OBJEKTspektrum
ONLINE Themen special

Fachartikel

IoT Testing? Challenge accepted!

Das Internet of Things - eine Herausforderung für Test-Athleten

von Ron Werner und Bastian Baumgartner

Bei bald 20 Milliarden verbundener Devices im Internet of Things (IoT) wird es für Tester garantiert nicht langweilig. IoT ist mit Alexa und Co. bereits in heimischen Wohnzimmern angekommen und verknüpft Computer, Sensoren, Netzwerke, Big Data und Machine Learning. Diese interdisziplinäre Ausprägung weckt hohe Erwartungen, nicht zuletzt an die Qualität. Welche Skills, Methoden und Werkzeuge für diesen Dekathlon der unterschiedlichen Testarten nötig sind, erklären wir hier und verdeutlichen dies mit Praxisbeispielen.

Einführung in die Welt der Things

Abb. 1: Things und deren Eigenschaften

Das *IoT (Internet of Things)* steht für eine durch sichere Netzwerkverbindungen und Cloud-Infrastruktur unterstützte M2M (Machine to Machine)-Technologie, die Daten in nützliche Informationen umwandelt und dadurch Mehrwert stiftet. *Things* sind smarte, kommunikationsfähige Produkte, die mit Sensoren und Software ausgestattet sind, wie Abbildung 1 zeigt. Oft werden Things auf Hardware reduziert, allerdings spielt die Software hierbei eine sehr wichtige Rolle.

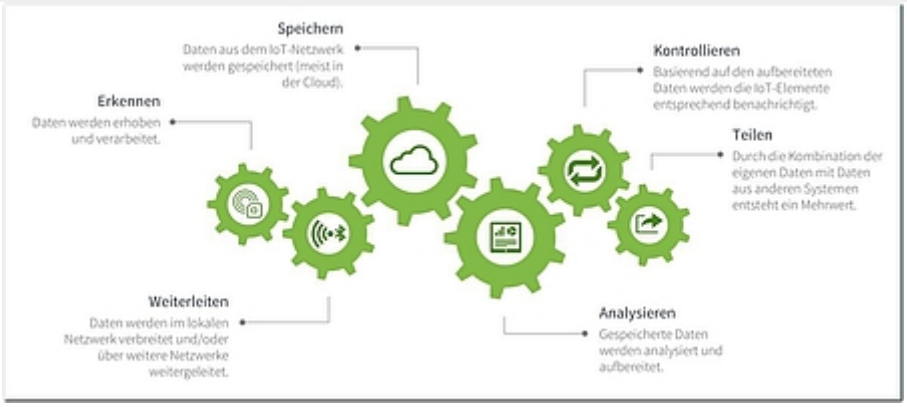

Abb. 2: Hauptaktionen von IoT-Elementen

Die Aufgaben der Elemente eines IoT-Ökosystems sind vielschichtig. Wie Abbildung 2 veranschaulicht, können diese auf sechs Hauptaktionen heruntergebrochen werden: Erkennen, Weiterleiten, Speichern, Analysieren, Kontrollieren und Teilen von Daten.

IoT zuhause und unterwegs

Nun aber genug der Definition: IoT ist schon ganz praktisch im heimischen Wohnzimmer angekommen. Ihre Nachbarn steuern ihr Abendprogramm mit Alexa oder Google Home und nutzen smarte Heizungs-Thermostate wie Google Nest oder Tado. Sie hingegen erfreuen sich vielleicht schon an einem vernetzten Backofen, Kühlschrank und anderen Hausgeräten und tracken dann noch Ihre Fitness über Wearables, wobei die Daten vom Handgelenk aus direkt in die Cloud gespeichert werden. Aber auch die Industrie schläft nicht und hat diese Anwendungsmöglichkeiten schon längst für sich entdeckt.

Während noch viel über Industrie 4.0 diskutiert wird, sind IoT-Anwendungen bereits weit verbreitet:

> Connected Cars (bessere Navigation durch vernetzte Autos, erweiterte Sicherheitsfunktionen und Komfortmöglichkeiten),
> Flotten-Management (Senkung von Wartungs- und Reparaturkosten),
> Predictive Maintenance (Reduzierung von Ausfallzeiten bei Maschinen),
> selbstoptimierende Produktion (Echtzeitüberwachung und Optimierung von Produktionsprozessen),
> automatisches Bestands-Management (Monitoring von Lagerbeständen und der Supply Chain),
> Fernüberwachung und Diagnose von Patienten (Ärzte interagieren mit Patienten und Live-Diagnosedaten),
> Smart Metering (Verbrauch von Strom, Gas und Wasser).

Diese Entwicklung stellt uns Tester auf den Prüfstand – wir können zwar vieles aus der bekannten Testwelt adaptieren (besonders aus dem Mobile Testing), mit Sicherheit lässt sich allerdings nur sagen, dass IoT Testing eine große Herausforderung darstellt, die den modernen Tester mit all seinen Skills benötigt.

Die Challenges: IoT Testing als moderner Dekathlon

Laut „World Quality Report 2017-18“ [Wqr] befassen sich 83 Prozent der befragten Unternehmen aktiv mit dem Thema IoT. Über die Hälfte der Befragten fühlen sich mit einer schier unermesslichen Anzahl von Testsituationen überfordert und glauben, dass Lösungen unter Einsatz von KI (Künstliche Intelligenz) und ML (Machine Learning) zur Bewältigung dieser Herausforderung nötig sind, um ausführlich genug zu testen. Der Survey weist im Vergleich zu den Vorjahresergebnissen allerdings nur eine Steigerung von 4 Prozent bei den Herausforderungen im Bereich IoT Testing auf. Weiterhin möchte die Hälfte der Befragten gerne mehr "IoT-Experience Testing" (aus Sicht der Endbenutzer) als nur rein funktionales Testing durchführen.

Sie merken, IoT wird als zunehmend komplexes Thema wahrgenommen, das einerseits Unterstützung durch smarte Tools erfordert. Andererseits wünschen Entscheider in der Industrie sich „End-to-End IoT Experience"-Testszenarien, die Interaktionen über verschiedene Stufen erforderlich machen: die "Things", der Kommunikations-Layer, der Datenspeicherungs- und Analytik-Layer sowie alle Applikations- und Benutzer-Schnittstellen.

Testing für IoT ist wie ein *Dekathlon* (aus dem Griechischen: „Zehnkampf“) der Testdisziplinen. Wir benötigen unterschiedlichste Fähigkeiten des modernen Testers auf menschlicher Ebene sowie diverse Testpraktiken und Tools auf methodischer Ebene, die für einen erfolgreichen Testabschluss eingesetzt werden müssen. Aber lassen Sie uns doch jetzt einen näheren Blick auf die

Challenges werfen.

Ready Player One? Die größten IoT Testing Challenges und mögliche Lösungsansätze

Sicherheit und Datenschutz

Die Europäische Datenschutz-Grundverordnung (GDPR – General Data Protection Regulation) tritt gerade in Kraft, und noch immer rütteln uns IoT-Sicherheits- und -Datenskandale auf. Diese Ereignisse lassen die Welt spüren, dass Datensicherheit eines der wichtigsten und risikoreichsten Themen für IoT ist. Damit sind bei sicherheitskritischen Systemen die Sicherheit für Leib und Leben sowie Besitz verknüpft, die Firmenreputation und damit schließlich auch finanzielle Werte (Risiken des Verlusts von Kunden und Geschäftspartnern, Vertragsstrafen, rechtliche Implikationen).

Im Bereich IoT für Endkunden erschütterte Anfang 2017 das Bekanntwerden eines Falls von IoT-fähigen (d. h. mit Bluetooth, Lautsprechern, Mikrofon und Cloud-Datenspeicherung versehenen) Kuscheltieren der Firma Cloudpets, mit der Eltern ihren Kindern persönliche Sprachnachrichten hinterließen, die diese über ihren Teddybär und IoT-Anbindung anhören konnten. Da die Cloud-Datenbank ungesichert war, konnten Hacker die Voice-Daten sehr einfach stehlen, die Datenbank durch Verschlüsselung unzugänglich machen und stellten dann Lösegeldforderungen an die Firma Cloudpets (vgl. [Net]).

Im Jahr 2016 machten Hacker Schlagzeilen durch ein Mirai-Botnet, das mit einem konzertierten DDoS-Angriff (distributed denial of service) das Internet für weite Teile der USA lahmlegte. Schuld daran waren hardgecodete Passwörter in der Firmware von chinesischen Webcams der Firma Xiongmai, die weder geändert noch deaktiviert werden konnten (vgl. [Gua]).

Kategorie	IoT-Sicherheitserwägungen
I1: Unsicheres Web-Interface	Schwache Passwörter, Konto-Lockout nach mehreren Login-Fehlschlägen, Konto-Konfigurationsmöglichkeiten, Web-Interface Vulnerabilitäten (XSS, SQLi and CSRF), Schutz durch Firewalls
I2: Ungenügende Authentifizierung/Autorisierung	Erforderlichkeit starker Passwörter, Zwei-Faktor-Authentifizierung, Kennwortwiederherstellung, Passwortänderungszyklen, Default-Passwörter
I3: Unsichere Netzwerk-Services	Schutz gegen Angriffe, die Buffer Overflow, Fuzzing, Denial-of-Service-Angriffe und offene Ports ausnutzen
I4: Keine Transport Encryption	Transportverschlüsselung zwischen Things, den Backend-Servern und Daten verarbeitenden Systemen, Vermeidung proprietärer Protokolle, Verwendung von Firewalls
I5: Datenschutz	Werden nur notwendige Daten gesammelt und verarbeitet (Datensparsamkeit), verschlüsselt übertragen und gespeichert und wird die Erlaubnis der Endnutzer darüber eingeholt? Können Benutzer ihre Daten anfordern und einsehen sowie löschen lassen (GDPR)?
I6: Unsichere Cloud-Interfaces	Sind bekannte Sicherheitslücken bei Cloud-Interfaces gepatched? Werden die unter I1 und I2 genannten Punkte beachtet? Wird Transport Layer Security (TLS) verwendet?
I7: Unsichere Mobile Interfaces	Werden die unter I1, I2 und I6 genannten Punkte beachtet?
I8: Ungenügende Security-Konfigurationsmöglichkeiten	Sind Passwortsicherheits-Optionen verfügbar (Zwei-Faktor-Authentifizierung, starke Passwörter)? Sind Verschlüsselungsoptionen gegeben und genutzt (z. B. Verwendung von AES-256 statt AES-128)? Gibt es Security Event Logging? Werden Benachrichtigungen für Security Events verschickt?
I9: Unsichere Software/Firmware	Können Devices schnell mit Sicherheitsupdates gepatched werden? Benutzen sie verschlüsselte und signierte Updates, und werden diese verschlüsselt übertragen?
I10: Ungenügende Physikalische Security	Minimalanzahl von externen Ports, um physikalischen Zugriff zu restriktieren? Können zusätzliche Ports abgeschaltet werden? Sind Admin-Konsolen nur über ein lokales Interface erreichbar?

Tabelle 1: Guideline für das Security Testing von IoT (vgl. [Owa])

Und sicher erinnern Sie sich an die IP-Kameras von Aldi, die 2015 ebenso ungenügend gesichert waren und Unbefugten ermöglichten, über das Internet an die Bilddaten zu gelangen und die Kameras zu steuern (vgl. [Hei]). Die Liste der Security Breaches ist beliebig verlängerbar und erweitert sich fast im Wochenzyklus.

Welche Besonderheit weist das Testing hier auf? Wir unterscheiden vier verschiedene Security-Layer-Tests:

1. Tests der Device Security der Hardware und Software der einzelnen Things (Sensoren, Aktuatoren und Mikrocontroller),
2. Web-Service-Security-Tests auf API-Ebene,
3. Datenbank- und Application-Security-Tests,
4. übergreifendes Penetration Testing der Infrastruktur.

Security-Experten erstellen hierfür zu Beginn eine Threat-Risk-Analyse, in der die wichtigsten Angriffsvektoren und Angriffsflächen analysiert und bewertet werden. Einen ersten Überblick verschaffen Sie sich hierfür mit dem Projekt OWASP IoT [Owa], das Basisrisiken aufzeigt und Guidelines für das Testing gibt – diese sind aufgrund der Neuheit des Themas teilweise noch im Draft-Status, verschaffen aber einen guten Einstieg (siehe Tabelle 1).

Security und Datenschutz sind unabdingbar und müssen mehrschichtig durch Experten in Hardware-, Software-, API- und Transport-Layer-Sicherheit getestet werden. IoT-Cloud-Services von Drittparteien wie AWS IoT, Microsoft Azure IoT oder Google Cloud IoT müssen dabei genauso unter die Lupe genommen werden wie eigens gehostete Implementierungen. Als „Otto Normaltester“ kann man mit dem notwenigen Hintergrundwissen und frei verfügbaren Tools (u. a. Wireshark, tcpdump, mqtt-spy, IoT-Testware) aber bereits in eine der wichtigsten Disziplinen im Dekathlon des IoT Testing einsteigen.

Skalierbarkeit und Performance

Weiter geht's: Gartner schätzt, dass es 2020 weltweit über 20 Milliarden Devices geben wird [Gar]. Dies ist ein erster Vorgeschmack auf die kombinatorischen Möglichkeiten für Testfälle. Wir kennen diese Komplexität bereits aus dem Mobile Testing, bei dem Fragmentierung ein großes Problem ist: Die große Zahl von unterschiedlichen Smartphones und Tablets in Kombination mit Betriebssystem-Versionen und -Flavours sowie Screengrößen bewirkt Komplexität im Testing. Dies bekommen wir in der Praxis durch cleveres Einteilen in Device-Gruppen (High-End-, Standard- und Low-End-Hardware) in den Griff – eine Priorisierung auf das Wesentliche.

Es wäre sehr aufwendig, ein adäquates Setting für das Testing von komplexen Szenarios wie IoT-Öl-Pipeline-Sensoren (das Szenario, für das bereits 1999 das MQTT-Protokoll entwickelt wurde) live aufzubauen. Es müsste so produktionsnah wie möglich sein, um aussagekräftige Performance-Tests zu ermöglichen.

Was tun? In diesem Fall hilft Virtualisierung des Setups und Simulation der Sensoren. Über den Transport-Layer können virtuelle Tools sensorische Testdaten schicken, ohne dass eine solche Unmenge an Sensoren verwendet werden muss. Für eine Virtualisierung können Cloud-Services verwendet werden, um einen entsprechend hohen Skalierungsgrad zu erreichen.

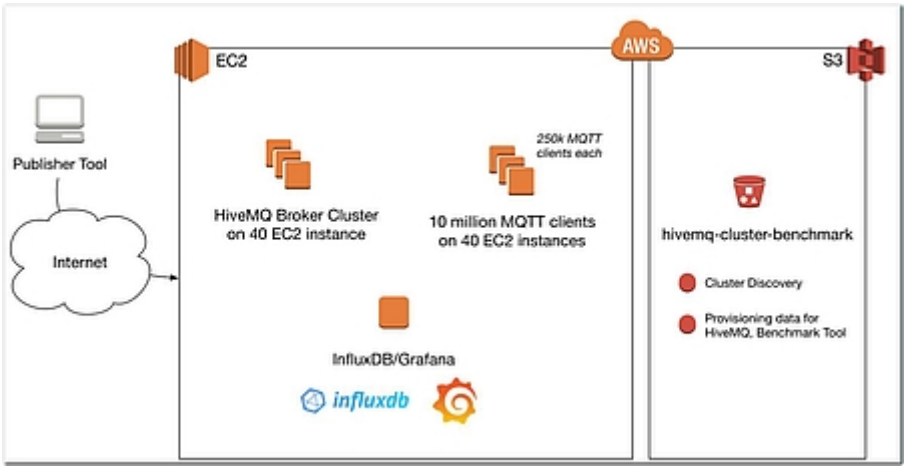

Abb. 3: MQTT Message Broker Performance Test Setup (mit freundlicher Genehmigung der dc-square GmbH)

Wie so etwas in der Praxis aussieht, konnten wir beim Performance- und Skalierbarkeits-Testing eines hochskalierbaren MQTT-Message-Brokers miterleben. MQTT ist ein weitverbreitetes, leichtgewichtiges Messaging-Protokoll, welches speziell für IoT entwickelt wurde. Gegenüber herkömmlichen Client-Server-Protokollen wie HTTP arbeitet MQTT mit einem ereignisorientierten Publish-Subscribe-Modell. Der Client muss nicht ständig beim Server anfragen, ob neue Daten existieren – ein Broker informiert die Consumer, wenn neue Daten zu einem bestimmten Topic vorliegen.

Die *dc-square GmbH* ist einer unserer Kunden und hat den MQTT-Message-Broker *HiveMQ* entwickelt. Die Firma hat zu

Benchmarking-Zwecken ein Setup aufgestellt, mit dem 10 Millionen gleichzeitiger Verbindungen in verschiedenen Konfigurationen über den Zeitraum von 30 Minuten getestet wurden.

Dazu installierten sie ein HiveMQ-Cluster mit 40 Nodes in der Cloud auf Amazons AWS-Instanzen, erstellten dort 40 weitere MQTT-Client-Instanzen und skalierten die Anzahl der subscribed Messages solange nach oben, bis 10 Millionen erreicht waren (siehe Abbildung 3). Lassen Sie sich das ruhig einmal auf der Zunge zergehen: Dies entspricht der 2,5-fachen täglichen Menge an weltweiten WhatsApp-Nachrichten 2014. Dabei wurden massive Durchsatzwerte von über 1,7 Millionen Nachrichten pro Sekunde erreicht. Auf den Client-Instanzen liefen Benchmarking-Tools mit, eine weitere Monitoring-Instanz mit InfluxDB (eine Zeitreihen-Datenbank) und Grafana (ein Analyse- und Monitoring-Werkzeug) ermöglichte mithilfe von Terraform (ein weiteres Tool für die Bereitstellung der Benchmark-Infrastruktur in der Cloud) das Mitschneiden und Interpretieren der Live-Daten (vgl. [Hmq]).

Dieses beeindruckende Beispiel zeigt, dass mithilfe einer Kombination von Cloud-Services, Open-Source-Tools und eigener Programmierung sehr aussagekräftige IoT-Skalierbarkeits- und Performance-Tests möglich sind.

Netzwerk und Connectivity

Innerhalb eines IoT Setups gibt es verschiedenste Netzverbindungen, Daten- und Transportprotokolle zu testen. Die „letzte Meile“ vom Thing zum Empfänger ist oft die entscheidende, da hier im Feld schlechte Netzbedingungen (ungenügende Reichweite, schwache Signale durch Abschirmung und Hindernisse, Störfrequenzen) und Latenzen zu erwarten sind.

Unter den Netzverbindungen gibt es WLAN, BLE, Bluetooth, NFC und RFID, auf Daten- und Transportprotokollebene gibt es eine Vielzahl unterschiedlicher Protokolle wie MQTT, CoAP, Zigbee, Z-Wave und weitere. Die Protokollebene lässt sich durch bewährte Tools wie Wireshark testen, mit denen der komplette Netzwerk-Traffic von einem Punkt zum anderen mitgeschnitten wird. Mit dem Mitschnitt können Sie, wie Sie es vom Testen anderer Protokolle wie REST/SOAP via HTTP gewohnt sind, Testmessages im Last- und Performance-Tool Ihrer Wahl aufsetzen und senden. Hier bieten sich etwa die MQTT oder CoAPApache JMeter Plugins und MQTTBox an, oder kommerzielle Tools wie Loadrunner. Um Paketverluste und unzuverlässige Verbindungen zu simulieren, gibt es weitere Tools wie Comcast (https://github.com/tylertreat/comcast).

Hier haben sich Crowdtesting-Ansätze wiederum bewährt. So stand einer unserer Kunden, der Hersteller einer mobilen, verschlüsselten Kommunikations-App, vor nicht nachvollziehbaren Synchronisations-Fehlern in einer Teststellung. Die Fehler konnten lange Zeit nicht reproduziert werden – bis wir schließlich in einem "Bug Bash", also einer intern beim Kunden organisierten Fehlersuche, das Problem im Feldtest nachgestellt bekamen, und die Ursache endlich gefunden und behoben werden konnte. Hier war der Root Cause der Übergang von WLAN zu einer Mobilverbindung mit schlechtem Durchsatz (EDGE), die bewirkte, dass die Mailserver-Synchronisation clientseitig fehlerhaft persistiert wurde.

In Ihrem Fall kann es an ganz anderen Ursachen liegen – herausfinden werden Sie es am besten mit User-Tests unter realistischen Umständen, nämlich durch "Testing in the wild", gepaart mit Crowdtesting-Ansätzen (Dogfooding: interne Feldtests unter Einbeziehung der Mitarbeiter als "own crowd"). Wir garantieren Ihnen, dass Ihre Kollegen Spaß dabei haben werden!

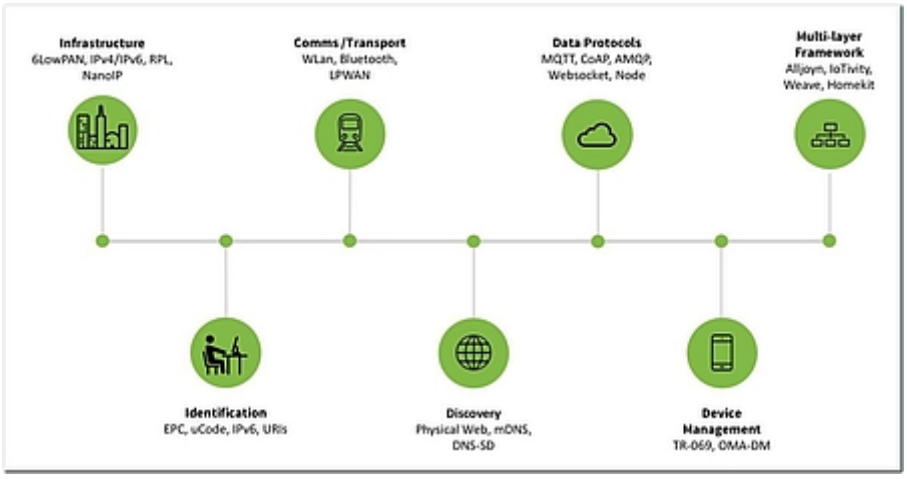

Abb. 4: Auszug aus den meistgenutzten IoT-Protokollen

Viele Standards, kaum Harmonisierung und Tools

Bestimmt erinnern Sie sich an die Anfangszeiten von Webseiten und später auch Mobile Apps – in den Pionierzeiten gab es noch keine industrieseitige Einigung auf Standards. Der Netscape-Browser, XHTML, Symbian und Blackberry OS kamen und gingen. Im IoT-Umfeld ist dies die nächste Herausforderung für Tester. Zugegeben, es gibt Ausnahmen wie das Projekt Eclipse IoT oder das MQTT-Nachrichtenprotokoll, das bereits 1999 entwickelt wurde und dessen neueste Version 5 de facto bald zum anerkannten Standard wird. Trotzdem weist die Tool-Landschaft für das Debugging und Testing von IoT-Lösungen noch überwiegend schwarze Flecken auf, und Testwerkzeuge müssen individuell programmiert werden.

Durch fehlende oder noch nicht klar definierte Standards gestaltet es sich auch für die Qualitätssicherung schwierig, Standards für Testmethoden oder Tools festzulegen. Tester stehen vor der Herausforderung, sich durch eine Flut von Vorgaben ackern zu müssen, dazu erschweren Versionsunterschiede und fehlerhafte Umsetzungen der Vorgaben beziehungsweise inkompatible Versionsstände bei den Devices die Qualitätssicherung. Abbildung 4 listet die meistgenutzten Protokolle aus [Pos] in Abhängigkeit zum Einsatzbereich auf.

Daraus resultieren viele Möglichkeiten, wie Things untereinander und mit dem Backend kommunizieren, was fürs Testing viel Wissen über alle beteiligten Protokolle erfordert, und Testen über Protokollgrenzen hinaus erforderlich macht.

Realistisches End-User Testing

End-User Testing ist im IoT und Mobile Testing äußerst wichtig. Bei dieser Testart werden Things im realen Kontext der Endbenutzer getestet, als sogenannte "Beta"- oder "Friendly User"-Tests. Die Ergebnisse geben einen Vorgeschmack auf den Erfolg des Produkts bei der Zielgruppe. Natürlich müssen die zu testenden Komponenten bereits erfolgreich niedrigere Teststufen durchlaufen haben und entsprechend gute Qualität vorweisen, um im End-User-Test eingesetzt werden zu können. Wir empfehlen, diese Methode unbedingt vor einem anvisierten Launch-Termin zu berücksichtigen.

Wir haben die Erfahrung gemacht, dass dabei aufgrund der schier endlosen Konfigurations- und Kombinationsmöglichkeiten eingesetzter Hardware in Haushalten Fehler aufgedeckt werden, die in früheren Teststufen nicht gefunden wurden. In unserem Fall waren wir für einen großen deutschen Hausgerätehersteller tätig – Geschirrspülmaschinen, Öfen, Waschmaschinen und Kaffeemaschinen waren via App auf dem Smartphone des Nutzers verbunden. Als während der End-User-Testphase eine ganztägige Downtime wegen eines Server-Crashs zu verbuchen war, wurde als Ursache ein Emoji im Gerätenamen eines der benutzten Smartphones (ein echtes Benutzer-Device) identifiziert. Dies ist im Labortest nicht aufgefallen.

Weitere Erfahrungen konnten wir in einem Connected-Car-Projekt eines großen deutschen Automobilherstellers sammeln: Hier waren wir für das Testing einer App verantwortlich, über die Fahrzeuge lokalisiert, gebucht und genutzt werden konnten. Ein wichtiger Use Case war die Ortung eines Fahrzeugs via GPS, die Anzeige der aktuellen Position, Entfernung, Route und weiterer Angaben zum Fahrzeug und der Strecke.

Wir arbeiteten hier mit Crowdtesting mit echten Fahrzeugen, welches wir nach dem User Acceptance Test (UAT) immer wieder als ergänzende Methode vor Go Live einsetzten. In jedem zweiwöchigen Testzyklus erhöhten wir die Anzahl der Crowdtester, beginnend mit 20 Nutzern, dann 200 und schließlich 2000 Nutzer. Wurde in einem Testfall angegeben, dass das Fahrzeug in einem Parkhaus zu parken ist, aber nicht klar definiert war wo genau, konnten die Fahrzeuge im Fall eines unterirdischen Parkplatzes vom System per GPS nicht mehr geortet werden. Der Standort musste über andere Kommunikationswege ermittelt werden. Die Aktionen der Nutzer mit den Things hatten damit wiederum Auswirkung auf die Interaktionen der Things untereinander, was die klare Definition von Testfällen erschwerte.

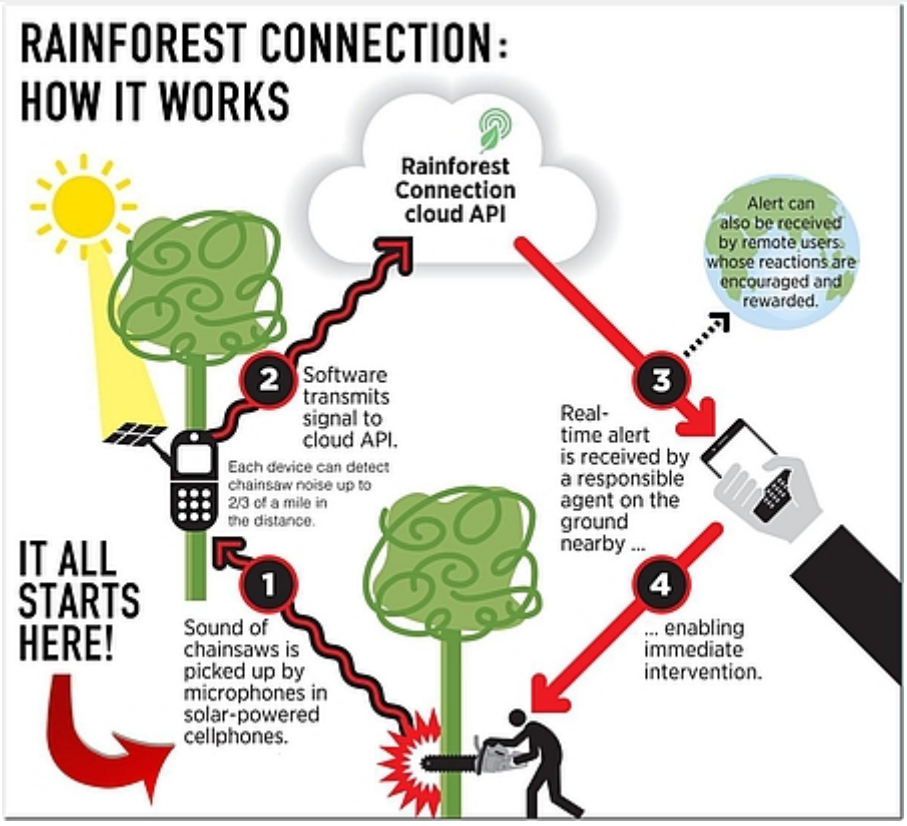

Abb. 5: Rainforest Connection – ein gemeinnütziges IoT-Projekt zum Schutz des Regenwalds (mit freundlicher Genehmigung von rfcx.org)

Analyse von Big Data mit ML und KI

Eine weitere große Herausforderung sind die Datenmengen, die von IoT-Sensoren erzeugt werden. Diese werden in Echtzeit übertragen, gespeichert und dann mittels ML- und KI-Algorithmen verarbeitet. Aber wie testet man das in Hinblick auf die Massen an *Testdaten*, die benötigt werden, sowie die *undefinierbaren Outcomes von Algorithmen*, die genauso wie kleine Kinder erst lernen und dabei auch Fehler machen sollen beziehungsweise dürfen? Kann man ML und KI aus diesem Blickwinkel überhaupt testen, und wenn ja wie?

Einen Einblick bekamen wir in dem gemeinnützigen Start-up *Rainforest Connection* (www.rfcx.org) aus San Francisco, das QualityMinds mit unserem QA Consultant Christian Krause unterstützt. Das Projekt hat es sich zum Ziel gesetzt, mittels IoT dem illegalen Abholzen von Regenwald und der Wilderei ein Ende zu bereiten, wie Abbildung 5 zeigt. Ein Netzwerk von solarbetriebenen recycelten Mobiltelefonen hoch oben in der Blätterdecke des Regenwalds zeichnet laufend Audiosignale auf, die an eine Backend-Cloud gesendet werden. Dort werden die Audiodaten mittels ML und KI analysiert, um Motorsägengeräusche und Wilderer eindeutig zu identifizieren. Im Falle eines Funds wird ein Notruf an die nächstgelegenen Ranger mit den berechneten Koordinaten des Tatorts gesendet, damit diese einschreiten können. Hierbei wird das ML-Framework *TensorFlow* von Google verwendet, um die Echtzeit-Daten mit Deep-Learning- und KI-Algorithmen zu verarbeiten.

Dazu erklärt Stefan Zapf (VP Engineering bei Rainforest Connection), dass es wichtig ist, Metriken für Akzeptanzkriterien zu definieren: „Nehmen wir an, dass wir 100.000 Sound-Dateien pro Tag erhalten. Ein Akzeptanzkriterium könnte sein, dass wir nicht mehr als einen falschen Alarm pro Tag auslösen.“ Akzeptanzkriterium wäre damit eine Fehlerquote kleiner gleich 0,001 Prozent.

„Daraufhin erstellt ein Tester ein sogenanntes Test Set. In unserem Fall ist das eine große Anzahl an Audiodateien, einige mit Kettensägen und andere ohne Kettensägen. Dieses Test Set muss auch ähnliche Geräusche wie die Kettensäge enthalten, zum Beispiel Autos oder Propellerflugzeuge. Nun passiert ein interaktiver Entwicklungsprozess der KI, wobei nach jeder KI-Entwicklung diese mit dem Test Set getestet wird. Wenn wir bei mindestens 99,999 Prozent der Dateien aus dem Test Set richtig liegen, ist das Akzeptanzkriterium erfüllt. Tester können jedoch weitere Test Cases, also weitere Audiodateien, vielleicht für Tiergeräusche aus dem Regenwald, hinzufügen, um die Qualität weiter zu sichern. Gerade weil wir immer noch Audiodateien hinzufügen können, werden wir niemals am Ende der Entwicklung sicher sein, dass wir jede Eventualität bedacht haben.“

Im Projekt wird auch Continuous Testing beziehungsweise Testing in Production praktiziert: „Wir wollen proaktiv die Detektionen

auf Akzeptanzkriterien überprüfen, da es uns erlaubt, agil auf Fehler zu reagieren und den bestmöglichen Schutz der Regenwälder zu garantieren. Gerade deswegen ist Continuous Testing ein essenzieller Bestandteil unseres Post-Production-Entwicklungsprozesses."

„Tester können jede Woche die Detektionen der KI überprüfen und überprüfen, ob die Akzeptanzkriterien eingehalten werden. Wenn die Akzeptanzkriterien nicht eingehalten werden, muss zunächst das Test Set erweitert werden mit den falsch erkannten Audiodateien. Beispielsweise haben wir, als wir mit der KI-Entwicklung anfingen, erstaunt festgestellt, dass Moskitos wegen des Dopplereffekts beim Umkreisen eines Mikrofons ganz ähnliche Geräusche erzeugen wie Kettensägen. Die Realität der Natur ist häufig komplexer und unvorhergesehener als unsere Vorstellungskraft und damit die Test Cases", so Zapf.

Mittlerweile ist die Fehlerrate bei der Erkennung enorm gering, aber gerade die nachhaltige Verbesserung auf Basis von Continuous Testing und nicht etwa die Annahme, dass man alles im Vornherein bedenken kann, führte zu qualitativem Erfolg: "Deploy early, test and improve continuously". Zapf und seine Mitstreiter betreten Neuland, es gibt kaum veröffentlichte Testing-Ansätze zu KI und ML, weswegen Testing-Methoden und -Tools durch Trial & Error selber entwickelt werden mussten.

Gesamtchallenge accepted!

Die Einzeldisziplinen sind herausfordernd, für den gesamten Zehnkampf braucht es wie bei Profisportlern mentale Vorbereitung. Um Ordnung ins IoT-Dickicht zu bringen, empfehlen wir analog zu [Cog] zunächst die Gliederung in zwei Testschichten:

Der *Device-Interaction Layer*befindet sich dort, wo Software- und Hardware-Komponenten einer Real-time-IoT-Umgebung interagieren. Dies könnte etwa die Bluetooth-Verbindung eines von vielen Beacons für Location-Based Services mit der App auf dem Smartphone eines Museumsbesuchers sein, oder die M2M-Interaktion der Things mit den Backends in der Cloud. Neben standardmäßigen funktionalen Tests sind nicht-funktionale Tests der folgenden Bereiche notwendig:

Standardkonformität,
Interoperabilität einzelner Systemteile (Hardware und Software),
Sicherheit und Datenschutz,
Qualität der Datenverarbeitung und -analyse.

Im *User-Interaction Layer* geht es um die HCI (Human-Computer Interaction)-Aspekte des IoT. Die Akzeptanz und damit der Erfolg eines Systems hängt sehr stark von der User Experience ab. Zu den Hauptgebieten dieser Tests gehören hierbei:

Performance-Tests
IoT-Experience- und Usability-Tests,
Service- und Backend-API-Tests.

Diese mentale Einteilung erleichtert das Testing von Interaktionen über die eingangs erwähnten Stufen: die Things, der Kommunikations-Layer, der Datenspeicherungs- und Analytik-Layer sowie alle Applikations- und Benutzer-Schnittstellen.

Zusammenfassung und Empfehlung

Testing im Bereich IoT ist wie ein Dekathlon (Zehnkampf, von griechisch *déka* ‚zehn' und *áthlos* bzw. *áthlon* ‚Heldentat', [Wiki]) der Testdisziplinen. Es benötigt auf menschlicher Ebene unterschiedlichste Skills des modernen Testing-Helden, sowie auf methodischer Ebene diverse Testpraktiken und Tools, die alle unter einen Hut gebracht werden müssen.

Durch die logische Einteilung der Testarten in *Device-Interaction Layer Tests* und *User-Interaction Layer Tests* kommt bereits Ordnung ins Test-Dickicht. Nach wie vor gilt aber, dass die Testbarkeit des Produkts ausschlaggebend für den Testaufwand ist. Daher sollte gute Testbarkeit bei IoT-Produkten von Anfang an mit eingeplant und einprogrammiert werden.

Um bei hochskalierbaren und verteilten Systemen schließlich auch aussagekräftige Tests durchführen zu können, sind moderne Ansätze wie Virtualisierung in der Cloud, Continuous Testing (um die produktive Infrastruktur 1:1 testen zu können) und Continuous Deployment (um mit Feature Toggles und A/B-Tests zu arbeiten) empfehlenswert. Trotzdem bleibt es Ihnen nicht erspart, die ersten Szenarien in eigenen IoT-Labs zu erstellen, die die komplette Infrastruktur ins Kleinformat runterskaliert wiedergeben. Moderner Modellbau für einen guten Zweck: Besser ein paar Things testen, als nothings.

Handlungsempfehlungen

Um IoT-Produkte im Prototyp-Status testen zu können, können Sie *eigene IoT-Infrastrukturen im Kleinformat* aufbauen. Dies kann als In-house-Labor mit echter Hardware analog zu den Open Device Labs [Ope] beim Mobile Testing sein, oder mithilfe von Cloud-Lösungen von Microsoft, Amazon und Google erreicht werden. Wir erwarten, dass sich im Laufe der nächsten fünf Jahre ähnlich wie in der Mobile-App-Entwicklung (siehe Testobject, Sauce Labs, Kobiton, Amazon Device Farm usw.) virtuelle "Thing Labs" auftun, die Fernzugriff auf echte Things und Devices inklusive der virtuellen Backend-Infrastruktur ermöglichen. Dazu müssen sich aber erst die Standards etabliert haben – und bis dahin gilt es, selbst komplette End-to-End-IoT-Testsetups nachzubauen.
Um eine bereits existierende reale Backend-Infrastruktur mit Cloud-Services, Load-Balancern und Firewalls zu testen, würde diese im klassischen Testing produktionsnah ein zweites Mal erstellt werden. Dies ist bei komplexen IoT-Setups entweder nicht möglich, da zu aufwendig, oder mit immensen Kosten verbunden. Wir raten hier zu einem *Continuous Testing*- beziehungsweise *Testing in Production*-Ansatz auf bestehenden IoT-Infrastrukturen – damit kann neuer Softwarecode bereits eingespielt und über Feature Toggles nur für bestimmte Things, Teilsysteme oder Nutzer freigeschaltet werden, oder ML- und KI-Algorithmen getestet werden. Vergessen wir hierbei nicht, dass auch auf Ebene der einzelnen Things und Enddevices Softwareupdates durch OTAF (Over the Air Flashing) möglich sein müssen, um die komplette IoT-Chain ohne großen Aufwand updaten zu können.
Bis sich Protokoll- und Netzstandards etabliert haben, müssen Sie sich weiterhin Testing-Tools selber programmieren oder zusammenstellen (HiL- bzw. SiL-Simulatoren, Proxy-, Performance-Testing- und Automationstools). Bei den vorwiegend agilen Entwicklungsmethoden empfiehlt es sich, eine *Community of Practice* für den Erfahrungsaustausch zu etablieren, sowie *eigene Tool-Workshop Teams* zu gründen, die sich mit der Programmierung solcher Testtools befassen.
Wie oft genug erwähnt, stellt "*Testing in the wild*" eine optimale zusätzliche Testart dar, die in höheren Teststufen eingesetzt werden sollte, um einen realen Eindruck der Funktionalität im Feld zu gewinnen. Ob dabei Crowdtesting, End-User Testing oder eine "own crowd" gewählt wird, sollte je nach Zielgruppe des Produkts entschieden werden. Hierbei werden Devices nicht in einem sterilen Labor-Test unter perfekten (und damit unrealistischen) Bedingungen getestet, sondern im echten Feldtest. In unserer Erfahrung werden häufig genau dadurch Fehler aufgedeckt, die sonst erst im Produktivbetrieb – und damit den leidtragenden Kunden – aufgefallen wären.

Literatur & Links

[Cog] Whitepaper "The Internet of Things: QA Unleashed", Cognizant, April 2015, siehe:
http://www.cognizant.com/InsightsWhitepapers/the-internet-of-things-qa-unleashed-codex1233.pdf

[Gar] Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016, Gartner.com, 7.2.2017, siehe:
https://www.gartner.com/newsroom/id/3598917

[Gua] Chinese webcam maker recalls devices, The Guardian, 24.10.2016, siehe:
https://www.theguardian.com/technology/2016/oct/24/chinese-webcam-maker-recalls-devices-cyberattack-ddos-internet-of-things-xiongmai

[Hei] IP-Kameras von Aldi als Sicherheits-GAU, Heise.de, 15.1.2016, siehe:
https://www.heise.de/security/meldung/IP-Kameras-von-Aldi-als-Sicherheits-GAU-3069735.html

[Hmq] "10,000,000 MQTT Clients - HiveMQ Cluster Benchmark Paper", dc-square GmbH, 2017, siehe:
http://www.hivemq.com/download.php?token=70e19cd8535b9a9b3745b5f97c04ea09

[Net] CloudPets: IoT-Spielzeuge fordern Lösegeld, Netzpolitik.org, 28.2.2017, siehe:
https://netzpolitik.org/2017/cloudpets-iot-spielzeuge-fordern-loesegeld/

[Ope] Open Device Labs, Liste öffentlich verfügbarer physikalischer Mobile Device Labs, siehe: https://opendevicelab.com/

[Owa] OWASP.org, Internet of Things Project, siehe: https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

[Pos] Postscapes.com, IoT Standards and Protocols, siehe: https://www.postscapes.com/internet-of-things-protocols/

[Wiki] Wikipedia, Dekathlon, siehe: https://de.wikipedia.org/wiki/Zehnkampf

[Wqr] World Quality Report 2017-18, publiziert durch Capgemini, Sogeti und Micro Focus, siehe:
https://www.sogeti.com/globalassets/global/downloads/testing/wqr-2017-2018/wqr_2017_v9_secure.pdf

Ron Werner

ist Team Lead Mobile Testing bei QualityMinds und Senior Test Consultant mit ausgeprägter Erfahrung in Web, Mobile, Automation und Crowdtesting. Ron hält bei internationalen Testing-Konferenzen immer wieder Vorträge zum Thema Mobile Testing und Test Automation und hat dazu auch eine Podcast-Serie gestartet.

E-Mail: ron.werner(at)qualityminds.de

Bastian Baumgartner

ist Team Lead Testing Essentials und Testmanager bei QualityMinds mit mehrjähriger Erfahrung im IoT- und Mobile-Testing-Umfeld.

E-Mail: bastian.baumgartner(at)qualityminds.de

Bildnachweise:

QualityMinds GmbH, dc-square GmbH, rfcx.org