Tom van de Ven
(tom.vande.ven@sogeti.com)
verfügt über 12 Jahre Erfahrung im Bereich Testen von Hightech-Lösungen. Er ist Autor des Fachbuches IoTMap, Experte zum Thema Testen von IoT-Lösungen und regelmäßig als Referent auf Hightech-Seminaren und Events vertreten. Bei Sogeti Niederlande leitet Tom das Testkompetenzzentrum Hightech und ist Mitglied des SogetiLabs.

IoT-Testen: Überlassen Sie das Denken dem Testfall

Das Internet der Dinge (engl. „Internet of Things", Kurzform: IoT) wächst exponentiell – und das Testen muss mit dieser Entwicklung Schritt halten. Möglicherweise haben Sie dies in einem Beitrag gelesen, den ich im vergangenen Jahr veröffentlicht habe [TSE]. Dieser neue Beitrag befasst sich mit der Technologie, die wir benötigen, um die exponentielle Zunahme von Testfällen einzuschränken. Wir werfen einen Blick in die Zukunft des IoT-Testens, in welcher der Testfall die Denkleistung übernimmt.

IoT auf den Punkt gebracht

Nach Pervasive Computing, Ubiquitous Computing und dem Internet der Dinge wurde um das Jahr 2006 das Konzept der cyber-physischen Systeme (CPS) eingeführt, um verschiedene Blickwinkel zusammenhängend zusammenzufassen und so Ordnung in das Durcheinander zu bringen. Aber es ist schwer, Begriffe wieder loszuwerden, und CPS wurde neben dem industriellen Internet, dem industriellen Internet der Dinge, Industrie 4.0 und – nicht zu vergessen – dem Internet of Everything lediglich zu einem weiteren Schlagwort. Heute bezeichnen wir dies alles einfach als „Internet der Dinge".

Wir erleben, wie Geräte, die wir vor Jahrzehnten in Science-Fiction-Filmen gesehen haben, Wirklichkeit werden. Unsere Kleider können unser Wohlbefinden erkennen, eine Bohrinsel auf hoher See kann datenbasiert zum richtigen Zeitpunkt präventiv gewartet werden und selbstfahrende Autos wurden Wirklichkeit. Das sind nur Beispiele für die Hightech-Geräte oder -Systeme, mit denen wir heute arbeiten. Diese Hightech-Systeme messen große Datenmengen und tauschen sich mit anderen Systemen aus.

Systeme, die Daten sammeln und austauschen, sind keinesfalls neu. Wir beobachten aber, wie die Anzahl der Geräte und die Menge der ausgetauschten Daten epische Ausmaße annimmt (siehe Abbildung 1). Diese verschiedenen Entwicklungen fassen

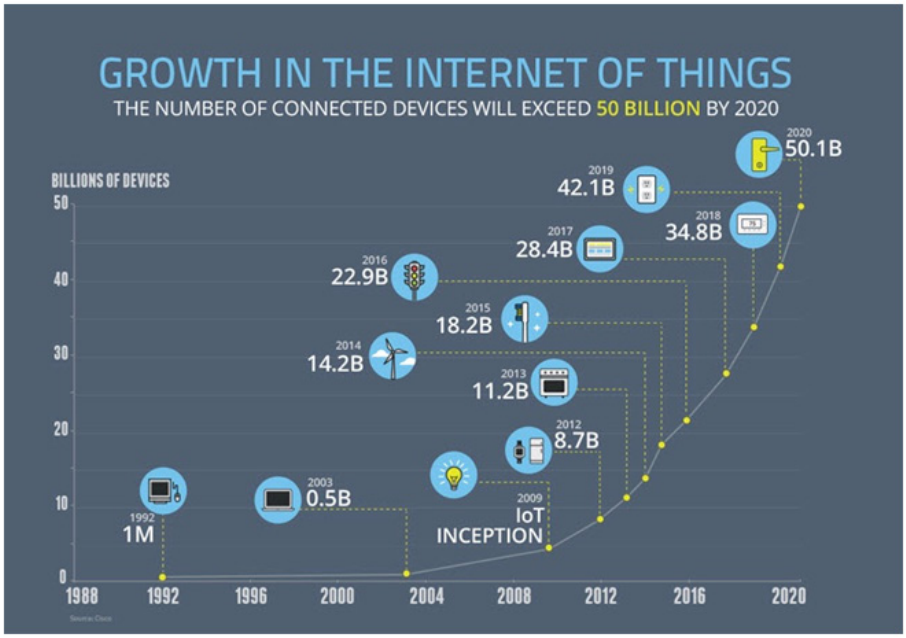

Abb. 1: Voraussichtliches Wachstum des Internet der Dinge (IoT) bis 2020 (Quelle/ Copyright: Cisco Systems).

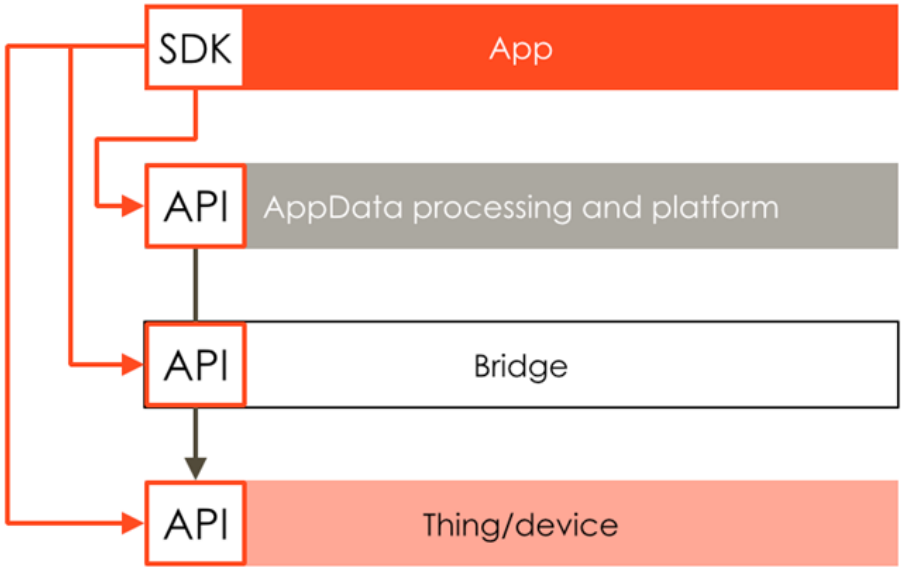

Abb. 2: IoT-Modell für das Testen in vier Schichten.

wir auch unter dem Begriff „Internet der Dinge“ zusammen.

Anhand der Science-Fiction-Beispiele lassen sich grundsätzlich zwei Modelle für IoT-Lösungen unterscheiden. Lösungsmodelle, die mehr mit Unterhaltungselektronik, und solche, die mehr mit industriellen Lösungen assoziiert werden. Letztere werden häufig unter dem Begriff „Industrielles Internet der Dinge“ (Kurzform: IIoT) zusammengefasst.

Bei IIoT-Lösungen haben wir es mit größeren Datenmengen zu tun und somit spielt Big Data hier eine wichtige Rolle. Die grundlegenden Elemente der beiden Modelle unterscheiden sich aber nicht. In diesem Beitrag verwenden wir somit im Kontext des Testens ausschließlich „Internet der Dinge“ als Oberbegriff.

Es gibt eine rasche Zunahme von IoT-fähigen Produkten (Synonyme: Dinge und Geräte). Die Eigenschaften von IoT-Produkten können über verschiedene Konnektivitätsoptionen miteinander verknüpft werden und so kann durch eine neue Kombination auch eine neue IoT-Lösung geschaffen werden. Die Anzahl solcher möglichen Kombinationen ist unendlich. Bestehende Produkte werden Teil von neuen IoT-Lösungen und das wirkt sich auf die Systemarchitektur und -entwürfe aus.

IoT als System der Systeme (SoS)

Bei IoT-Lösungen handelt es sich um ein System der Systeme (engl. „System of Systems“, Kurzform: SoS). Jedes System kann in seine einzelnen Elemente aufgeteilt werden. Schließlich gelangen wir zu den kleinstmöglichen (noch funktionsfähigen und verteilten) Einheiten eines IoT-Ökosystems, die wir „Microservices“ nennen wollen.

IoT-Lösungen, in der jedes IoT-fähige Gerät über Rechenleistung verfügt und über eine drahtlose Verbindung kommuniziert, passen nicht in eine klassische monolithische IT-Architektur [UMS]. Wir brauchen also Microservices, um Systeme mit einem System der Systeme zu verbinden und Daten austauschen zu lassen.

Ein gutes Beispiel für ein System der Systeme ist das Fitness-Tracking. Das Armband, das die Fitnessdaten einer Person erfasst, ist ein System. Das System, das die gesammelten Daten an zentraler Stelle speichert, ist ein weiteres System. Die Webseite, auf der Sportler durch einen Ergebnisvergleich virtuell gegeneinander antreten können, ist auch ein System. Über das Smartphone kann dann noch Standortinformation hinzugefügt werden oder festgehalten werden, welche Musik Sie beim Laufen gehört haben.

Aus den vielen Systemen erhalten wir ein System der Systeme: das Fitness-Tracking. Die Microservices leisten einzelne Aufgaben, wie das Erkennen, dass Sie gerade laufen. Oder einen Microservice, der auf diese Erkenntnis reagiert und die Ermittlung von Daten an Ihre Laufwettkampf-Webseite startet. Oder einen weiteren Dienst, der auf Ihrer Facebook Timeline veröffentlicht, dass Sie gerade laufen und dabei einen bestimmten Song hören.

Die Herausforderung bei den SoS besteht darin, festzulegen, was wo verarbeitet wird (siehe [SCC]). In (naher) Zukunft werden die größten Herausforderungen Benutzerfreundlichkeit, Leistungsfähigkeit, Sicherheit und Zuverlässigkeit sein.

Das Testen eines zusammengestellten verteilten IoT-Ökosystems ist eine echte Herausforderung. Das Testen funktionaler Aspekte der einzelnen Elemente (Microservices) ist problemlos möglich, aber die nicht-funktionalen Qualitätsmerkmale lassen die Zahl der Tests exponentiell ansteigen. Wir brauchen daher neue technologische Ansätze, die uns helfen, das alles angemessen zu testen.

Ein Merkmal des SoS besteht darin, dass das Testen der Gesamtlösung nur nach vorhergehender Integration aller einzelnen Systeme möglich ist. Die Herausforderung liegt in der Bedarfserfüllung, rechtzeitig mit dem Testen beginnen zu können. Die Erstellung von Modellen für die Systeme und deren Microservices kann beim frühzeitigen Testen helfen.

Dies impliziert aber die Verwendung von „Modellen von Modellen“ (engl. „Models of Models“, Kurzform: MoM). Mithilfe der Modelle kann herausgefunden werden, ob das IoT-Ökosystem funktionieren wird. Das alles ohne vollständige Bereitstellung der Funktionalität und noch ohne aufwendige Tests (z. B. mit Crowd Testing).

Funktionalität wird idealerweise auf der Ebene der Microservices getestet, während nicht-funktionale Qualitätsmerkmale für die vollständigen IoT-Lösungen mithilfe des Modells frühzeitig getestet werden können.

Drei Regeln für das IoT-Testen

Um zu verstehen, warum sich IoT-Testen vom klassischen Testen unterscheidet, müssen wir die Elemente verstehen, aus denen die IoT-Lösung besteht. Lassen Sie uns also zunächst ein IoT-Modell [IoTMap] zeichnen.

IoT-Modell

Eine IoT-Lösung lässt sich als vierschichtiges Modell darstellen (siehe Abbildung 2), von oben nach unten:

- **Die Anwendungsschicht („App“)**
  (Trend-)Daten können in einer grafischen Benutzeroberfläche wie Webseiten oder Apps auf Mobilgeräten, aber auch auf dem Touchscreen eines Dings visualisiert werden.
- **Die Datenschicht („AppData processing and platform“)**
  Alle Daten, die das Ding sammelt, werden in dieser Schicht gespeichert. Die Daten können lokal in einer Datenbank auf dem Gerät oder in einer Cloud-Lösung gespeichert werden. Intelligente Algorithmen verarbeiten

die Daten und suchen nach Korrelationen zwischen ihnen. Dies geschieht in einer plattformähnlichen Umgebung (z. B. in einer Datenbank oder in einem Betriebssystem mit echtzeitfähigen Algorithmen) auf dieser Schicht.

- **Die Brückenschicht („Bridge“)**
  Die Verbindung zwischen dem Ding und dem Datenspeicher kann viele Formen haben: Wi-Fi-Verbindung, Bluetooth, aber auch eine Kabellösung mit entsprechenden Protokollen sind Beispiele für Lösungen in dieser Schicht.
- **Die Dingschicht („Thing/device“)**
  Das Ding sammelt Daten mithilfe von Sensoren und leitet Daten über Aktoren an den Nutzer zurück.

Die Schichten arbeiten zusammen und formen eine IoT-Lösung, von unten nach oben: Ein Ding mit Aktoren und Sensoren sammelt Daten in seiner Umgebung. Über die Brückenschicht oder Schnittstelle werden die Daten an eine Datenerfassungsplattform geschickt, wo die Daten gespeichert werden. Die Daten werden jetzt verarbeitet und auf einer Benutzeroberfläche (App, Webseite, Applikation, Bedienungsfläche einer Maschine) visualisiert. Die Datenschicht kann mithilfe intelligenter Algorithmen Aktionen erzeugen, die dem Ding rückgemeldet werden, damit es entsprechend agieren kann. Alternativ kann das Ergebnis der Datenverarbeitung auch für einen Nutzer visualisiert werden.

Wir sehen, dass die verschiedenen Schichten innerhalb der IoT-Lösung ständig miteinander kommunizieren. Um eine Trennung zwischen den Schichten herzustellen, empfiehlt es sich, eine stabile Schnittstelle pro Schicht zu erzeugen. Diese sogenannten Programmierschnittstellen (engl. „Application Programming Interface“, Kurzform: API) können ebenfalls genutzt werden, um eine Schicht isoliert zu testen. Treiber und Platzhalter am Anfang und Ende von klar definierten Schnittstellen sind leicht zu erstellen und ebenso leicht lässt sich relevanter Input und Output für eine Schicht prüfen.

Nach außen kann eine IoT-Lösung eine Schnittstelle haben, da wir sie möglicherweise mit anderen IoT-Lösungen (von Drittanbietern) verknüpfen wollen. In diesem Fall entspricht ein Software Development Kit (SDK) der Luxusversion eines API. Das SDK liefert eine gut dokumentierte Schnittstelle, über die kommuniziert werden kann, aber auch eine Umgebung für die Zusammenstellung neuer IoT-Lösungen.

Im IoT-Schichtenmodell befindet sich diese in der Anwendungsschicht. Für eine Anwendungsschicht existieren typischerweise zahlreiche SDK. Dies vereinfacht das Kombinieren bestehender IoT-Lösungen zu neuen IoT-Lösungen.

IoT-Testen

Das vorgestellte IoT-Modell bringt uns einige Erkenntnisse:

1. Eine IoT-Lösung ist in ihre Einzelteile aufzuteilen, damit eine klare Sicht auf die einzelnen Systeme und Elemente entsteht. Das kann durch die Definition von Microservices erfolgen. Idealerweise werden diese Elemente funktional getestet. Fehler können in einem frühen Stadium aufgespürt werden und die Möglichkeiten, die getestet werden müssen, halten sich noch in Grenzen. Wenn man die gesamte IoT-Lösung betrachtet, ist die Anzahl der Möglichkeiten, die getestet werden müssen, schier endlos (d. h. steigt schnell exponentiell an). Aus Gesamtsicht der IoT-Lösung sind jedoch andere Qualitätsaspekte wichtiger (z. B. Benutzerfreundlichkeit, Leistungsfähigkeit, Zuverlässigkeit). Der Nutzer möchte die komplexe Zusammenstellung einer IoT-Lösung nicht bemerken und erlebt diese nicht-funktionalen Qualitätsattribute stärker als funktionale Aspekte. Nehmen Sie zum Beispiel ein Lenkrad in einem Rennwagen: Als Einzelelement kann es funktional getestet werden, aber in einem Rennwagen mit einem Fahrer auf verschiedenen Rennstrecken und bei unterschiedlichen Wetterbedingungen ergeben sich unendliche Testmöglichkeiten aufgrund verschiedener Umgebungsparameter. Die wirkliche Leistungsfähigkeit (Beschleunigung, Höchstgeschwindigkeit) des Rennwagensystems ist sehr viel wichtiger als die Funktionalität eines einzelnen Schalters am Lenkrad. **Ein IoT-System wird weniger auf Funktionalität getestet und der Fokus verschiebt sich zu anderen Qualitätsattributen.**
2. IoT-Lösungen lassen sich von Nutzern häufig nicht direkt als solche erkennen. Die IoT-Lösung unterstützt und verbessert aber ein spezifisches Benutzererlebnis. Es ist diese subjektive Erfahrung, die schwierig zu testen ist. Wie kann man eine solche IoT-Erfahrung testen? [IoTMap] beschreibt dafür eine Reihe von IoT-spezifischen Bausteinen und zeigt einige vorhandene Testbausteine in einem IoT-Kontext. **Ein Unterschied zu klassischem Testen ist, dass beim IoT-Testen die subjektive Erfahrung (das IoT-Benutzererlebnis) zu testen ist.**
3. Die vier Schichten des Modells entsprechen einzelnen Expertisebereichen. Für die Entwicklung eines selbstfahrenden Fahrzeugs sind z. B. nicht nur Kraftfahrzeugkenntnisse erforderlich, sondern auch Kenntnisse im Bereich der Videobearbeitung und der Telematik. Das Testen von jedem dieser Bereiche erfordert eine spezifische Expertise. **Das Testen der IoT-Lösung in ihrer Gesamtheit erfolgt durch das Kombinieren von Testexpertisen. Nur so kann eine umfangreiche Abdeckung erreicht werden.**

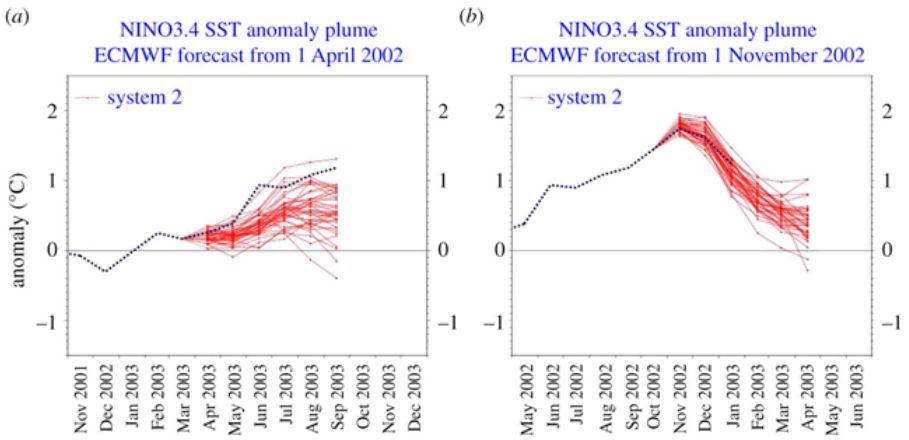

Abb. 3: Die sogenannte “Feder”-Prognose: mit der Zeit nimmt die Unsicherheit der Multimodellergebnisse zu.

Evolutionäre Algorithmen helfen beim IoT-Testen

An diesem Punkt kommt künstliche Intelligenz (Kurzform: KI) für das Auswählen

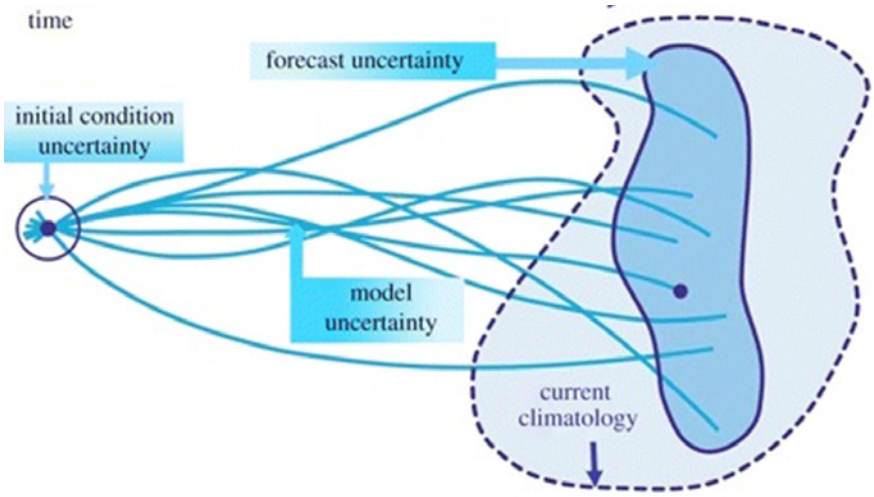

Abb. 4: Multimodell-Vorhersagen gewähren mit entsprechender Unsicherheit Einblick in künftige Ergebnisse [WFP].

von Umgebungsszenarien für die Testumgebung ins Spiel. KI ist ein weites Feld. Viele existierende Algorithmen können als KI klassifiziert werden. Ein evolutionärer Algorithmus (Kurzform: EA) ist ein Beispiel für KI und wird vorrangig zur Optimierung oder Suche eingesetzt. EA ist ein naturanaloges Optimierungsverfahren und nutzt Operatoren wie Mutation, Rekombination und Selektion.

Alles abdecken

Systeme der Systeme zeichnen sich dadurch aus, dass die Zahl der Kombinationen von Situationen (oder Konfigurationen) sehr hoch (d. h. nahezu unendlich) ist. Es ist unmöglich, alle Kombinationen in der üblicherweise vorhandenen Zeit zu testen. EA-Algorithmen können helfen, den Prozess der Auswahl möglicher Situationen zu beschleunigen – mit dem Ziel, Kombinationen zu finden, auf die wir verzichten könnten. Zum Beispiel: nach maximaler Belastung des SoS oder nach Timing-Problemen zu suchen.

Bei all dem muss uns bewusst sein, dass mit EA ein sogenanntes lokales (und nicht globales) Maximum und lokales Minimum gefunden werden können. Damit können wir uns nicht sicher sein, alles abgedeckt zu haben. Mit einem schnellen Algorithmus können jedoch einige Zeitprobleme aufgedeckt werden, die sonst erst später in der Praxis festgestellt worden wären. Das ist aber schon ein erheblicher Vorteil.

Multimodell-Vorhersagen

IoT-Lösungen oder SoS agieren im Kontext ihrer eigenen Umgebung oder in der Umgebung der jeweils verbundenen IoT-Lösung, wodurch unerwünschte Seiteneffekte ausgelöst werden können.

Nehmen wir das Beispiel eines Fahrzeugs mit adaptivem Tempomat. Der Tempomat empfängt Radarinformationen von der Fahrzeugfront und stellt anhand dieser Informationen fest, welche anderen Objekte sich vor dem Fahrzeug befinden, wie weit diese entfernt sind und wie schnell sich diese relativ zum eigenen Fahrzeug bewegen. Das alles ist zu testen und in einem Zulassungsverfahren erfolgreich zu bestätigen.

Stellen Sie sich nun vor, es kommt ein neues Fahrzeugmodell auf den Markt mit einer neuartigen Fahrhilfe. Es könnte sich um eine andere Art von Radar handeln oder um ein völlig neues Konzept automatischer Fahrassistenzsysteme. Es besteht jedoch die Möglichkeit, dass das neuartige Fahrerassistenzsystem die Radarsignale Ihres Fahrzeugs stört oder beeinträchtigt. Die Anzahl der möglichen Situationen, mit denen Ihr bisheriges SoS oder Ihre bisherige IoT-Lösung kombiniert werden kann, erweitert sich nun.

Wenn man das Verhalten eines Systems (oder SoS) in seiner Umgebung betrachten will, werden häufig Modelle verwendet. Das empfiehlt sich, um Konzepte in einer frühen Phase ausprobieren und Fehler feststellen zu können. Im Beispiel des adaptiven Tempomats wollen wir viele Modelle unterschiedlicher Umgebungen mit dem Modell des adaptiven Tempomats kombinieren. Dies ist besonders beim Testen in einem frühen Stadium hilfreich. Durch den Einsatz von Modellen werden deutlich weniger Crash-Tests benötigt, um zu beweisen, dass die Verwendung des adaptiven Tempomats sicher ist.

Neue Fahrhilfen kommen auf den Markt, aber auch neue Verkehrssituationen treten auf, in denen die Nutzung des adaptiven Tempomats sicher sein muss. Die Umgebungsmodelle werden im Laufe der Zeit immer zahlreicher. Selbst wenn der adaptive Tempomat nicht aktualisiert wird, sollte ein regelmäßiger – z. B. wöchentlicher – Test mit neuen Modellen durchgeführt werden.

Singularität (aus der Meteorologie) [SVV]

Der Begriff bezeichnet eigenartige Witterungsregelfälle in der Meteorologie. Dieses Konzept kann in IoT-Umgebungen hervorragend genutzt werden.

Warum nicht ständig neue Umgebungsmodelle entwickeln und einen EA verwenden, um sich Testfälle für extreme Situationen ausdenken zu lassen?

Sie sollen beweisen, dass die IoT-Lösung noch immer sicher ist und keine systemkritischen Grenzen überschritten werden. Wir können sogar so weit gehen und eine Art „predictive modelling situation“ schaffen, in der wir bewusst nach diesen Situationen suchen.

Genau das geschieht bei der Wettervorhersage. Hier laufen Modelle mit vorgegebenen Parametern ab, um etwa herauszufinden, wie hoch die Temperatur in einigen Monaten sein wird (siehe Abbildung 3, [WFP]).

Nehmen wir an, wir wenden das Konzept auf das IoT-Testen an. In dem Fall sind die Testingenieure zunehmend damit beschäftigt, die richtig modellierten Umgebungen zusammenzusetzen, die mit dem Modell eines SoS interagieren sollen. Sie spielen all diese Kombinationen durch, um herauszufinden, welche möglichen Endsituationen für Qualitätsattribute wie Stabilität (ist eine Situation denkbar, in der ein SoS zusammenbrechen kann?) oder Zeitverhalten (wird zeitlich abgestimmte Information rechtzeitig übermittelt?) möglich sind. Der Testingenieur wird zu einer Art Technologiemeteorologe.

In Zukunft können Testingenieure laufende Systeme und die Qualität ihrer Arbeit im Blick behalten. Sie werden mit solchen Bildern wie in Abbildung 4 arbeiten und auch solche Bilder erstellen. Nimmt die Zahl der intelligenten Systeme weiter zu, könnte es sogar so weit kommen, dass der Testbereich sich hin zur Überwachung des Intelligenzgrades von SoS verschiebt.
Der Testingenieur ist der „Meteorologe für Singularität“ der Zukunft!

IoT benötigt intelligente Testfälle

Das Internet der Dinge mit seinem System der Systeme liefert uns die Herausforderung

unendlicher Testkombinationen und sich ständig ändernder (Test-)Umgebungen. Die Durchführung von Tests in IoT-Umgebungen erfordert einen angepassten Testansatz:

- Weg vom funktionalen Testen hin zum Testen nicht-funktionaler Qualitätsattribute.
- IoT-Testen hat mit dem Testen von subjektiven Erfahrungen und vom IoT-Benutzererlebnis zu tun.
- Die verschiedenen (Test-)Expertisen für das vierschichtige IoT-Modell müssen kombiniert werden.

Andere Techniken wie Künstliche Intelligenz können uns beim Testen von IoT-Lösungen helfen. Testfälle können mit einem evolutionären Algorithmus erzeugt werden. In Kombination mit Multimodell-Vorhersagen für komplexe Umgebungen werden Testfälle intelligent.

Die Zukunft sieht Testingenieure als Technologiemeteorologen und diese steuern die Nutzung intelligenter Testfälle. Diese Testfälle werden so lange weiterentwickelt, bis sie für das Testen der IoT-Lösung optimiert sind.

Referenzen

[TSE] Tom van de Ven und Alexander van Ewijk, Der Testaufwand steigt exponentiell durch das Internet der Dinge (IoT), Artikel im objektspektrum.de, http://www.sigs-data-com.de/fachzeitschriften/objektspektrum/archiv/artikelansicht/artikel-titel/der-testaufwand-steigt-exponentiell-durch-das-internet-der-dinge-iot.html

[SCC] Alur R., Berger E., Drobnis A. W., Fix L., Fu K., Hager G. D., . . . Zorn B. (2015). System Computing Challenges in the Internet of Things: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association. http://cra.org/ccc/resources/

[UMS] Ben Linders (2015), Using Microservices in the Internet of Things, blog on https://www.infoq.com/news/2015/12/microservices-iot

[IoTMap] Tom van de Ven (2016), IoTMap: Testing in an IoT environment, Book published by: Kleine Uil, ISBN: 9789075414868

[MLA] Samuel, Arthur L (1959). „Some Studies in Machine Learning Using the Game of Checkers," IBM Journal of Research and Development, 44:1.2, 210–229

[WFP] Julia Slingo, Tim Palmer (2011) "Uncertainty in weather and climate prediction" article in the discussion Meeting Issue ‚Handling uncertainty in science' http://rsta.royalsocietypublishing.org/content/369/1956/4751

[SVV] Vernor Vinge (1993) "Singularity" essay, Department of Mathematical Sciences San Diego State University