Kundenorientiertes Testen von Künstlicher Intelligenz

Thomas Fehlmann und Eberhard Kranich, Euro Project Office AG, Zürich und Duisburg
thomas.fehlmann@e-p-o.com, eberhard.kranich@e-p-o.com

Zusammenfassung

Testen von Software wird heute immer noch gelegentlich mit Testen von Code verwechselt. Das ist im Zeitalter von Cloud Services und lernenden Systemen ziemlich fatal. Noch schlimmer wird diese Verwechslung, wenn man an cyber-physikalische Produkte denkt, etwa medizinische Instrumente oder autonome Fahrzeuge, wo es durchaus nicht nur der Code ist, der getestet werden muss, um Unfälle zu verhindern.

Das Verhalten lernender Systeme wird nicht alleine durch Algorithmen bestimmt, sondern auf Grund von Lernprozessen und Trainingsdaten. Es ist sinnlos, ein solches System nur im Auslieferungszustand zu testen, denn es kann auch im Betrieb lernen, aber auch verlernen. Um sicher zu bleiben, muss es immer und immer wieder getestet werden.

Dieser Artikel schlägt einen autonomen Testprozess vor, der relevante Testfälle ohne menschliches Zutun auswählen kann dank Bezug zu den Kundenbedürfnissen. Mit Hilfe statistischer Methoden und linearer Matrizenalgebra gelingt eine vollständige und individualisierbare Sicherstellung relevanter Verhaltensweisen künstlich intelligenter Systeme.

Schlüsselwörter

Künstliche Intelligenz, Kundennutzen, Sicherheit, Autonomes Testen, Kontinuierliches Testen, Transferfunktionen, Sensitivitätsanalyse.

Einführung

Die Grundlagen der KI sind die Klassifizierung von Entitäten und die Lösung der Gleichung

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) \tag{1}$$

wobei $\boldsymbol{x}$ und $\boldsymbol{y}$ Vektoren in Räumen unterschiedlicher Dimension und Semantik sind. So könnte $\boldsymbol{y}$ beispielsweise das beobachtbare Verhalten von Menschen oder extrasolaren Planeten sein und $\boldsymbol{x}$ die unbekannte Ursache dafür.

Wenn nicht viel über die Übertragungsfunktion $\boldsymbol{f}$ bekannt ist, können neuronale Netze eingerichtet werden, die die Übertragungsfunktion auf der Grundlage von Erfahrungen erlernen.

Ein Bilderkennungssystem lernt, zwischen gehenden, rennenden, spielenden, wartenden Menschen und fahrenden oder geschobenen Fahrrädern zu unterscheiden. Es sollte auch lernen, Personen und Dinge zu erkennen, die nicht vollständig zu sehen sind, beispielsweise hinter einem Busch, durch Nebel oder bei Nacht verborgen sind. Weder Bildmerkmale noch Algorithmen zur Mustererkennung sind einprogrammiert.

Leider lernen solche neuronalen Netze nicht nur, sondern sie verlernen auch. Van Gerven et. al. [1] haben gezeigt, dass sie wie jede neuronale Informationsverarbeitung, ähnlich wie beim Menschen, in Bedrängnis („verrückt“) geraten können. Bevor ich es wage, mich in ein autonomes Auto zu setzen, möchte ich wahrscheinlich wissen, ob und wann dieses Auto mit seinem aktuellen Stand des lernenden Systems seinen letzten Test bestanden hat.

Unser Ansatz

Dies erfordert die Fähigkeit, softwareintensive Systeme autonom zu testen. Es muss möglich sein, die Bildanalysefähigkeit eines Autos in regelmäßigen Abständen zu testen, um zu sehen, ob sie noch wie erwartet funktioniert, und so den "Wahnsinn" früh genug zu erkennen, um Schäden zu vermeiden. Ob die alten Fähigkeiten noch funktionieren, muss getestet werden.

Außerdem müssen sich auch die Tests weiterentwickeln. Sie können nicht statisch sein; Testsuiten müssen für neue Erkenntnisse, neue Umgebungen und neue Normen und Vorschriften erweitert werden.

Dies ist das autonome Echtzeit-Testen (*Autonomous Real-time Testing*, ART) [2]. ART benötigt KI-Techniken, um genügend Testfälle zu generieren, und dank dieser Erweiterbarkeit ist ART in der Lage, andere KI zu testen.

Metriken für das Testen von Software

Ein Testszenario ist eine Sammlung von Test Stories. Eine Test Story besteht aus Testfällen, die sich durch einen gemeinsamen Geschäftsbezug auszeichnen. Der Geschäftsbezug wird durch User Stories definiert. Test Stories sind mit den User Stories verwandt, aber in der Regel nicht identisch. Test Stories und Testfälle beziehen sich auf mehr als nur eine User Story und kombinieren verschiedene Aspekte aus lose miteinander verbundenen Fachanwendungen.

Testfälle beginnen mit Vorbedingungen $x_1, x_2, \ldots, x_n$ und liefern eine bekannte Antwort y. Diese werden als *Pfeilterme* notiert:

$$\{x_1, x_2, \ldots, x_n\} \rightarrow y \tag{2}$$

Testfälle enthalten immer schwächste Behauptungen; also Ungleichungen oder Bereichsangaben statt Stichprobenzahlen; siehe dazu [3, p. 319ff].

Tests komplexer Systeme adressieren primär die Softwarefunktionalität. Funktionale Modelle werden seit 40 Jahren für die Dimensionierung von Software verwendet. Die Tests decken die Elemente des Modells ab. Wir haben uns für den ISO/IEC 19761 COSMIC-Standard entschieden [4]. Dieses Modell der Softwarefunktionalität besteht aus Datenbewegungen, bei denen

Datengruppen von einem Objekt von Interesse in ein anderes verschoben werden.

Die Anzahl dieser Datenbewegungen dient dabei als Maß für die funktionale Größe einer Software. Dasselbe Maß passt jedoch auch zu Testfällen. Somit hat man zugleich ein Maß für die Testintensität, die Testabdeckung und die Fehlerdichte einer Software.

Der ISO/IEC Standard 19761 genügt insbesondere auch der Norm ISO/IEC 14143 [5]. Deswegen sind Modelle vollständig durch Kundenanforderungen, meist in Form von User Stories, definiert. Code braucht nicht dazu. Und somit lassen sich auch lernende Systeme und KI-Anwendungen mittels Modellen darstellen.

Testabdeckung

Darüber hinaus erlaubt das von ISO/IEC 19651 definierte Maß jedoch, die Testabdeckung genauer zu analysieren. Denn die User Stories haben ebenfalls eine funktionale Größe, und es lassen sich die Datenbewegungen identifizieren, die für eine User Story ausgeführt werden müssen. Das ist eigentlich nichts anderes als Anforderungsverfolgung.

Wenn nun aber von Datenbewegungen bekannt ist, welchen User Stories sie zudienen, und umgekehrt dieselben Datenbewegungen in einem ausführbaren Testfall eine Test Story vorkommen, dann liegt es auf der Hand, dass man eine Matrix erstellen kann, eine Gegenüberstellung zwischen User Stories und Test Stories, deren Zellen aus der Anzahl Datenbewegungen besteht, die in einem gegebenen Testfall eine bestimmte User Story testen. Somit erhält man die Testabdeckungsmatrix als „Dichtefunktion“ des Testens der verschiedenen User Stories.

Kennt man dann zusätzlich die relative Wichtigkeit der User Stories für den Kunden des Produktes, dann kann man sogar feststellen, ob ein Testszenario – also die Summe aller Testfälle aller Test Stories – die für den Kunden relevanten User Stories testet, oder ob man den Testaufwand verschwendet auf möglicherweise einfache, aber letztlich weniger wichtige Testfälle.

Zum Begriff „Kunde“

Mit dem Begriff „Kunde“ bezeichnen wir grundsätzlich jeden, der Nutzen aus einem Produkt bezieht. Dabei kann es sich um mehrere Parteien handeln – etwa Fahrgäste, Betreiber und Eigentümer eines Schienenfahrzeugs, deren Interessen und Sichtweisen keineswegs immer übereinstimmen müssen.

Es kann also mehrere Arten von „Kunden“ geben, die unterschiedliche Testfälle als relevant sehen. Das bedeutet, dass es auch unterschiedliche Testszenarien für ein und dasselbe Produkt gibt, genauso wie es auch in User Stories unterschiedliche Anforderungssteller mit teilweise widersprüchlichen Anforderungen gibt. Der Begriff „Kunde“ enthält also Mehrdeutigkeiten, mit denen man umgehen muss.

Die Wichtigkeit von User Stories

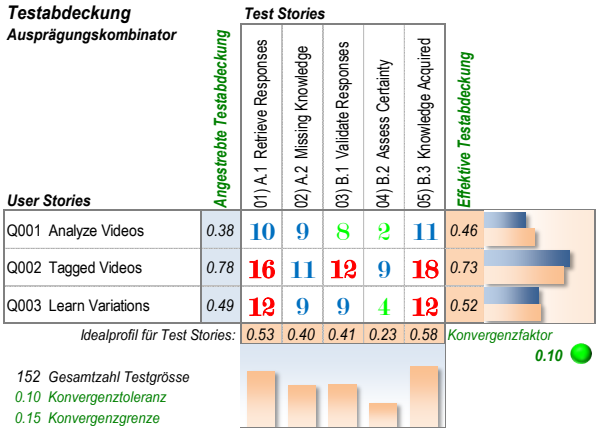
In der agilen Softwareentwicklung dient die relative Wichtigkeit von User Stories der Planung von Sprints, und hilft gleichzeitig, unnötige Anforderungen rechtzeitig zu erkennen. Auch dazu gibt es einen ISO/IEC Standard, nämlich ISO/IEC 16355 [6], der sich eignet, User Stories zu priorisieren. Bekannt ist dieser Standard für die Anwendung statistischer Methoden in der Produktentwicklung unter dem Namen *Quality Function Deployment* (QFD) [6].

Dieser Standard verlangt, dass Prioritäten in Form von Wichtigkeitsprofilen erfasst werden. Profile sind Vektoren im Raum aller Anforderungen, respektive aller Test Stories, dem Testszenario. Diese Profilvektoren werden mittels der Euklidischen Norm auf Länge Eins normiert, was erlaubt, sie ohne Gefahr der Verzerrung zu kombinieren. Das ist das übliche Vorgehen in der Statistik und erlaubt die Anwendung statistischer Methoden. Priorisiert man nämlich anders, etwa mit linearen Skalen, ist Statistik grundsätzlich nicht anwendbar. Mehr dazu findet man zum Beispiel im Buch über „Managing Complexity“ [3, p. 33].

Ein Beispiel einer Testabdeckung zwischen 3 User Stories und 5 Test Stories zeigt die folgende Matrix in Abbildung 1. User und Test Stories beziehen sich auf ein Applikationsmodell von 8 CFP. Das einfache Modell beschreibt die korrekte Erkennung szenischer Abläufen („Videos“), etwa im Straßenverkehr. Die Korrektheit der Erkennung wollen wir testen.

Eine Beschreibung der Applikationsmodelle findet man in unserem Artikel über das Testen von KI [7], sowie im Buch über ART [2, p. 73].

Abbildung 1: Testabdeckung



Die Zahlen in den Zellen der Matrix bezeichnen die Summe der Datenbewegungen aller Testfälle, die einer User Story zugeordnet werden können, und damit das funktionale Gewicht, das den Tests für eine bestimmte User Story zukommt. Das Total dieser Testfälle bestimmt die Testgröße, die einige Größenordnungen, genannt Testintensität, über der funktionalen Größe der getesteten Software liegen sollte.

Der Konvergenzfaktor

Die Frage, ob ein Testszenario der vom Kunden bestimmten Wichtigkeit der User Stories entspricht, entscheidet der *Konvergenzfaktor*. Dieser berechnet sich aus der Vektordifferenz

$$\|\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{x})\| \quad (3)$$

also dem Abstand zwischen dem funktionalen Gewicht eines Testszenarios $\boldsymbol{f}(\boldsymbol{x})$, wo $\boldsymbol{f}$ die von der Testabdeckungs-Matrix berechnete Funktion ist, und der Zielvorgabe $\boldsymbol{y}$ des Kunden, der von ihm bestimmten Profils der Wichtigkeit der User Stories.

Der Konvergenzfaktor zeigt an, wie gut eine approximative Lösung der Gleichung (1) ist. Mit diesem Mechanismus kann man aus einer großen Anzahl von denkbaren Testfällen diejenigen auswählen, welche für den Kunden relevant sind.

Generieren von Testfällen

Um Testfälle kombinieren zu können, muss man Formel (2) rekursiv verallgemeinern. So lassen sich Pfeilterme beliebiger Stufe definieren, und damit auch die Kombination von zwei Mengen M, N solcher Pfeilterme

$$M \bullet N = \{\gamma | \exists \{x_1, x_2, \dots, x_m\} \rightarrow y \in M;\ \beta_i \in N\} \quad (4)$$

Testfälle sind kombinierbar, wenn es Testfälle gibt, welche die Vorbedingungen $x_1, x_2, \dots, x_n$ verifizieren, die für die erwartete Antwort y gelten sollen. Formel (4) definiert die Datenstruktur einer solchen Algebra der Testfälle.

Auf diese Weise lassen sich aus einer ursprünglichen Menge von Testfällen, etwa den Unit Tests, beliebig viele neue Testfälle kombinieren.

Aber es fehlt die Fähigkeit, relevante Testfälle zu erkennen. Dazu braucht es etwas KI, sowie die Grundlagen ISO/IEC 16355 QFD und ISO/IEC 19761 COSMIC.

Modellbasiertes Testen

Insbesondere bei lernenden Systemen gibt es keine Software, die man testen könnte. Zwar steckt in einer Support Vektor Maschine (SVM) auch Software drin, ziemlich viel sogar, aber diese Software liest und kategorisiert die erhaltenen Eingaben, was eigentlich für den Kunden nicht interessant ist. Der Kunde interessiert sich vielmehr für die Folgerungen, die aus dieser Kategorisierung gezogen werden; diese müssen stimmen.

Dafür muss der Tester Modelle aufstellen. Diese beschreiben in allgemeiner Weise das erwartete Verhalten, wie durch eine programmierte Software verursacht, aber dazu braucht es weder Code noch einen Algorithmus.

Für das modellbasiertes Testen gibt es einen für Tester besonders wichtigen ISO-Standard: ISO/IEC 14143 [5]. Dieser Standard beschreibt, wie man die Granularität eines Modells, respektive der Messung der Größe dieses Modells, bestimmt.

Die Granularität ist durch die Sicht des Kunden gegeben und kann auf einem beliebigen Niveau sein. Eventuell müssen nur ein paar wenige Datenbewegungen getestet werden, um das kritische Verhalten eines zu testenden Produktes vollständig zu erfassen. Die technischen Details kann man sich sparen und erzielt trotzdem ein hohes Niveau an Vertrauen in das Produkt.

Natürlich kann man sich eine hohe Granularität nur erlauben, wenn man Grund dazu hat, dem Testzustand der verwendeten Komponenten zu vertrauen. Um mit funktionalen Tests auf System- oder Produktebene Komponenten zu validieren, braucht den entsprechenden Fokus mit einer viel feineren Auflösung.

Die Aussage von ISO/IEC 14143-1 ist im Wesentlichen, dass die funktionale Größe einer Software vom Blickwinkel des Kunden abhängt. Das heißt, gemessen wird, was der Kunde als Funktionalität sieht. Die Granularität des Modells ist also wesentlich bestimmt durch die Anforderungen und Erwartungen, auf die sich der Kunde verlassen muss, oder möchte. Dafür braucht es ein Modell, das man testen kann. Dies ermöglicht ein passgenaues Testen auch von komplexen Anwendungen und insbesondere von KI. Getestet wird also nicht die SVM oder das neuronale Netzwerk, sondern was es gelernt hat. Auch bei einem Menschen, der eine Prüfung ablegt, wird nicht das Gehirn gescannt.

Wie man KI testet

Computer Vision und Künstliche Intelligenz (KI) überschneiden sich. KI unterscheidet sich von gewöhnlicher Software durch ihre Lernfähigkeit. Das heißt, KI kann sich an neue Umgebungen, Daten, Bilder und Videos anpassen. Während KI auch für andere Aufgaben eingesetzt werden kann, befasst sich Computer Vision mit der Theorie hinter künstlichen Systemen, die Informationen aus Bildern extrahieren. Bereiche der KI befassen sich mit der autonomen Planung oder Beratung von Robotersystemen, die durch eine Umgebung navigieren. Um sich in dieser Umgebung zurechtzufinden, ist ein detailliertes Verständnis der Umgebung erforderlich. Informationen über die Umgebung können von einem Computer Vision System geliefert werden, das als Vision Sensor fungiert und Informationen über die Umgebung und den Roboter liefert.

KI und Computer Vision haben auch andere Themen wie Mustererkennung und Lerntechniken gemeinsam. Folglich wird die Computer Vision manchmal als Teil der KI betrachtet. Das Testen von KI in der Computer Vision ist nicht so einfach, vor allem, weil es nicht möglich ist, das richtige Ergebnis vorherzusagen. Der Testfall kann verschiedene Antworten hervorbringen, die alle in einem bestimmten Stadium der Erfahrungssammlung richtig sind.

KI ist im Grunde genommen eine Sortierung von Daten in Kategorien auf der Grundlage früherer Lernprozesse oder von Mustersätzen. Ein autonom fahrendes Auto reagiert nur auf das, was sein Lidar und seine visuellen Kameras erkennen. Die Schwierigkeit besteht

darin, die richtige Kategorie zu finden. Menschen haben die gleiche Schwierigkeit, wenn ein Radfahrer vom Gehweg auf die Straße fährt. Entgegen der ursprünglichen Erwartung, dass es sich um einen Fußgänger handelt, müssen sie die verwendeten Kategorien schnell an ein Fahrrad anpassen, das anderen Verkehrsregeln folgt als ein Fußgänger. Noch komplizierter wird es, wenn der Fußgänger plötzlich ein Skateboard oder einen Motorroller herbeizaubert. Die Verkehrsregeln für die beiden letztgenannten Verkehrsmittel sind unbekannt oder existieren keineswegs überall. Der Mensch ist verwirrt, und die visuellen Erkennungssysteme sind es auch, trotz Lidar und Kameras.

Da der wichtigste Beitrag des visuellen Erkennungssystems in der Kategorisierung besteht, sollte geprüft werden, ob die vom visuellen Erkennungssystem erkannten Kategorien über seine gesamte Lebensdauer hinweg gleichbleiben. Aber das ist nicht genug. Das Verhalten bei bestimmten Beispielbildsequenzen sollte ebenfalls stabil bleiben – es sei denn, neue Lernerfahrungen sagen etwas anderes.

Offensichtlich müssen sich die Tests an das Gelernte anpassen. Andererseits können lernende Systeme neurotisch gestört werden - geistig krank, wie Menschen [8]. Deswegen muss man ART einsetzen [9]. Das Testen von KI muss jederzeit, autonom und ohne menschliches Eingreifen möglich sein.

Erweitern der Testfälle

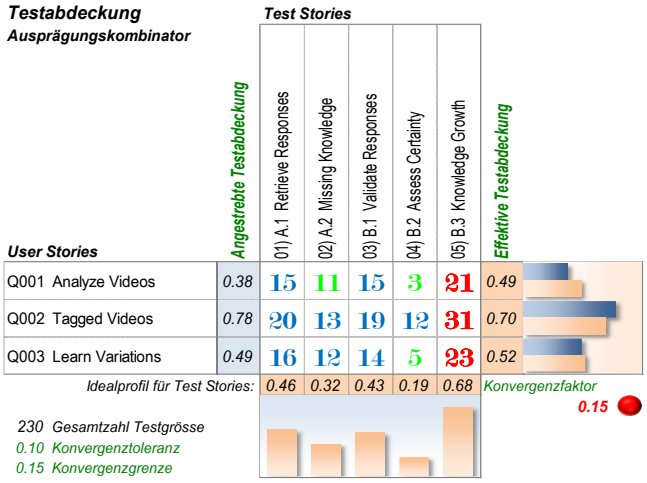
Die Erweiterung von Testfällen erfolgt analog dem Lernen in der AI-Domäne. Für Verkehrsfahrzeuge verwendet man Videosequenzen von Verkehrsszenen, die bereits in der Test Story beschrieben wurden. Dabei kommen Videosequenzen zum Einsatz, die für Deep Learning verwendet wurden, und andere, die nicht verwendet wurden. Man muss die Videos manuell für die jeweilige Verkehrskategorie klassifizieren; es handelt sich also um die gleiche Art von Arbeit für das Testen wie für das Lernen.

Mit ART bleiben die Test Stories aus der ursprünglichen Testsuite stabil, während weitere Testfälle hinzugefügt werden, um die Testintensität zu verbessern und mehr Fehler zu erkennen. Bei visuellen Systemen sind die Hauptquelle für neue Testfälle neue Bilder und Videos. Es ist etwas einfacher als in anderen ART-Fällen, eine ausreichend gute Testabdeckung aufrechtzuerhalten, da Sie nur Testdaten austauschen. Sie ändern nicht das Ziel des Testens.

Eine primäre Quelle für neue Testfälle sind die Inhalte der Datengruppen, die durch die Datenbewegungen im ISO/IEC 19761 COSMIC-Modell bewegt werden. Alles, was in einen bestimmten funktionalen Prozess eintreten kann, muss Teil eines Testfalls werden, und sei es nur, um zu beweisen, dass es keine Wirkung hat. Nach dem Ansatz der kombinatorischen Logik kombinieren wir also alle möglichen Eingaben als Testdaten in den Testfällen und wählen diejenigen Testfälle aus, die den Konvergenzfaktor der Testabdeckungsmatrix klein halten.

Dazu kategorisiert man die Daten aus den Datengruppen nach ISO/IEC 19761 COSMIC. Diese Aufgabe kann mittels KI-Techniken automatisiert werden. Die Details findet man in Fehlmann & Kranich [7]. Die Gesamtzahl der Testfälle in der Matrix aus Abbildung 1 wächst, etwa wie in Abbildung 2.

Abbildung 2: Testabdeckung mit erweitertem Testszenario



Die gesamte Testgröße wächst dabei von 152 CFP auf 230 CFP, aber der Konvergenzfaktor wächst ebenfalls von 0.10 auf 0.15, und damit über die Toleranzgrenze. Bei einer funktionalen Größe von 8 CFP ergibt sich damit eine passable Testintensität von $230/8 = 28.8$. Ergänzt man das Testszenario weiter, verbessert sich die Testintensität, aber der Konvergenzfaktor wird sich höchstwahrscheinlich weiter verschlechtern.

Das Auffinden geeigneter Zellen

Reale Testabdeckungs-Matrizen haben nicht 6×11, sondern Hunderte von Zeilen und Kolonnen. Offensichtlich hat man mit ART zwar eine Lösung, um automatisch Testfälle zu generieren. Damit kann man das lernende Verhalten eines intelligenten Systems prüfen. Doch ist es wenig praktisch, eine Unmenge neuer Testfälle zu generieren, diese dann in eine Testabdeckungs-Matrix einzupflegen, nur um dann den größten Teil wieder wegzuwerfen, weil der Konvergenzfaktor explodiert.

Um zu wissen, bei welchen Test Stories man ansetzen muss, möchte mal gerne wissen, wo, in welcher Zelle der Testabdeckungsmatrix man ansetzen soll. Dies erreicht man mit einer Sensitivitätsanalyse.

Dazu wendet man das „Wendeltreppenverfahren“ (*Winding Stairs,* WS) an, das Fehlmann & Kranich in [10] beschrieben haben. Bezogen auf die Testabdeckungsmatrix in Abbildung 2 durchläuft die WS Methode bei jeder Iteration alle Elemente der Matrix in einer zuvor festgelegten Reihenfolge, z. B. erst alle Elemente der Zeile 3, dann Zeile 1, dann Zeile 2. Jedes Mal wird ein

aktuelles Element a_{ij} geändert und zwar einmal um -1, 0 und $+1$, da die Anzahl der Testfälle immer ganzzahlig ist, siehe Abbildung 2. Dann wird die jeweilige effektive Testabdeckung durch Lösung der Gleichung (1) mittels Eigenvektoren berechnet, woraus sich der jeweilige Konvergenzfaktor g_{-1}, g_0, g_{+1} ergibt. Durch „Minimum-Vergleich“ von g_{-1}, g_0, g_{+1} erkennt man, welche Testabdeckungsmatrix mit $a_{ij} - 1$ oder a_{ij} oder $a_{ij} + 1$ für die nächste Iteration aktuell ist. Eine Verschiebung des Konvergenzfaktors nach oben oder nach unten hängt allein von den Werten g_{-1} bzw. g_{+1} ab, bei g_0 ändert sich nichts.

Man wiederholt dies so lange, bis sich der Konvergenzfaktor nicht mehr verbessern lässt. Da die Zellen der Matrix – in unserem Fall – ganzzahlig und begrenzt sind, weiß man sogar, dass das Verfahren stoppt.

Leider decken die generierten Testfälle in der Regel mehr als eine einzige User Story ab. Das heißt, dass mit jedem hinzugefügten, oder weggelassenen Testfall Seiteneffekte auftreten, wie üblich in QFD. Man kann aber das Wendeltreppenverfahren daran anpassen und nur Modifikationen zulassen, die man mit neu generierten Testfällen erhalten kann.

Schlussbemerkung

Mit dem raschen Aufkommen von lernfähigen, cyberphysikalischen Produkten und Systemen verändern sich die Anforderungen ans Testen massiv. Mit dem Fokus auf Kunden und mathematischen Verfahren lassen sich die Zahl der benötigten Testfälle effektiv eingrenzen.

Zwar sind Kompetenzen wie das Erkennen der Erwartungen des Kunden („Customer Experience“) meist nicht bei der Abteilung für Softwaretests angesiedelt. Doch passt die Methode bestens zur agilen Produktentwicklung. User Stories und Test Stories werden parallel zueinander entwickelt und laufend auf optimale Testabdeckung, also auf einen kleinen Konvergenzfaktor, überprüft. Die Qualität solcher Produkte dürfte stark zulegen, und die Marktakzeptanz auch.

Zitierte Literatur

[1] van Gerven, Marcel; Bothe, Sander, "Artificial Neural Networks as Models of Neural Information Processing," Lausanne, 2018.

[2] T. M. Fehlmann, Autonomous Real-time Testing – Testing Artificial Intelligence and Other Complex Systems, Berlin, Germany: Logos Press, 2020.

[3] T. M. Fehlmann, Managing Complexity - Uncover the Mysteries with Six Sigma Transfer Functions, Berlin, Germany: Logos Press, 2016.

[4] ISO/IEC 19761, "Software engineering - COSMIC: a functional size measurement method," ISO/IEC JTC 1/SC 7, Geneva, Switzerland, 2011.

[5] ISO/IEC 14143-1, "Information technology - Software measurement - Functional size measurement - Part 1: Definition of concepts," ISO/IEC JTC 1/SC 7, Geneva, Switzerland, 2007.

[6] ISO 16355-1:2021, "Applications of Statistical and Related Methods to New Technology and Product Development Process – Part 1: General Principles and Perspectives of Quality Function Deployment (QFD)," ISO TC 69/SC 8/WG 2 N 14, Geneva, Switzerland, 2021.

[7] T. M. Fehlmann and E. Kranich, "Testing Artificial Intelligence by Customers' Needs," *Athens Journal of Sciences,* vol. 6, no. 4, pp. 265-286, 2019.

[8] M. v. Gerven and S. Bohte, "Artificial Neural Networks as Models of Neural Information Processing," in *Frontiers in Computational Neuroscience*, Lausanne, 2017.

[9] T. M. Fehlmann and E. Kranich, "Autonomous Real-time Software & Systems Testing," in *The 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, Göteborg, 2017.

[10] T. M. Fehlmann and E. Kranich, "A Sensitivity Analysis Procedure for Matrix-based Transfer Functions," *Athens Journal of Sciences (proposed),* 2022.