– Verlagsbeilage –

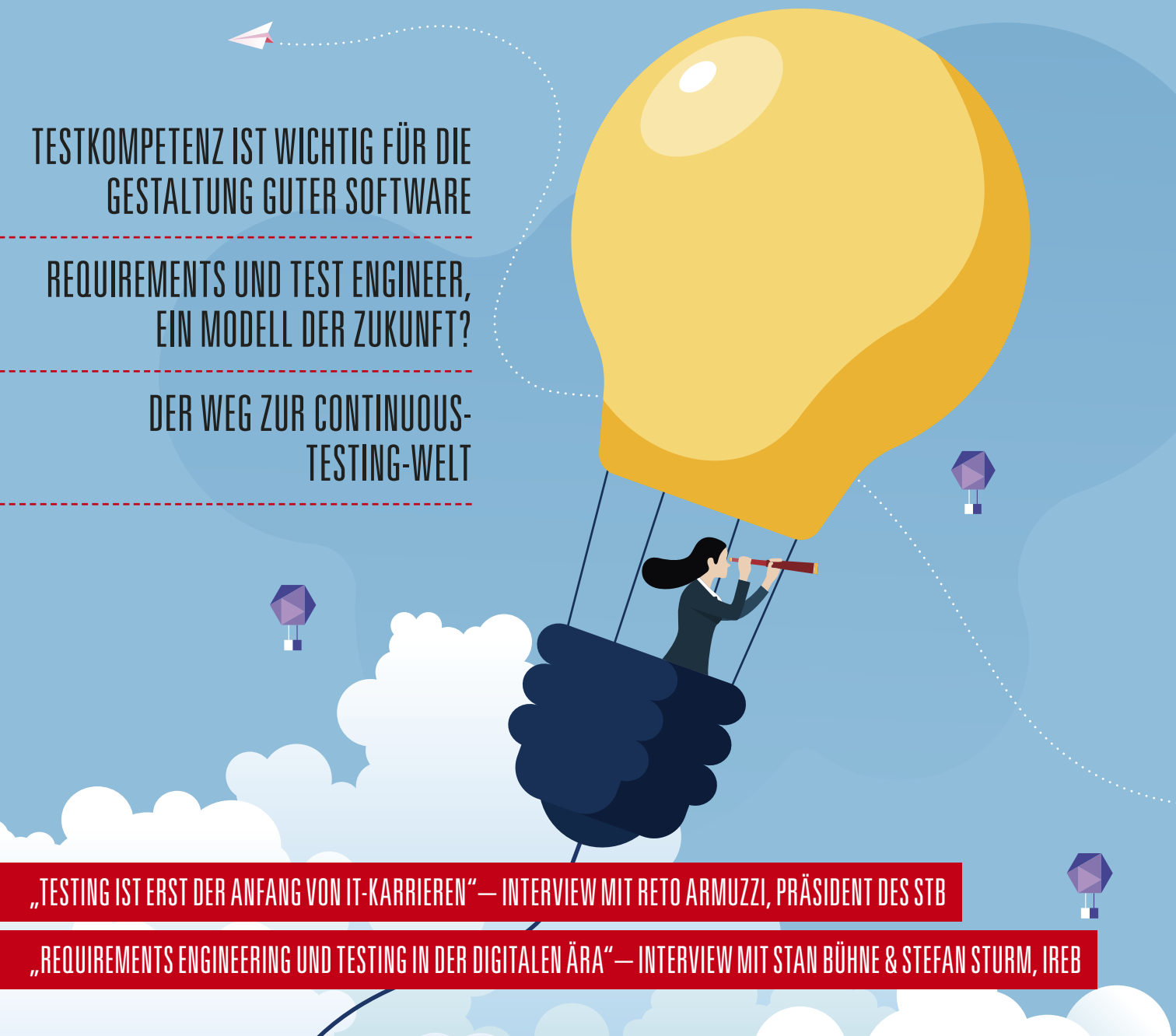
Powered by D·A·CH TESTING BOARDS

GERMAN TESTING MAGAZIN

Das unabhängige Magazin zu Software-Qualität

FUTURE TESTING

WIE SIEHT DIE ZUKUNFT DES TESTENS AUS UND WIE GELANGEN WIR DAHIN?



TESTKOMPETENZ IST WICHTIG FÜR DIE GESTALTUNG GUTER SOFTWARE

REQUIREMENTS UND TEST ENGINEER, EIN MODELL DER ZUKUNFT?

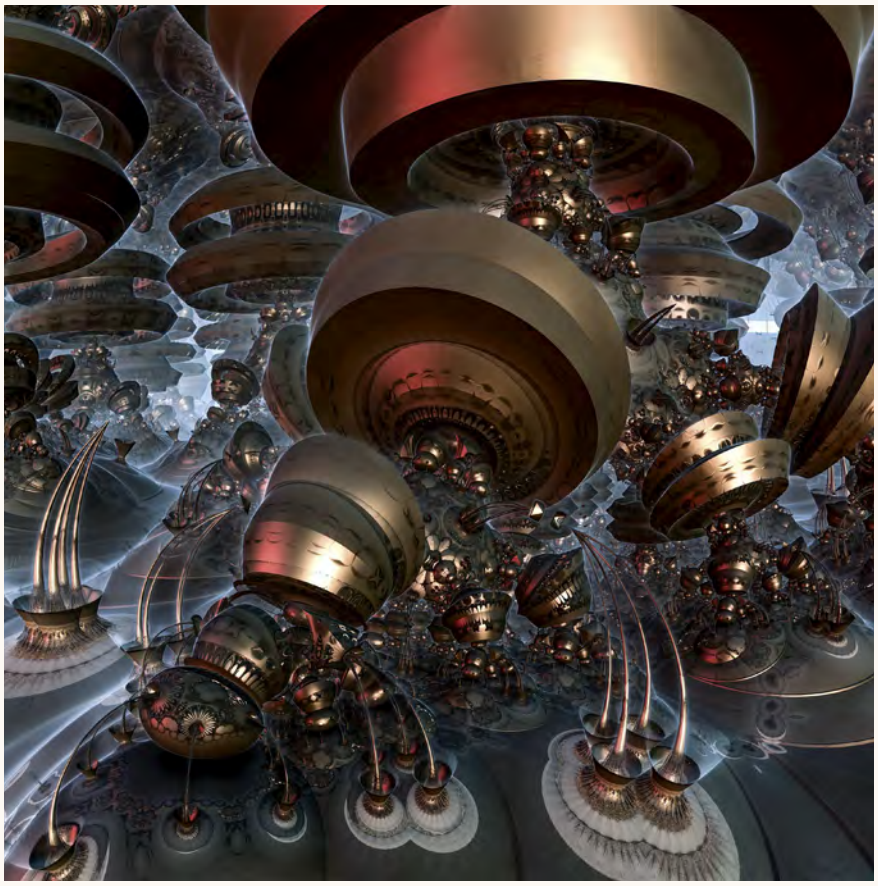
DER WEG ZUR CONTINUOUS-TESTING-WELT

„TESTING IST ERST DER ANFANG VON IT-KARRIEREN" – INTERVIEW MIT RETO ARMUZZI, PRÄSIDENT DES STB

„REQUIREMENTS ENGINEERING UND TESTING IN DER DIGITALEN ÄRA" – INTERVIEW MIT STAN BÜHNE & STEFAN STURM, IREB

»Von Bits und Qubits«

Quantencomputer gelten als die Zukunft schlechthin. Sie sind um ein Vielfaches schneller als herkömmliche Computer und können große Datenmengen in wenigen Sekunden verarbeiten, Verschlüsselungen knacken oder komplexe Aufgaben berechnen. Was ändert sich im Softwaretest von Quantenprogrammen gegenüber dem Test klassischer Software, oder reichen die bekannten Testverfahren noch immer?



Herkömmliche IT-Systeme (PC, Smartphones) steuern Informationen in Form von Nullen und Einsen über *Bits* beziehungsweise *Bytes* an. Die Zahl *42* wird zum Beispiel durch die Bit-Reihe *101010* dargestellt und der Buchstabe *S*, per ASCII-Konvention, durch die Bit-Reihe *01010011*. Der Computer merkt sich, ob eine Bit-Reihe eine Zahl, einen Buchstaben, ein Bild oder Musik darstellt.

Die Bits existieren in Form von Milliarden elektrischer An-/Ausschalter, den Transistoren. Ein Computerprogramm manipuliert diese Bits in einzelnen Rechenschritten über die elementarsten Bausteine der menschlichen Logik: UND-, ODER- und NICHT-Verknüpfungen. Diese Verknüpfungen werden *Gatter* genannt. Eine NICHT-Verknüpfung zum Beispiel kehrt ein einzelnes Bit in sein Gegenteil um: NICHT 0 = 1, NICHT 1 = 0.

Quantenmechanische Eigenschaften von Quantenbits

Die Informationsträger in Quantencomputern bezeichnet man dagegen als *Quantenbits* oder *Qubits*. Im Gegensatz zu Bits befinden sich Qubits während der Berechnung gleichzeitig sowohl im Zustand *0* als auch *1* – jeweils mit einer bestimmten Wahrscheinlichkeit. Erst am Ende des Berechnungsvorgangs befindet sich ein Qubit sicher in einem der beiden Zustände.

Quantenbits unterliegen im Gegensatz zu klassischen Bits den physikalischen Gesetzen der Quantenmechanik und bringen einige Eigenschaften mit, die klassische Bits nicht besitzen. Die wichtigsten dieser Eigenschaften sind zum einen die Superposition und zum anderen die Verschränkung:

- Während des Berechnungsvorgangs können Qubits in eine sogenannte *Superposition* überführt werden. Für ein konkretes Qubit, welches sich in der Superposition befindet, lässt nicht mit Sicherheit sagen, ob es nun den Wert *0* oder *1* besitzt, sondern nur noch, dass es mit einer bestimmten Wahrscheinlichkeit am Ende des Berechnungsvorgangs einen der beiden Werte haben wird.

- Durch verschiedene Operationen des Quantencomputers können Qubits *verschränkt* werden. Sind mehrere Qubits miteinander verschränkt, kann der Zustand eines Qubits nicht mehr unabhängig von den Zuständen der anderen Qubits des Systems beschrieben werden.

Das Ziel beim Quantencomputing ist es, Algorithmen (Handlungsvorschriften zur Lösung eines Problems) zu entwickeln, die das jeweilige Problem unter Ausnutzung der Superposition sowie der Verschränkung lösen. Durch das Benutzen dieser beiden quantenmechanischen Eigenschaften kann sich, je nach Problem und Algorithmus, ein potenziell großer Geschwindigkeitsvorteil ergeben.

Quantencomputer programmieren

Ein Quantencomputer kann analog zu einem mathematischen Modell, das eine abstrakte Maschine definiert (Turingmaschine), programmiert werden. Es werden spezielle Algorithmen benötigt, damit die besonderen Eigenschaften, wie Superposition und Verschränkung, nicht ungenutzt bleiben.

Es gibt Open-Source-Frameworks für die Programmierung von Quantencomputern. Standard Application Programming Interfaces (API) mit Client-Bibliotheken der Programmiersprachen C/C++, Python, MATLAB und einige mehr sind verfügbar. Von IBM gibt es eine grafische Benutzeroberfläche, über die Interessierte mit den Quantencomputern experimentieren können ([IBM], **Abbildung 1**).

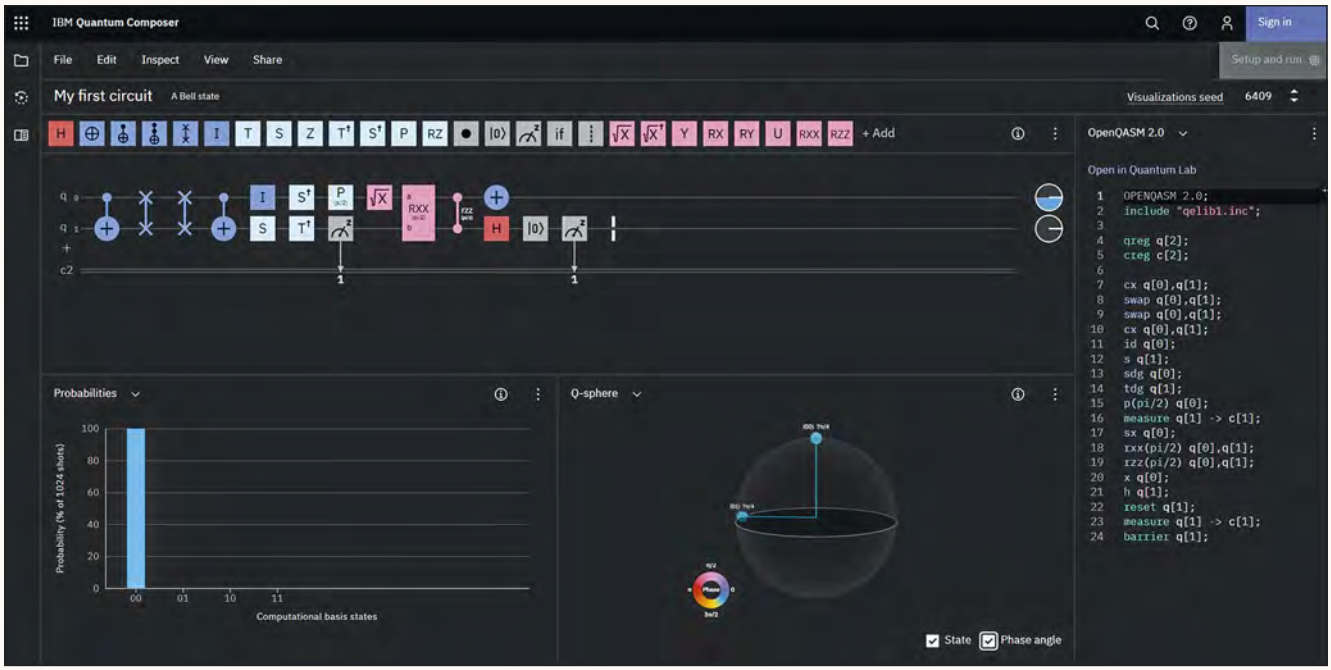


Abb. 1: IBM Quantum Composer

Je nach Architektur der Quantencomputer werden unterschiedliche Programme benötigt. Der Compiler muss die Befehle im Programm in physikalische Manipulation der Qubits umsetzen – und die Ausführung der Befehle unterscheidet sich für Photonen, Atome oder Supraleiter deutlich. Welche Quantenalgorithmen schließlich genutzt werden, hängt davon ab, welche Architektur sich durchsetzen wird.

Theoretisch könnte man auf einem Quantencomputer die klassischen Algorithmen implementieren. Auf einem Quantencomputer lassen sich alle Berechnungen durchführen, die auf einem klassischen Computer möglich sind. Allerdings wären diese Berechnungen aufgrund der Eigenschaften der quanten-physikalischen Systeme, mit denen Quantencomputer realisiert werden, äußerst ineffizient und die Vorteile des Quantencomputers blieben ungenutzt. Daher geht es darum, spezielle Quantenalgorithmen zu entwickeln, welche die Eigenschaften der Quantencomputer effektiv nutzen. Mit diesen Quantenalgorithmen ist es möglich, komplexe mathematische Anforderungen, etwa multiple Optimierungsprobleme, in deutlich kürzerer Zeit zu lösen.

Test von Quantenprogrammen

Wie bei herkömmlichen Computerprogrammen muss auch in der Programmierung von Quantencomputern überprüft werden können, ob sich die Quantenprogramme wie vorgesehen verhalten. Zudem muss es möglich sein, abweichendes Verhalten zu analysieren. Im Gegensatz zur herkömmlichen Programmierung ist es nicht einfach, den Zustand eines Quantensystems zu beobachten und das Verhalten eines Quantenprogramms nachzuverfolgen.

Eines der Merkmale der mit Quantencomputern durchgeführten Berechnungen ist die Unbestimmtheit der Ergebnisse, die nicht immer gleich sind: Ein und dasselbe Programm kann bei zweimaliger Ausführung unterschiedliche Ergebnisse mit unterschiedlichen Wahrscheinlichkeiten liefern. Das Programm muss also viele Male ausgeführt werden, um für das Problem einen Durchschnitt der Ergebnisse zu ermitteln, anstatt einem „ordentlichen", logischen Pfad zu folgen. Diese Unsicherheit führt zu interessanten Herausforderungen beim Testen von Quantenprogrammen.

Zum einen gibt es einen sehr großen, komplexer Quantenzustand. Der Zustand eines Quantensystems, das aus n Qubits besteht, wird durch einen Zustandsvektor der Größe 2^n oder eine Dichtematrix der Größe 2^n x 2^n dargestellt, da ein Quantenzustand in einer Superposition von höchstens 2^n klassischen Zuständen für n Qubits sein kann. Zum anderen muss das Quantensystem gemessen werden, um Informationen über den geprüften Zustand zu erhalten. Der Messvorgang selbst beeinflusst jedoch den Zustand der Qubits. Die Messung eines Quantenzustands, der eine Überlagerung vieler klassischer Zustände ist, liefert nur einen dieser Zustände und lässt die Überlagerung auf diesen Zustand kollabieren. Einzelne Messungen charakterisieren also nicht die vollständige Überlagerung, sondern zerstören sie sogar.

Anders verhält es sich, wenn das Quantenprogramm auf einem Simulator ausgeführt wird, der deterministische Ergebnisse liefert. Die Komplexität eines Quantenprogramms kann das Testen auf einem Simulator zu einer unlösbaren Aufgabe machen.

In einem klassischen Softwaretestprojekt ist ein Testfall eine Spezifikation der Eingaben, Ausführungsbedingungen und der erwarteten Ergebnisse. Diese Spezifikation definiert einen einzelnen Test, der auszuführen ist, um ein bestimmtes Testziel zu erreichen. Eine der gewünschten Eigenschaften von Testfällen ist ihre Wiederholbarkeit: Das heißt, das Ergebnis eines Tests *muss immer das Gleiche sein*.

Um einen Testfall auf einem Quantencomputer qualifiziert auszuführen, muss derselbe Testfall mehrmals ausgeführt werden, um das Ergebnis, das am häufigsten auftrat, mit dem erwarteten Ergebnis zu vergleichen.

Testverfahren

Im klassischen Softwaretest unterscheiden wir gemäß International Software Testing Qualifications Board (ISTQB®) zwischen Blackbox- und Whitebox-Testverfahren [GTB].

Im Whitebox-Testverfahren wird die interne Struktur der Maschine, des Algorithmus oder Geräts, allgemein als System under Test (SUT) bezeichnet. Im Whitebox-Testverfahren haben wir Zugang zum Code des Softwaresystems. Quantensysteme bringen in diesem Testverfahren viele Einschränkungen mit sich. Selbst wenn wir die in der Maschine stattfindenden Transformationen kennen, können wir das Ergebnis nicht überprüfen, da jede Messung den Zustand des Systems unwiderruflich zerstört.

Blackbox-Testverfahren, klassisch oder im Rahmen von Quantensoftware genutzt, erlauben es dem Tester nicht, in das Innere des SUT zu blicken. Die Interaktion mit der Maschine erfolgt über deren Ein- und Ausgänge.

Formal unterscheiden wir bei den Testverfahren für Quantensoftware also auch zwischen Black- und Whitebox-Szenarien, indem wir berücksichtigen, ob die am System vorgenommenen Transformationen und Messungen bekannt sind, auch wenn der Zustand der Transformation selbst unbekannt ist.

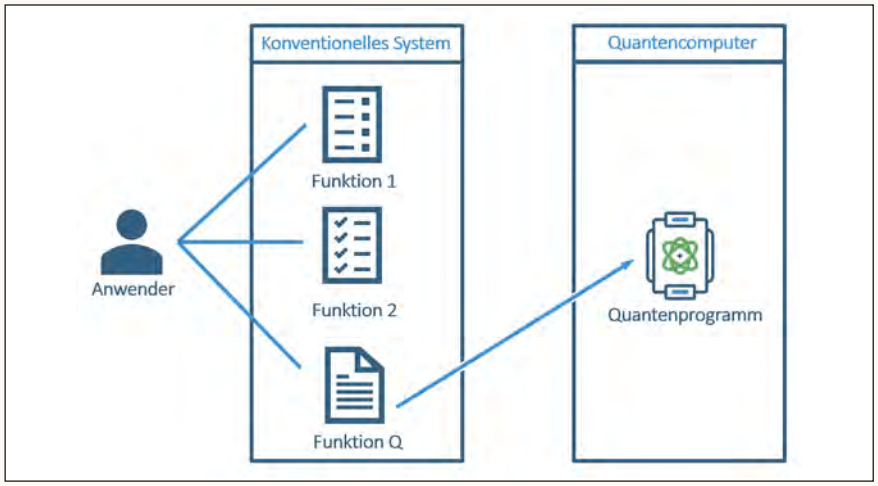


Abb. 2: Der Quantencomputer bietet Dienste für ein konventionelles IT-System

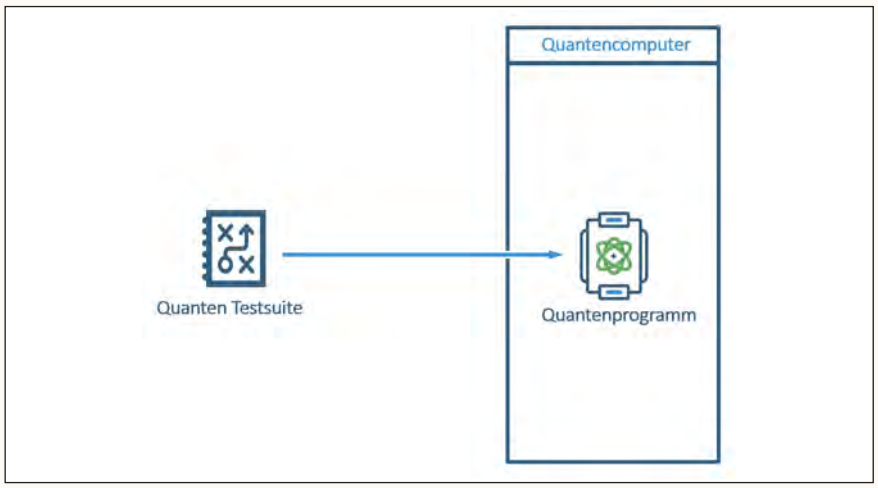


Abb. 3: Funktionale Testsuite ersetzt den Anwender

Referenzen

- [GTB] https://www.german-testing-board.info/lehrplaene/istqbr-certified-tester-schema/glossar/
- [IBM] https://quantum-computing.ibm.com/composer/files/new
- [MatL] https://de.mathworks.com/products/matlab.html
- [Pyth] https://www.python.org/

Funktionstest

Aktuell stellen in der Praxis Quantenprogramme konventionellen Programmen Quantenfunktionalitäten über eine Art Schnittstelle zur Verfügung. Das konventionelle Programm ruft das Quantenprogramm auf, um eine bestimmte Berechnung durchzuführen **(siehe Abbildung 2)**.

Beim Funktionstest werden durch die Menge der Testfälle die verschiedenen Funktionalitäten, die das SUT bietet, getestet. Typischerweise werden durch die Testsuite die gleichen Operationen auf dem System ausgeführt, die ein Anwender ausführen könnte.

Eine funktionale Testsuite könnte den Anwender ersetzen. Auf die gleiche Weise könnte eine Quanten-Testsuite – zum Testen des eigentlichen Quantenprogramms – die konventionellen Systemfunktionalitäten ersetzen **(siehe Abbildung 3)**. Um den erwähnten Nicht-Determinismus des Testfalls zu berücksichtigen, muss dieser mehrmals ausgeführt werden.

Um die dreiteilige Struktur (Eingabespezifikation, Ausführungsbedingungen, erwartete Ergebnisse) eines Testfalls zu gewährleisten, ist der oben beschriebenen Unsicherheit mit der mehrmaligen Ausführung jedes Testfalls zu begegnen:

- Die Ausgangssituation eines Quanten-Testfalls bestimmt den Anfangszustand der Qubits.
- Wie bei einem herkömmlichen Softwaretest wird die Quantenschaltung ausgeführt.
- Danach speichert die Testsuite das erzielte Ergebnis, um in einem letzten Schritt zu berechnen, welches das wahrscheinlichste tatsächliche Ergebnis ist.

Fazit

Nach heutigem Stand – wobei das Thema noch in den Kinderschuhen steckt – lässt sich sagen, dass sich beim Softwaretest von Quantenprogrammen gegenüber klassischer Software nichts ändern wird.

Auch im Software-Testing von Quantenprogrammen wird grundsätzlich zwischen Testverfahren und Testarten unterschieden. *Testverfahren* geben an, wie getestet wird. Die bekannten Testverfahren wie Black- und Whitebox-Testverfahren werden auch in der operativen Testdurchführung von Quantenprogrammen Anwendung finden.

Die grundlegenden *Testarten* wie funktionale Tests und nicht-funktionale Tests und Anforderungs- und strukturbasierte Tests werden je nach Ziel der Softwaretests ausgewählt und angewendet. Dabei subsumiert die Testart funktionale Tests alle Testverfahren beziehungsweise Testmethoden, mittels derer das von außen sichtbare Ein-/Ausgabeverhalten eines Testobjekts geprüft wird.

Auch bei Quantenprogrammen müssen die Testergebnisse eineindeutig sein, um sie als korrekt (im Sinne der Anforderung) zu bewerten. Daher gilt wohl das Ergebnis als wahr, welches in den Testläufen eines Quantenprogramms am häufigsten errechnet wurde und vom Entwickler als korrekt eingestuft wird.

Sören Schmock

soeren.schmock@itgain.de

ist seit über 15 Jahren in Projekten im IT-Umfeld tätig. Dabei hat er überwiegend im Qualitätsmanagement gearbeitet und deckt alle Fragestellungen zu diesem Thema ab. In seinen Projekten war er für die Prozessoptimierung, Anforderungserhebung sowie die Planung und Durchführung von Test- und Qualitätssicherungsmaßnahmen mehrerer Anwendungskomponenten für bankfachliche, kommunikationstechnische sowie versicherungsspezifische Anwendungsverfahren verantwortlich. Darüber hinaus zählen Standardisierung sowie Trainings zu seinen Aufgaben. In der ITGAIN verantwortet Sören Schmock den Bereich Financial Solutions. In dieser Rolle ist er für die thematische Weiterentwicklung des Bereichs sowie für die Mitarbeiter verantwortlich.