen Anlagen des gleichen Herstellers (Benchmarking) anbieten. Sie kann emittierte Schadstoffmengen abschätzen und ausrechnen, wie viele Bäume man zur Kompensation pflanzen müsste. Aber nur, wenn sie dies mit ihrem öko-orientierten Benutzer vereinbart hat. Normalerweise verbieten sich moralische Zeigefinger für *Underdogs*.

Was passiert, wenn ein *Underdog* Begehrlichkeiten entwickelt? Malen wir uns folgendes Szenario aus: Der Heizungsbauer bietet uns an, dass er für 500€ eine technische Veränderung einbauen könnte, die sich bereits nach fünf Jahren amortisiert. Erzählt er etwas von verbesserten Einspritzdüsen, die zu gleichmäßigerer Verwirbelung führen und für bessere Verbrennung sorgen? Damit haben wir ganz bestimmt kein Problem. Oder spricht er etwa davon, dass er ein Software-Update einspielen würde, welches unseren Google-Kalender mitliest, Abwesenheitszeiten erkennt und zum Energiesparen nutzt? Außerdem möchte die neue Software selbst festlegen, bei welchem Provider die Primärenergie eingekauft wird und wann frisches Öl bestellt wird. Sicherheitshalber möchte sie dazu auch die Liquidität unseres Girokontos abfragen...

Es kann sein, dass die Hersteller von *Underdog Softaware* in den Club der *Companions* aufsteigen möchten. Dabei ist Vorsicht geboten. Das Vertrauensverhältnis zum *Underdog* beruht darauf, dass man seinen Wirkungsbereich einschätzen und definitiv abgrenzen kann. Die Sensibilität der Menschen gegenüber „invasiver Software“ wird weiter steigen. Und manchem Besitzer eines technischen Geräts wird es ein Vergnügen sein, dessen Kooperationsgelüste zu unterbinden.

Big Brother

Wenn wir unser Handy mit einem Fingerabdruck entsperren, dann sieht das zunächst einmal nach einer weiteren typischen *Underdog*-Bequemlichkeit aus. Wir könnten ja auch eine PIN verwenden oder ein Muster zeichnen. Vertrauen wir darauf, dass der Gerätehersteller unseren Fingerabdruck

verschlüsselt im Gerät speichert? Er könnte ihn auch als Bild zusammen mit unseren Personendaten und einem Foto von uns an den Geheimdienst seines Landes weitergeben. Wenn wir zwei Geräte vom selben Hersteller haben, finden wir es vielleicht praktisch, den Fingerabdruck „zwischen den Geräten zu teilen“. Die Daten würden dabei zwischendurch zwangsläufig auf einem Server des Herstellers landen und wären dort zumindest für dessen technische Administratoren zugänglich, potentiell auch für Hacker und – je nach den AGB, die wir ignoriert haben – auch für seine Tochtergesellschaften und „befreundete Unternehmen“...

Mit schärferen Sinnen als früher erkennen wir heute, ob ein Softwaresystem das Potential hat, sich für uns zu einer Bedrohung auszuwachsen. Mit der Kategorisierung als *Big Brother* tragen wir diesem Potential Rechnung – wir unterstellen nicht automatisch, dass die Systeme böswillig sind.

Gute *Big Brother Software* hat eine Vorstellung davon, dass sie Menschen dient, die elementare Bedürfnisse nach Sicherheit und Privatheit haben, jedoch bereit sind, manche davon zurückzustellen zugunsten der Kommunikation mit anderen Menschen, die ihnen mehr oder weniger vertraut sind. Die *Softaware* versteht diesen Prozess der Güterabwägung und orientiert ihr Verhalten daran. Sie verwendet daher die Fingerabdruck-Erkennung in der Grundeinstellung ohne unsere explizite Zustimmung gar nicht. Sie erbittet allerdings im passenden Moment den Mindestumfang an Berechtigungen, den sie benötigt, wenn wir die Funktion nutzen wollen (also: Fingerabdruckdaten verschlüsselt auf dem Gerät speichern, nie nach außen geben). Sie enthält im Idealfall sogar ein Verfallskonzept (gespeicherte Fingerabdruckdaten sechs Monate nach der letzten Benutzung löschen). Gute *Softaware* verwendet ein allgemein bekanntes Verschlüsselungsverfahren zur Speicherung des Fingerabdrucks; insbesondere speichert sie nicht die grafische Kontur, sondern nur einen Zahlenwert (Hash), der sich aus einer mathematischen Formel ergibt, in welche die Bilddaten einfließen. Aus dieser Zahl kann man nicht rückwärts das Bild ermitteln, sondern man kann lediglich feststellen, ob ein anderes Bild dem ersten Bild so ähnlich ist, dass die mathematische Formel denselben Zahlenwert produziert.

Um zu beweisen, dass die Software des Herstellers dies alles tut, veröffentlicht ein gutwilliger *Softaware*-Anbieter den Quellcode und lässt sein System von unabhängigen Experten zertifizieren. Ein böswilliger oder unfähiger Hersteller spricht natürlich ebenfalls vollmundig von hohen Sicherheitsstandards, veröffentlicht den Quellcode wenn überhaupt jedoch nur in Auszügen, baut Hintertüren für den nationalen Geheimdienst ein und sendet unter dem Vorwand Daten an seinen zentralen Server, dass „gewisse Stichproben zur Verbesserung der Servicequalität“ unerlässlich seien. Vielleicht ist er aber einfach nur inkompetent, beschäftigt dilettantische Software-Entwickler oder verwendet ahnungslos Softwarekomponenten, die Angriffspotential für Hacker bieten.

Viele Webdienste finanzieren sich durch Werbung, weil keiner von uns gewillt ist, für die Kernleistung Geld zu bezahlen. Je mehr sie von uns wissen, desto zielgenauer wird ihre Werbung. In Idealfall bringt sie dem Unternehmen dann mehr Geld und ist für uns weniger lästig als ungezielte Werbung. Durch die Nutzung werbefinanzierte Websysteme akzeptieren wir diesen Deal – fühlen uns aber doch recht unbehaglich, insbesondere, wenn mehrere solche Dienste auch noch fusionieren oder ihre Daten über uns anderweitig vernetzen.

Gute *Softaware* geht nach dem „need to know“-Prinzip vor und verlangt nur Informationen, bei denen wir unmittelbar einsehen, dass sie zur Erfüllung der primären Systemleistung (oder zu ihrer Finanzierung!) erforderlich sind. Wenn ich ein Programm zur QR-Code-Erkennung auf mein Handy herunterlade, dann benötigt es ganz offensichtlich Zugang zur Kamera. Stand da eventuell „Zugang zu Kamera und Mikrofon?“ Vielleicht ist es gar nicht möglich, das so genau zu unterscheiden? Mit der Handykamera ist ja auch das Mikrofon gekoppelt, etwa wenn man Videos damit macht. Böse *Big Brother* Software könnte dauerhaft alle unsere Gespräche mithören, sobald wir einmal einen QR-Code gescant haben.

Wir müssen auf Marktmechanismen vertrauen; darauf, dass ein solcher Betrug von irgendeinem technisch versierten „guten Hacker“ irgendwann aufgedeckt werden würde. Solche Fälle gibt es bereits. Einige besonders günstigen Handymodelle brachten in der Vergangenheit ab Werk „Spyware“ mit. Ob deren nachträgliche Entdeckung die Käufer wohl jemals erreicht hat?

Wenn wir akzeptieren, dass unsere Gesellschaft einen berechtigten Anspruch auf Schutz gegenüber solchen Praktiken hat, dann sollten wir Zertifizierungsverfahren für *Big Brother Softaware* festlegen und unabhängige Institutionen mit der Prüfung beauftragen, bevor Produkte in den Markt kommen. Aber kann das jemals gelingen?

Bei Geräte-bezogener Software mag ein solches Verfahren noch vorstellbar sein – auch wenn entsprechende Zertifizierungen vermutlich erst nach der jeweiligen Markteinführung erfolgen könnten und daher nicht als Zulassungsbarriere wirken würden. Aber viel sensibler noch als die einzelnen Geräte sind zentrale Softwaresysteme, welche große Kommunikationsnetze oder Industrieanlagen steuern. Sie von Amts wegen kontrollieren zu wollen, ist utopisch. Manchmal wird argumentiert, dass zumindest im „Business-to-Business“-Bereich sich halbwegs gleichwertige Partner gegenüber stünden. Schließlich seien ja auch die Einkäufer solcher Systeme Unternehmen mit entsprechender Kompetenz. Darf man also auf die Mündigkeit und technische Professionalität der Beteiligten vertrauen? Kann beispielsweise ein Verkehrsunternehmen, das seine öffentlichen Verkehrsmittel mit Sicherheitstechnik ausstatten möchte, wirklich verhindern, dass es eine Software-Lösung bekommt, die nebenbei noch anderen Zwecken nachgeht? Vielleicht bei der initialen Abnahme dieser Software, aber wie sieht es bei Updates aus?

Gute *Softaware* muss offenlegen, über welche Schnittstellen sie in welchen Fällen wohin Kontakt aufnimmt. Bösartige Systeme, insbesondere natürlich Viren, sammeln lange Zeit unbemerkt Informationen und übertragen sie dann verschlüsselt an ihre Leitzentrale. Das Zeitalter, in dem sie sofort eine Schadensfunktion ausführten („Ihre Festplatte wurde soeben verschlüsselt …), geht zu Ende. Die dauerhaft unbemerkte Unterwanderung ist die weit größere Bedrohung. Wir sollten daher von zertifizierter *Big Brother Softaware* verlangen, eine konfigurierbare *Monitor-Schnittstelle* anzubieten, über die der Betreiber oder ein von ihm beauftragter Sicherheitsinspektor Filterkriterien festlegen kann, um auffällige Kommunikationsvorgänge automatisch zu erkennen, zu überprüfen und im Zweifelsfall zu unterbinden.

Kehren wir nochmals kurz zum klassischen Fall zurück: Individuum gegenüber zentralem System (Business to Consumer). Wer soll *Facebook* überprüfen? Ein nationaler TÜV? Eine internationale NGO? Eine Kommission der Vereinten Nationen? Woher sollte eine solche Einrichtung die notwendige fachliche Kompetenz nehmen? Selbst wenn der Hersteller kooperativ wäre, würde die Prüfung Monate dauern. Monate, in denen zahlreiche Funktionen schon wieder verändert werden.

Wenn Kontrolle so schwierig ist, kann man dann nicht vielleicht wenigstens an das Gute im Menschen glauben, an das persönliche und berufliche Ethos der Entwickler von *Big Brother Softaware*? Leider etwas zu romantisch und naiv, insbesondere, in Ländern, bei denen der Staat Zugriffe auf Unternehmen durch nationale Interessen rechtfertigt.

Die aktuelle Entwicklung in China zeigt, dass ein Staat, der sich wohlmeinend gibt, mindestens so bedrohlich sein kann, wie ein gewalttätiger Überwachungsstaat. Wenn Eltern von Schulkindern kranke Angehörige pflegen, ist es dann nicht fair, wenn ihre Kinder in eine Schule mit besonders guten Lehrern gehen dürfen, damit wenigstens das Thema Hausaufgaben-Unterstützung nicht zur Zusatzbelastung wird? Viele von uns halten eine solche Forderung vermutlich für legitim. Wenn man aus solchen Überlegungen heraus ein System der „social credit points“ schafft, – wie dies aktuell in China passiert – dann entsteht allerdings eine Dynamik, die viele von uns dann wohl doch lieber vermeiden würden. Plötzlich wird man belohnt, wenn man in sozialen Medien Zustimmung zu Äußerungen führender Politiker verbreitet. Falls man dies jedoch routinemäßig tut, etwa um den eigenen Kindern Bonuspunkte für den Zugang zur guten Schule zur ermöglichen, wird dies als

Betrugsversuch algorithmisch entlarvt – sei es durch Auffälligkeiten bei Text, Menge und Zeitpunkten der Postings oder sogar durch die Erkennung der Mimik bei der Eingabe des entsprechenden Kommentars. Werden solche Systeme von der Spitze einer Gesellschaft aus eingeführt, so ist der freundliche große Bruder allgegenwärtig, es sei denn, das Volk erhebt sich zur Revolution und zum modernen Maschinensturm.

Das Einzige, was in freiheitlichen Gesellschaftsordnungen funktionieren kann, ist der Wettbewerb und der Wunsch der Kunden. Wenn sich genügend Menschen an der Idee berauscht haben, 20.000 Follower zu haben, stellen viele von ihnen vermutlich eines Tages fest, dass sie eigentlich doch eher in einem virtuellen Dorf leben möchte, welches im Kern weniger als zweihundert Menschen umfasst. Wenn aus solcher Einsicht heraus dann das Bedürfnis nach Vertraulichkeit wächst, wird man Software schaffen, welche genau diese Bedürfnis-Struktur abdeckt. Vielleicht stehen die Server dafür dann in kleineren Ländern mit einer gewissen Autonomie und liberaler Rechtsordnung. Möglicherweise entsteht mittelfristig eine Art *Medium Brother Softaware*, die sich besonderen Glaubwürdigkeitsüberprüfungen stellt. Wenn ihr das gelingt, dann wird sie wahrscheinlich versuchen, sich als *Companion* darzustellen, dem man ruhig jederzeit voll vertrauen kann, von Einkaufsvorlieben und politischen Interessen über den Familienstatus bis zum aktuellen Aufenthaltsort. Ist man mit einem Stammesmitglied zusammen, wird es der *Companion* ohnehin wissen. Der Gipfel des Vertrauens ist erreicht, wenn ein solches System auch noch so diskret ist wie ein guter Freund und mit sich ringt, ob es außereheliche Beziehungen decken soll ;-)

Companions

Zuerst fallen einem vielleicht Systeme wie *Alexa* oder *Siri* ein, aber das Thema *Partnerschaft* wird uns in Zukunft in wichtigerem Zusammenhang begegnen, nämlich im Auto. Wir erleben heute eine Phase, in der das Fahrzeug einzelne Assistenzfunktionen anbietet, die teil-autonomes Fahren erlauben, etwa in der Verbindung aus Spurhalte-Assistent und adaptiver Abstandsautomatik. Der erste Impuls kommt vom Fahrer, indem er die Assistenzfunktion aktiviert. Während sie aktiv ist, kann er entweder selbst eingreifen („over-ruling“) oder das System erkennt eine Situation, mit der es nicht zurecht kommt und fordert sein Eingreifen aktiv an. Sobald solche Assistenzsysteme einen bestimmten technischen Reifegrad erreicht haben, arbeiten sie erfahrungsgemäß innerhalb ihres Spezifikationsbereichs besser und zuverlässiger als der Mensch.

Der Mensch macht heute als Autofahrer eine Erfahrung, die Flugzeugpiloten schon seit 50 Jahren kennen: Er gerät allmählich „out of the loop“, soll aber im Notfall sofort übernehmen. Dafür ist der Mensch jedoch nicht geschaffen. Gibt man ihm ein Navigationssystem, so verlernt er das Kartenlesen. Gibt man ihm einen Autopiloten, so verlernt er das manuelle Fliegen. Aus diesem Grund ist es für Berufspiloten nützlich, wenn nicht sogar vorgeschrieben, immer wieder Starts und Landeanflüge von Hand durchzuführen. Und deswegen sind die halbjährlichen Checkflüge im Simulator so wichtig, denn dort werden vor allem Notfall-Situationen trainiert, die manuelles Eingreifen verlangen.

Im Folgenden erläutern wir das Konzept der *Companion Softaware* anhand von zwei Unfällen, die sich in jüngerer Zeit mit dem neu entwickelten Modell *Boeing 737 MAX* ereignet haben.

Um Treibstoff zu sparen im Vergleich zu den bisherigen Modellen der 737, musste man das Nebenstromverhältnis der Triebwerke verbessern, also ihren Durchmesser vergrößern. Damit weiterhin genug Bodenfreiheit blieb, musste man die Triebwerke etwas höher befestigen. Der Platz unterhalb des Flügels war jedoch knapp; also verlegte man den neuen, größeren Lufteinlass weiter nach vorn, vor die Vorderkante des Flügels. Dies bewirkte eine Verschiebung des Flugzeugschwerpunkts (center of gravity) gegenüber dem Angriffspunkt der Auftriebskraft (center

of lift), der in Flugsituationen mit großem Anstellwinkel rascher als bisher zu einem Strömungsabriss (*Stall*) führen kann. Man ahnte daher: Durch den Umbau würde sich das Flugzeug weniger „gutmütig“ als bisher verhalten.

Bei kleinen Flugzeugen kündigt sich ein Stall durch Vibrationen an. Während ihrer Ausbildung auf Kleinflugzeugen lernen Piloten, dass ein nahender *Stall* unregelmäßigen Druck auf die Steuerflächen ausübt, d.h. der mechanisch mit den Steuerflächen verbundene Steuerknüppel vibriert, manchmal schüttelt sich sogar das gesamte Flugzeug.

Um den Piloten großer Flugzeuge genau dieses Gefühl zu vermitteln, obwohl ihr Steuerknüppel nicht mehr mechanisch mit den Steuerflächen verbunden ist und obwohl ihre Masse zu träge für rechtzeitig wahrnehmbare Erschütterungen ist, gibt es seit Jahrzehnten Sensoren, welche einen nahenden Strömungsabriss detektieren und elektrische „Stick-Shaker“ aktivieren, welche den elektrischen Joystick des Piloten vibrieren lassen.

Dies ist ein Beispiel für besonders gute *Softaware*: Sie versetzt sich in die Erfahrungswelt des Benutzers und spielt ihm absichtlich etwas vor, was er aufgrund seiner Ausbildung für die Folge eines kritischen Flugzustands hält, den die Software gerade anderweitig ermittelt hat.

Dieses bewährte Verfahren hat Boeing offenbar bei der *737 MAX* nicht mehr für ausreichend gehalten. Man wollte offenbar bereits bei der Annäherung an den kritischen Bereich „sanft eingreifen“. Eine kleine gegensteuernde Aktion durch die Höhentrimmruder sollte das Problem ursprünglich lösen. Als man diese Maßnahme der Aufsichtsbehörde erklärte, ging sie davon aus, dass wegen der vorgesehenen Begrenzung der Trimmkorrektur auf ein halbes Grad keine sicherheitsrelevante Veränderung entstehen könnte. Man überließ daher dem Hersteller die weitere Entwicklung, ja sogar Test und Abnahme.

Es muss in der Folge (bei Testflügen?) Ereignisse gegeben haben, die es sinnvoll erscheinen ließen, die gefährliche Stall-Zone energischer zu vermeiden, als es ursprünglich vorgesehen war. Man stellte der Software daher den gesamten Einstellbereich der Trimmung zur Verfügung. Da die Trimmung wie ein Multiplikator auf das eigentliche Höhenruder wirkt, hatte dies einen enormen Effekt. Vermutlich entstanden zwischendurch auch noch weitere Erwartungen an die neue Software-Komponente, denn ihr Name (MCAS = Maneuvering Characteristics Augmentation System) klingt mehr nach einem allgemeinen „Besänftigungssystem“ als nach einem spezifischen „Early-Stall-Warning-System“. Von dieser Idee ausgehend erschien es logisch, das MCAS mit dem Erreichen der normalen Flugkonfiguration, d.h. nach dem Einzug des Fahrwerks und dem Einfahren der Startklappen automatisch zu aktivieren. Für den Piloten sollte sich einfach „alles so anfühlen wie bisher“. Mit dieser Grundhaltung ging der Hersteller in den Markt.

Die Fixierung auf das sichere Vermeiden einer unerwünschten Situation (Stall) und die Idee, den Piloten automatisch „im Hintergrund“ vor Fehlhandlungen zu bewahren, hat zu vielen Toten und zu großem Vertrauensverlust in die Professionalität von *Boeing* geführt. Was ist schief gelaufen?

1. Die Existenz des MCAS wurde gegenüber Piloten bei Schulungsveranstaltungen verschwiegen und gegenüber Fluggesellschaften heruntergespielt, welche die neue Version der B 737 bestellten. Eine Zusatzanzeige, die das MCAS darstellt und auch ein Warnsignal (master caution) bei einer Störung des MCAS hätte geben können, wurde lediglich als kostenpflichtige Sonderausstattung angeboten – und daher manchmal nicht bestellt.
2. Die Aktivierung des (den Piloten zumindest in einem Fall unbekannten) Systems erfolgte automatisch auf völlig intransparente Weise durch das endgültige Einfahren der Startklappen. Auf diese Weise hätte man es übrigens auch wieder abschalten können, aber auf diese Idee konnte niemand kommen! Es hätte auch genügt, den Circuit Breaker für die Höhentrimmung zu ziehen. Aber ein Mensch, der nicht weiß, gegen welches System er kämpft, findet unter Stress diesen rettenden Ausweg nicht.

3. Über ein Bordhandbuch konnten die Piloten herausfinden, dass ein MCAS existiert und wie man es gezielt deaktiviert. Sie folgten beim zweiten Unglück den vorgesehenen Schritten, aber kurz danach aktivierte sich das System offenbar wieder von alleine.

Diese drei Punkte untergraben den Kontrakt der Kooperation vom Mensch und Software-Assistenzsystem im Cockpit in jeder Hinsicht. Ein besonders eklatanter Verstoß gegen „fairen Umgang“ zwischen Mensch und Maschine ist die automatische intransparente Reaktivierung des Systems nach einer expliziten Abschaltaktion durch den Menschen.

Aus einem kleinen „Besänftigungssystem“ wurde quasi ein neuer unbekannter und resistenter Bestandteil des „vegetativen Nervensystems des Flugzeugs“. Und dies ganz ohne Evolutionsgeschichte, einfach so nebenbei. Was in der Natur die Evolutionsgeschichte leistet, muss in der Softwareentwicklung der „System-Test“ übernehmen: Nämlich das Erkennen und Entfernen von Fehlern, die auf Umständen beruhen, an die man bei der Entwicklung nicht gedacht hat.

4. Man darf davon ausgehen, dass das MCAS seine Besänftigungs- und *Stall*-Verhinderungs-Aufgabe in simulierten Tests, vielleicht auch bei Probeflügen sicher gemeistert hat – spätestens nachdem man ihm den vollständigen Einstellbereich des Trimmruders zur Verfügung gestellt hatte. Dieser Test des sogenannten „Positivfalls“ ist jedoch nie ausreichend.
   Man muss zusätzlich andere Menschen damit beauftragen, ungewollte Seiteneffekte zu entdecken und Anfälligkeiten für die Variation von Umweltbedingungen zu untersuchen. So wie Unternehmen heute „gute Hacker“ damit beauftragen, heimlich in ihr IT-Netz einzudringen, muss man kritische Softwaresysteme auf ihre impliziten Annahmen hin überprüfen und sie „ärgern“, um ihnen auf den Zahn zu fühlen. Denken wir an die *Underdogs* zurück: Wenn man ihr den Strom wegnimmt, muss die Gasheizung das Ventil schließen. Wenn man einem MCAS einen defekten Sensor unterschiebt, muss es sich deaktivieren und dies gegenüber dem Piloten anzeigen.
5. Wie testet man also? Man benötigt ein genaues Verständnis davon, auf welche Weise ein Sensor defekt sein kann: Er kann beispielsweise überhaupt nicht antworten, ungültige Werte liefern, oszillierende oder zufällige Werte liefern, konstante Werte am Rande des zulässigen Bereichs senden usw.). Hat man ein solches „Modell eines defekten Sensors“, kann man dem MCAS im Test jede Art von defektem Sensor unterschieben und beobachten, wie es damit umgeht. Man muss herausfinden, ob die Software genügend *awareness* besitzt für ihre physikalische Umwelt, ob sie Naturgesetze einbezieht, Plausibilitäts-Checks durchführt und Fehlermöglichkeiten antizipiert.

Diese Art von Tests können im vorliegenden Fall nicht stattgefunden haben, denn sie hätten unweigerlich gezeigt, dass das MCAS bei einem Sensor, der konstant einen sehr hohen Anstellwinkel liefert, das Flugzeug in einen steilen Abwärtsflug zwingt, aus dem es kein Entrinnen gibt. Soviel zum Test.

Fehler, die man beim Test entdeckt, verursachen ein Vielfaches der Kosten von Fehlern, die man im Entwurf entdeckt oder gar bereits bei der Konzeption eines System vermeidet. Das MCAS-Beispiel ist auch dazu recht ergiebig:

6. Das MCAS stützt sich auf einen Sensor, der den relativen Anstellwinkel (Angle Of Attack, AOA) misst. Es benutzt nur einen einzelnen solchen Sensor, obwohl das Flugzeug zwei davon hat und obwohl an anderer Stelle (in der Autopilot-Software) beide Signale ausgewertet und verglichen werden. Wieso geht ein Hersteller so naiv und dilettantisch vor, der bereits so viele sicher funktionierende Softwaresysteme gebaut (oder beauftragt) hat?

Die wahrscheinlichste Antwort ist, dass eine Integration von MCAS in die vorhandene Software des Autopiloten als zu teuer und vor allem zu langwierig erschien, weil diese Software aus Redundanz-Gründen von unterschiedlichen Teams parallel entwickelt wird und bei Änderungen aufwendige Tests und Re-Zertifizierungen fällig werden.

7. Hätte man nicht trotzdem beide AOA-Sensor-Signale abgreifen können? Möglicherweise ist die Software-Schicht, welche die Signale der AOA-Sensoren miteinander abgleicht, integraler Bestandteil des Autopiloten, d.h. sie war nicht als unabhängig verwendbarer Baustein für ein anderes System verfügbar. Hätte man sie verwenden können, so hätte allein dies wohl schon ausgereicht, die Katastrophe zu verhindern.
8. Gute *Softaware* schützt sich jedoch auch ohne physische Redundanz durch doppelte Sensoren zusätzlich, indem sie bestimmte Annahmen über ihre Peripheriegeräte trifft. Beispielsweise kann sie den Daten eines Sensors misstrauen, wenn sich diese sprunghaft ändern, wie dies bei defekten Sensoren vorkommen kann. Gute *Softaware* hat also eine *Kontinuitätserwartung* und besitzt „Abwehrmechanismen“. Sie kann z.B. einzelne „Ausreißer“ ignorieren, wenn sie vorübergehender Natur sind. Selbstverständlich wird sie solche Vorkommnisse in ihrem Logbuch speichern und (später) eskalieren.
9. Gute *Softaware* hat, quasi als Gegenpol zur Kontinuitätserwartung, eine *Volatilitätserwartung* gegenüber einem Sensor. Ein Sensor, der eisern einen konstanten Wert liefert, ist defekt, weil die physikalischen Prozesse und sein elektrischer Aufbau quasi wesensnotwendig zu ständigen winzigen Änderungen bei den Nachkommastellen führen. Sind solche Änderungen nicht feststellbar, ist der Sensor nicht mehr vertrauenswürdig.
10. Gute *Softaware* hat eine *Reiz-Reaktions-Erwartung*: Hat sie ein Stellglied betätigt, etwa im konkreten Fall die Höhentrimmung verändert, so muss sie nach Abwarten einer gewissen Verzögerung eine Reaktion des Sensorwerts „in die richtige Richtung“ und mit einer erwarteten Größenordnung feststellen können. Tritt diese Reaktion nicht ein, so ist entweder der Aktor (die Höhentrimmung) defekt, d.h. die angestoßene Aktion hat gar nicht stattgefunden, oder der Sensor ist defekt – oder der Pilot hat gegengesteuert! Würde sich die MCAS-Software mit den Reaktionen des Piloten befassen, so könnte sie ganz anders reagieren. Es wäre übrigens eine gute Idee, eine Software-Komponente, welche Reiz-Reaktions-Erwartungen überprüft, von einem anderen Team entwickeln zu lassen als ein System wie das MCAS – das aber nur nebenbei.
11. Der Schlusspunkt und sozusagen die Königsdisziplin für gute *Softaware* besteht darin, ein *Weltmodell* zu besitzen und ständig alle ausgelösten Aktionen auf das Weltmodell anzuwenden und ermittelte Sensordaten auf Vereinbarkeit mit dem so fortgeschriebenen Weltmodell zu prüfen. Dabei sind Zeitverzögerungen verschiedener Art aufgrund der Trägheit physikalischer Prozesse zu berücksichtigen. Ein genaues Weltmodell kann sehr aufwendig und rechenintensiv sein. Oft genügt aber schon ein einfaches Modell mit wenigen Variablen. Einem Stall geht beispielsweise eine stark abnehmende relative Luftgeschwindigkeit voraus sowie eine Höhenzunahme, die in einem „ungesunden Verhältnis“ zu dieser langsamen Geschwindigkeit steht. Die eingeleitete Gegenmaßnahme muss zu reduziertem Anstellwinkel und zunehmender Luftgeschwindigkeit führen. Passen die gemessenen Werte für Geschwindigkeit, Triebwerksleistung und Höhenrudereinstellung zusammen, so kann ein AOA-Sensor als defekt erkannt werden, falls er einen Wert zeigt, der zu diesem Mehrheitsbild *nicht* passt. Das alles muss allerdings geistig entworfen, durchdacht, programmiert und getestet werden.

Wenn ein Softwaresystem dann irgendwann so robust ist, dass es selbst Doppelfehler bei Sensoren und Aktoren noch als solche erkennen kann, dann hat man eine Stufe erreicht, auf der die Software ein echter Partner für den Menschen ist. Möglicherweise würde sich der Mensch über die

„Instinktsicherheit“ dieser *Softaware* wundern, sie sogar bewundern. Auf jeden Fall wird er sich ihr über kurz oder lang anvertrauen.

Die Zügel nicht verlieren

Die Zeiten, in denen ein unbekümmerter Fortschrittsglaube eine glänzende, vollautomatisierte Zukunft an den Horizont malte, sind vorbei. Dystopien und Kassandrarufe werden uns aber auch nicht davon abhalten, das Machbare auszuloten und zu konstruieren. Die Bequemlichkeit des Menschen wird die meisten von uns dazu bringen, Software-Systemen einen noch größeren Raum in ihrem Leben einzuräumen als heute. Erfolgsstatistiken werden belegen, dass *Companion Softaware* zuverlässiger ist als der Mensch. Schon heute setzen wir uns in führerlose U-Bahnen (Nürnberg, London-Docklands). Morgen werden diese Ausnahmen zur Gewohnheit, übermorgen zur Regel.

Heutige Software-System arbeiten überwiegend mit Algorithmen, deren Wirkungsweise gezielt entworfen und programmiert wurde. Zumindest Experten können haargenau erklären, wie das System sich unter bestimmten Bedingungen verhält. Inzwischen gibt es jedoch auch Algorithmen, die mit selbstverstärkendem Lernen „trainiert“ werden, deren Wirkmechanismus also nicht von einem Menschen konzipiert wurde. Solche System haben oft erstaunliche Fähigkeiten, besitzen jedoch prinzipbedingt keine „Erklärungskomponente“. Der Mensch kann bei solchen Systemen also nicht mehr nachvollziehen, warum sie eine bestimmte Entscheidung getroffen haben. Unser Vertrauen solchen Systemen gegenüber wird sich nur noch auf „Blackbox“-Beobachtung gründen können. Manche Banken vertrauen beispielsweise bei der Prüfung von Unterschriften heute solchen Systemen, weil ihre Erkennungsrate bei Fälschungen statistisch besser ist als die von Menschen. Noch haben solche Systeme ihren Schwerpunkt bei der Erkennung von Mustern, sei es bei Unterschriften, bei Gesichtern, bei Zellmutationen in Gewebeproben oder bei der Erkennung des Sprechers einer Audionachricht. Aber er es ist auch denkbar, lernende Systeme mit Daten zu trainieren, die sich auf den Zustand eines Flugzeugs, Fahrzeugs oder einer Chemieanlage beziehen. Wollen wir zulassen, dass so konzipierte Systeme in physikalische Regelkreise eingreifen, also die Autoräder lenken oder die Brennstäbe eines Reaktors bewegen?

Jedes mal, wenn wir einem Softwaresystem größeren Spielraum einräumen als bisher, müssen wir prüfen, ob das Verhältnis Mensch-Maschine dadurch nicht nur quantitativ tangiert wird, sondern auch eine andere *Qualität* bekommt. Sorgen wir dafür, dass wir jeden einzelnen Schritt mit Umsicht gehen und Systeme schaffen, die uns mit mehr Bewusstsein für menschliche Bedürfnisse und für technische Unzulänglichkeiten begegnen als heute!

Dazu müssen wir vor allem bei uns selbst anfangen. Wir müssen ein Rollenverständnis entwickeln und die Rahmenbedingungen vorgeben, in denen dann Mensch und Maschine kooperieren.