We Are Developers!

Eine Themenbeilage der
Heise Medien GmbH & Co. KG

_KARRIERE

Warum Entwickler auch Business-Developer sein müssen

_BLOCKCHAIN

Smart Contracts in Rust

_MACHINE LEARNING

ML-Modelle mit DVC, Cortex und ONNX deployen

_WORLD WIDE WEB

Die beste Zeit, um Webentwickler zu werden

_CODEREVIEWS

Codequalität lehren und lernen

_ZUKUNFSTECHNIK

Das Potenzial von Quantum Computing

Powered by heise Developer


#IWILLNOTSTOP
UNTIL ALGORITHMS
CONTAIN A CODE
OF ETHICS.
Francisco arbeitet an der Zukunft von AI und Machine Learning. Unsere Leitlinien zur digitalen Ethik bilden dafür den Rahmen. Denn wir sind davon überzeugt, dass Fortschritt und Verantwortung zusammengehören. Unsere Experten setzen sich dafür ein, dass künstliche Intelligenz unser Leben positiv bereichert und Technologie für alle fair und transparent wird. Erfahren Sie mehr über die Arbeit unserer Experten unter telekom.com/iwillnotstop
ERLEBEN, WAS VERBINDET.

EDITORIAL

Eine Zeit, in der es mittlerweile allen bewusst ist, dass ohne Software eigentlich nichts mehr geht, sollte es eine Selbstverständlichkeit sein, dass Softwareentwicklern beziehungsweise -entwicklerinnen eine immens wichtige Rolle zukommt. Ohne sie geht quasi nichts mehr. Umso wichtiger ist es, dass eine Gesellschaft alles dafür tun muss, die Rahmenbedingungen dafür zu schaffen, dass der Strom neuer Developer nicht abreißt, um die vielfältigen Herausforderungen einer zunehmend mehr vernetzten digitalisierten Welt zu meistern. Schon angefangen bei den Schulen, dann im Studium, durch Förderprogramme, aber auch in Unternehmen, in denen die jungen Talente schließlich aufschlagen – allerorten sind wir in der Pflicht, ein attraktives Lernumfeld für eine gute oder gar bessere Softwareentwicklung zu schaffen.

Die Heise Medien, bekannt durch Marken wie c't, iX, heise online und einige weitere mehr, gehören ebenfalls zu denen, die in der Pflicht stehen, junge Talente in der Softwareentwicklung, ja, in der IT insgesamt, zu fördern. Hier kommt nun diese Beilage „We Are Developers!" ins Spiel, die in Zusammenarbeit mit unseren Freunden von der IT-Job-Plattform WeAreDevelopers entstanden ist. Wir haben nach jungen aufstrebenden Developern für Artikel zu wichtigen, vorrangig aber spannenden neuen Themen gesucht – ganz bewusst mit dem Ziel, ihnen eine Kommunikationsplattform und ihren Altersgenossen einen faszinierenden Einblick in die Welt der Softwareentwicklung zu bieten, und zwar aus unterschiedlichen Perspektiven wie Karriere, Erfahrungen und aufregenden Technologien.

Viel Spaß bei der Lektüre!

Alexander Neumann

INHALT

Young Professionals schreiben für Young Professionals

Du willst ebenfalls dein Talent als schreibender Developer unter Beweis stellen? Dann bist du bei uns richtig. Lerne in Zusammenarbeit mit erfahrenen Redakteuren das Konzipieren, Verfassen und Veröffentlichen von Fachartikeln und präsentiere dich in den Heise Medien als junges Entwickler-Talent.

Kontakt: developer@heise.de

>Kreativität gefragt

Antti Pitkänen

Code zu „schrubben“ ist eine Sache, zusammen mit Kunden kreative Lösungen für deren Anforderungen zu finden, eine ganz andere.

Code zu schreiben oder Softwaretests und Entwicklungswerkzeuge zu konzipieren – diese Aufgaben haben lange Zeit das Beschäftigungsfeld der Entwickler maßgeblich bestimmt. Doch im Zuge der Digitalisierung hat sich dieses Spektrum deutlich erweitert – und ist dabei spannender geworden. Denn heute ist ihr kreativer Beitrag gefragt, um Geschäftsmodelle von „analog“ auf „digital“ umzustellen. Das gilt nicht allein für die Innovations- und Digitalberatungen, sondern auch für die „klassischen“ Softwareentwickler.

Man sollte sich von dem Gedanken verabschieden, dass Entwickler oft nur das umsetzen, was Kollegen aus dem Beratungsteam zusammen mit Kunden vereinbart haben – etwa Web-Anwendungen zu erstellen oder Legacy-Applikationen per „Lift and Shift“ in eine Cloud-Umgebung zu überführen. Gefordert sind hingegen zunehmend Fachleute, die die Rolle von kreativen Problemlösern übernehmen. Das heißt, sie müssen sich vermehrt in die speziellen Anforderungen der Kunden hineindenken. Das gilt nicht nur für technische Aspekte, sondern auch für wirtschaftliche und strategische Erwägungen.

_Sanft beginnen

Doch nach diesem Ausflug in strategische Höhen zunächst einmal zurück auf den Boden der Realität. Begonnen sei mit dem Berufseinstieg von Entwicklern oder dem in den Arbeitsalltag. Empfehlenswert ist, die neuen Kollegen zu Beginn mit Aufgaben zu betrauen, die weniger „kritisch“ sind. Das kann zum Beispiel das Abarbeiten von Tickets bei Produkten sein, die bereits bei Kunden im Einsatz sind. Häufig geht es dabei darum, Features hinzuzufügen oder Bugs zu beseitigen. Ein solch sanfter Einstieg ist vor allem für Mitarbeiter hilfreich, die erst am Anfang ihrer Karriere stehen, sprich „frisch von der Schulbank“ kommen. Selbiges mag aber auch für Quereinsteiger gelten. Durch diese Vorgehensweise hat das neue Teammitglied die Chance, sich ohne allzu großen Druck mit den Workflows und eingesetzten Tools vertraut zu machen.

Wichtig ist zudem, dass die bestehenden Beschäftigten den Neueinsteigern das Gefühl geben, dazu zu gehören, und dass die neue Meinung im Team ebenso zählt. Nicht förderlich ist dagegen, wenn den neuen Kollegen gleich zu Beginn klargemacht wird, dass ihre Python- oder JavaScript-Kenntnisse unterirdisch seien. Es hängt maßgeblich von der sozialen Kompetenz und den „Soft Skills“ der Kollegen und der Teamleads ab, ob sich eine produktive und vertrauensvolle Zusammenarbeit ergibt. Das bestätigt auch eine Studie des Personaldienstleisters Manpowergroup (www.manpowergroup.de/neuigkeiten/studien-und-research/skills-revolution-ii/) von 2018. Demnach suchten 78 Prozent der Unternehmen Mitarbeiter, die gut kommunizieren können, an die 86 Prozent wollten Beschäftigte mit ausgeprägten Team-Fähigkeiten. Das galt auch für technische Sparten wie die Softwareentwicklung und das Programmieren.

_Soft Skills machen den Unterschied

Allerdings hat laut der Studie etwa ein Drittel der Unternehmen Probleme damit, Mitarbeiter mit solchen Soft Skills zu

finden. Daher hängt es in hohem Maß von der Unternehmenskultur und der Vorbildfunktion der Belegschaft ab, inwieweit die Mitglieder von Entwicklungsteams diese Fähigkeiten erwerben. Hilfestellung geben dabei Unternehmensgrundsätze, die mittlerweile viele Technologieunternehmen definiert haben. Doch was in einem Mitarbeiterhandbuch steht und wie Unternehmenskultur in der Praxis gelebt wird, sind leider in vielen Unternehmen oft grundverschiedene Dinge. Dennoch lohnt sich ein Blick auf die Versprechen, die Unternehmen ihren Mitarbeitern machen. Wichtig ist beispielsweise, dass sie nicht ausschließlich Kundenbedürfnisse in den Mittelpunkt stellen.

Eine zentrale Rolle spielt zudem die Weiterentwicklung der Fähigkeiten der eigenen Mitarbeiter, sowohl auf der fachlichen Ebene als auch im Bereich der Soft Skills. Hierfür sollte Arbeitgeber regelmäßig Möglichkeiten einräumen. Diese sollten auch nicht einfach nur vorgesetzt werden – nachhaltiger wirken die Programme, wenn diese Fortbildungen kooperativ erarbeitet werden, also dass das Unternehmen auf die Vorstellungen der Mitarbeiter in puncto Weiterbildung und Entwicklungswünsche eingeht.

Zur Kultur der Arbeitgeber zählt auch ganz wesentlich die interne Kommunikation. Hier hat sich in den letzten Jahren ein Wandel bemerkbar gemacht. Hierarchische Führungskulturen nach dem Motto „Ich bin der Chef und weise hiermit an“ treten mehr und mehr hinter kooperativen Ansätzen zurück, die den Mitarbeitern zunehmend Freiheit und Eigengestaltung zugestehen. Ob diese Ansätze im Alltag auch gelebt werden, zeigt sich dann allerdings erst in der Praxis. Wie auch immer hier die Erfahrungen sind, sicher ist, dass sich hier das Klima geändert hat und viele junge Arbeitnehmer die kooperativen Kulturen bevorzugen. Allerdings sollte man sich dabei vor Augen halten, dass mit diesem Mehr an Freiheit auch ein Mehr an Verantwortung verbunden ist.

Arbeitgeberkultur im Wandel

_Mit den Aufgaben wachsen

Nach der Aufwärm- und Eingewöhnungsphase folgt für Entwickler typischerweise ein erster Schritt in Richtung Kundenberatung. Ein Beispiel: Ein Unternehmen will zwei Services zusammenfassen, die bei der Kundenbetreuung eingesetzt werden, nämlich das Übermitteln von Kontaktformularen via E-Mail sowie das Buchen eines Telefontermins bei einem oder einer Support-Mitarbeiter/in. Das Problem dabei: Beide Services haben aus technischer Sicht nichts miteinander zu tun; sie zu kombinieren – so der Ausgangsplan des Unternehmens – hätte neue Backend-Services erfordert. In einem solchen Fall gilt es, auch beratend tätig zu werden:

- Es ist zu prüfen, welche Relevanz die gewünschten Änderungen für den Geschäftsbetrieb der Anwender haben. Das kann auch die Entwicklung von Szenarien umfassen, mit denen das Unternehmen seinen Kunden ergänzende Services anbieten und somit Mehrwert generieren kann.

- Es gilt zu ermitteln, welche technischen Alternativen es gibt, um diese Anforderungen umzusetzen.
- Außerdem sollten die technischen und finanziellen Aufwendungen, die damit verbunden sind, geprüft und im letzten Schritt mit den Kunden diskutiert werden, welches Szenario für sie akzeptabel ist.

_Business-Development inklusive

Dieses Aufgabenportfolio geht weit über die Definition des Entwicklerberufs im landläufigen Sinn hinaus. Die Fachkraft ist nicht nur damit beschäftigt, Lösungen umzusetzen. Die Tätigkeit setzt viel früher an, und zwar mit Beratungsleistungen: Prognosen zu Änderungen, Entwicklung möglicher Szenarien, Aufzeigen von Alternativen und Kostenbewertungen – das sind Tätigkeiten, die eher im Bereich Business-Development verortet sind als in der reinen Softwareentwicklung. Die damit aufgerufene Beratungskompetenz ist dabei nicht nur etwas, was Digital- oder Innovationsfirmen leisten, auch interne Entwickler sollten für diese Art des Mit- und Weiterdenkens erhöhte Wertschätzung erfahren. Auch das ist Teil der oben erwähnten Verantwortung in kooperativen Firmenkulturen.

Zurück zum Beispiel: In diesem Fall war die Lösung relativ einfach. Man entschied sich, den Kunden ein Minimalszenario vorzuschlagen, das sich schnell und kostensparend umsetzen ließ. Statt die beiden Services aufwendig zu kombinieren, wurde eine ID in die Kontaktformulare eingefügt. Diese Kennung nutzt auch der Dienst, der für den Telefon-Support zuständig ist. Die Arbeitskräfte können dadurch schneller und effektiver auf Anfragen eingehen.

_Wachsen mit den Anforderungen

Bedeutet das nun, dass Teammitglieder der Entwicklung als Business-Development-Manager eingesetzt werden und diese so den Bezug zum Programmieren, zur technischen Seite, verlieren? Das Gegenteil ist der Fall. Vor allem Beratungsfirmen können es sich nur selten leisten, Entwickler in einem eng umgrenzten Umfeld einzusetzen. Stattdessen ist Vielseitigkeit gefragt. Ein Teammitglied der Entwicklungsabteilung setzt beispielsweise im Rahmen eines Projekts Nginx- oder Linux-Webserver auf und verwaltet sie über SSH. In einem anderen Fall entwickelt man Algorithmen, welche die Ausfallwahrscheinlichkeit von Komponenten in Stromversorgungseinrichtungen oder Fertigungsumgebungen vorhersagen. Auch der Umgang mit Container-Techniken und DevOps kann ebenso zum Alltag gehören wie spezielle Projekte. Ein Beispiel: Um Legacy-Anwendungen über DevOps-Verfahren debuggen zu können, kann es nötig sein, die Applikation mit dem JavaScript-Framework Redux zu „verheiraten“. Ein Mangel an technisch orientierten Aufgaben besteht also nicht.

Vielgestaltiges Aufgabengebiet: Ein Entwickler darf bei Bedarf auch mal in ein Drachenkostüm schlüpfen, etwa im Rahmen einer Marketingaktion.

_Technologie ist wichtig, aber nicht alles

Apropos Legacy-Anwendungen: Zu den anspruchsvollsten Aufgaben zählt, die Code-Basis der Services beziehungsweise Applikationen auf einen aktuellen Stand zu bringen. Die meisten Entwickler kennen Probleme wie Spaghetti-Code, lückenhafte oder fehlende Dokumentationen, komplexe technische Abhängigkeiten, bei denen häufig veraltete Komponenten mit im Spiel sind, und fehlende Testmöglichkeiten.

Sicherlich gibt es etliche Optionen, solche Klippen zu umschiffen. In den Diskussionen von Entwicklern fallen dann häufig Begriffe wie React-Hooks, Serverless Computing, Infrastructure as Code und Cloud. Selbstverständlich sollten Entwicklungsprofis damit vertraut sein. Für sie geht es jedoch primär darum, Anwendern die Möglichkeit zu geben, der Kundschaft zu einem Mehrwert zu verhelfen. Die Arbeit besteht in diesem Fall darin, Services zu entwickeln, welche die Wertschöpfung von Angeboten und Produkten verbessern. Der Code und andere technische Details spielen dabei eine untergeordnete Rolle.

Für klassische Programmierer mag das wie Hochverrat oder blanke Hybris klingen. Aber für ein Teammitglied der Entwicklung ist ein solcher Ansatz nur konsequent. Höchste Priorität ist es, den Anwendern eine praktikable Lösung zu bieten, die sich nicht in technischen Spielereien verliert, sondern neben Technologien auch Faktoren wie Wirtschaftlichkeit, Value-to-Market und Zukunftsorientierung berücksichtigt.

_Praxisbeispiel: Predictive Maintenance in der Energieversorgung

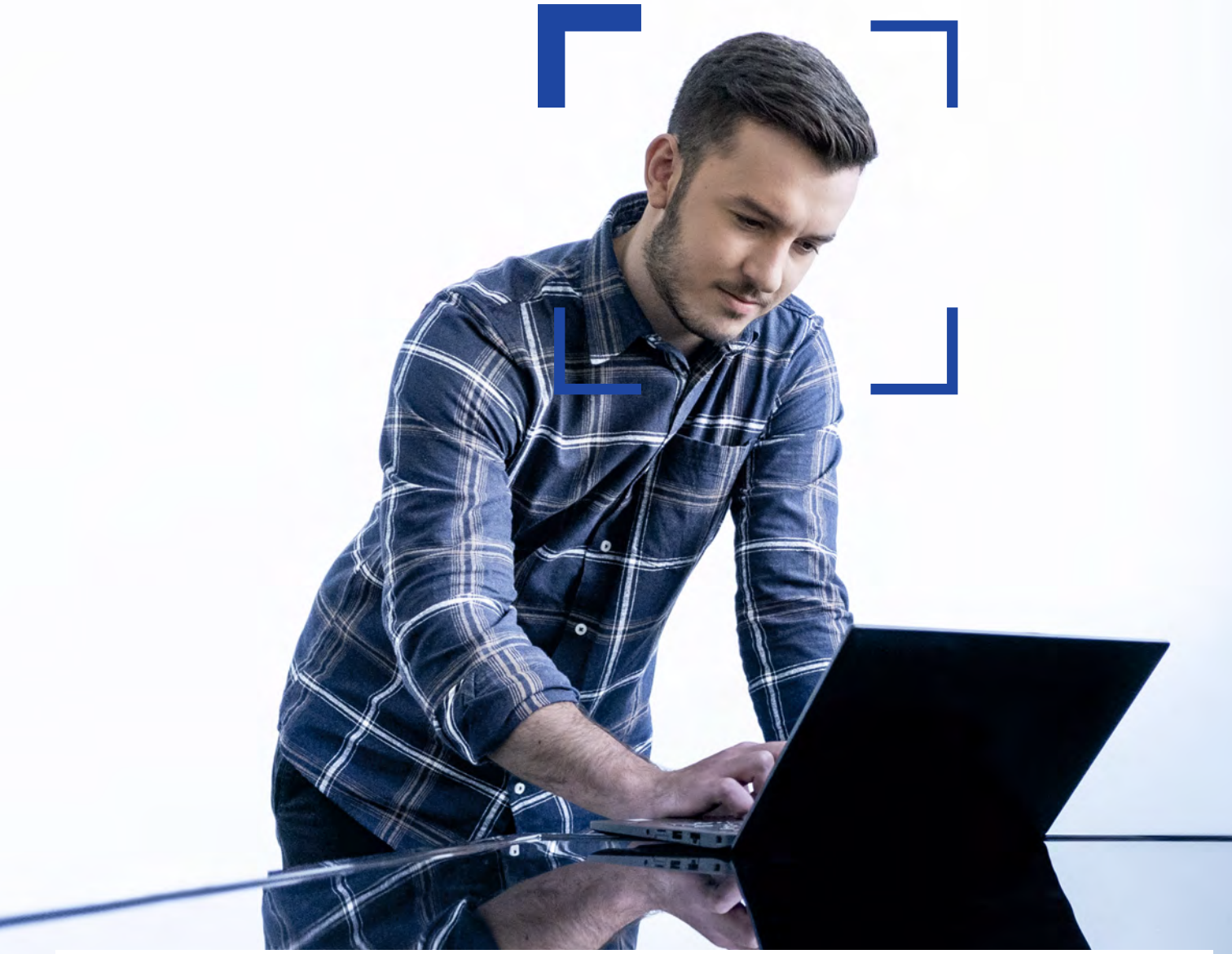
Wie das in der Praxis aussehen kann, zeigt das Beispiel einer Software, die „proaktiv“ Ausfälle von Systemen in einem

Zukunft passiert nicht einfach.

Ich schreibe sie.

ZEISS

Seeing beyond



ZEISS bietet Dir zahlreiche Möglichkeiten, Zukunft und Fortschritt aktiv mitzugestalten. Innovationen bei ZEISS entstehen im Zusammenspiel von präziser Hardware mit leistungsfähiger Software.

Wenn Du Software Deine Handschrift geben möchtest, wenn Du komplexe Architekturen entwickeln möchtest, die einfach mehr bieten, wenn Du den Unterschied machen willst, bewirb Dich jetzt!

#kameradraufhalten
#jobsentdecken
#teamzeiss

Stromnetz erkennt: Stichwort Predictive Maintenance. Der vorhandene Software-Stack und Sourcecode der Services waren in diesem Fall nicht auf dem Stand der Technik. Doch daran ließ sich nichts ändern. Zwei Herausforderungen des Projekts waren:

- Es galt, die Software um eine Predictive-Maintenance-Funktion zu erweitern. Das war wegen des alten Source codes eine anspruchsvolle technische Aufgabe.
- Dabei sollte gleichermaßen das Ziel im Auge behalten werden: Die Betreiber des Stromnetzes wollten die Möglichkeit erhalten, Servicefachleute bereits dann zu Kraftwerken oder Umspannanlagen zu schicken, bevor diese überhaupt ausfielen. Denn die Netzbetreiber können so Ausfallzeiten reduzieren und die Servicequalität steigern.

_Dauerhafte Partnerschaften statt „Code and Forget“

Ein positiver Nebeneffekt dieser Arbeitsform ist, dass Entwickler in den meisten Fällen sehen, wie sich die Arbeit in der Praxis bewährt. Es kann also nicht einfach darum gehen, nur Code zu erstellen und zu testen, dann ab zum Kunden und fertig. Ganz nach dem Motto „Code and Forget“.

Vielmehr muss es darum gehen, dass Entwickler im Idealfall oft mehrere Jahre lang einen Account betreuen. Für Kunden bedeutet das, dass Software und Services aus einem Guss sind. Zudem gibt es weniger Reibungsverluste in der Abstimmung zwischen beiden Parteien, und die Kundschaft profitiert von einer länger andauernden Co-Creation-Partnerschaft. Schließlich haben es die Entwickler in der Hand, eine Applikation „sauber“ zu designen und zu testen. Das reduziert den Wartungsaufwand der Lösung. Außerdem ist es dann einfacher, sie auf Wunsch um neue Funktionen zu erweitern.

_Rotation erhält den Spaß an der Arbeit

Vernünftige Unternehmen handeln sicherlich nicht nach dem Motto „Einmal dieser Kunde, immer dieser Kunde“. Denn dann würde vermutlich ein Großteil der Entwickler spätestens nach 18 Monaten davonlaufen. Die Regel sollte sein, dass ein Teammitglied bei mehreren Accounts mitarbeitet. Außerdem besteht die Möglichkeit, nach einiger Zeit den Schwerpunkt von einem Projekt auf ein anderes zu verlagern. Das ist das normale Prozedere, allerdings sollten Interessierte im Vorfeld nachfragen, wie es das Unternehmen mit der „Job-Rotation“ hält.

Jedoch sollte nicht nur die Möglichkeit bestehen, zwischen den Projekten zu wechseln. Auch eine Bewegung in „vertikaler“ Richtung ist denkbar, vor allem bei Projekten mit einer längeren Laufzeit. So hat der Autor dieses Beitrags mittlerweile die Funktion eines Scrum Master bei einem solchen Vorhaben übernommen. Dadurch hat sich das Aufgabenspektrum deutlich erweitert. Die Agenda umfasst nun sowohl die Kommunikation mit Kunden und die Verteilung von Entwicklungsaufgaben an die Teammitglieder, als auch das Coding auf der Anwendungsebene und Aufgaben aus dem Bereich DevOps.

Selbst das Onboarding neuer Mitarbeiter, die Teilnahme als Sprecher an Fachveranstaltungen sowie Meetings mit Vertrieb und Account-Management stehen auf der Tagesordnung. Das untermauert, dass Entwickler ein weitläufiges und interessantes Betätigungsumfeld vorfinden, allerdings auch eines, das durchaus herausfordernd sein kann und weit mehr Skills erfordert als nur das Programmieren.

_Developer müssen keine Nerds sein

Doch welche Ausbildung oder speziellen technischen Fertigkeiten müssen Entwickler mitbringen? Zumeist kommt es nicht darauf an, perfekten Code erstellen zu können. Sicherlich ist Coden ein Bestandteil des Jobs, aber eben nur einer von vielen. Daher können auch Personen, die quer einsteigen, eine Entwicklerlaufbahn einschlagen – etwa Absolventen einer Ausbildung mit dem Schwerpunkt auf Betriebswirtschaft.

Gemeinsam mit den Kunden arbeiten!

Wichtig ist, dass Bewerber bereit sind, sich mit technischen Fertigkeiten wie Programmieren sowie den Technologien vertraut zu machen, die bei der Digitalisierung von Prozessen und Geschäftsmodellen des Unternehmens eine Rolle spielen. Aber ebenso relevant ist, dass Entwickler mehr erreichen können, als nur die technischen Hausaufgaben umzusetzen, die andere definiert haben. Zu ihren Tätigkeiten gehören, zusammen mit Kunden neue Ideen zu entwickeln und umzusetzen, kreative Lösungen zu finden und das eine oder andere Mal auch eine konstruktive Auseinandersetzung zu führen.

Dabei steht immer ein Gedanke im Mittelpunkt: gemeinsam mit den Kunden Strategien und digitale Ansätze zu erarbeiten, die einen klaren Mehrwert bieten. Langweile kommt bei dieser Interpretation des Entwicklerberufs jedenfalls nicht auf.

Antti Pitkänen

ist 29 Jahre jung und arbeitet als Software- und Business-Developer in der Digitalberatung Futurice. Sein Arbeitsplatz befindet sich in der finnischen Stadt Tampere, einem der acht internationalen Standorte des Unternehmens.

Erleichtern Sie mit uns das Leben von acht Millionen Bürgerinnen und Bürgern Österreichs!

Das Bundesrechenzentrum ist das Kompetenzzentrum für die Digitalisierung des Public Sectors in Österreich. Wir entwickeln smarte und sichere IT-Lösungen und nutzen dabei innovative Technologien wie Big Data, Artificial Intelligence, Blockchain oder Robotics.

Wir sind stolz auf unsere hochqualifizierten und engagierten Kolleginnen und Kollegen und arbeiten gemeinsam daran, Österreich fit für die Zukunft zu machen.

Das BRZ sucht IT-Spezialistinnen und Spezialisten (m/w/d) in unterschiedlichen Tätigkeitsbereichen:

- **> Project Management**
- **> Lead Development**
- **> Software-Testing**
- **> System Management Virtualization**
- **> Application Management Predictive Analytics**
- **> SAP BI-Consultants**
- **> Kreative Köpfe, die Österreichs IT weiterbringen wollen**

Wir freuen uns darauf Sie kennenzulernen!
www.brz-jobs.at

BRZ

>PRO WEBENTWICKLUNG

Christian Liebel

Jungen Informatikern wie dir stehen so viele Optionen offen wie noch nie. In welche Richtung sollte man also starten? Ich selbst bin 2014 aus der Windows-Welt in die Webentwicklung gewechselt und mit der Entscheidung nach wie vor sehr zufrieden. Warum ist jetzt die beste Zeit, Webentwickler zu werden?

Das World Wide Web war ursprünglich als Plattform zum einfachen Austausch von Forschungsergebnissen gestartet. Ganz am Anfang gab es noch nicht einmal einen Weg, in Webseiten Bilder einzubinden. Binnen weniger Jahre wurde das WWW äußerst populär und die Möglichkeiten sind regelrecht explodiert. Weil statische Webseiten nicht ausreichten, wurde mit JavaScript ein Weg zur clientseitigen Dynamisierung geschaffen. Schnell wurden auch Anwendungen auf Basis von Webtechniken entwickelt. Mit HTML5 verwandelte sich das Web schlussendlich zur Multimedia- und Anwendungsplattform. Seitdem sind weitere Schnittstellen im Web gelandet, von denen Entwickler früher nur hätten träumen können. Heute ist das Web nahezu allgegenwärtig und seine Zukunftsaussichten sind blendend.

_Ein bisschen was von allem

Die Webentwicklung ist genauso breit gefächert wie die Anwendungsentwicklung an sich. Es gibt unter anderem folgende Disziplinen (in Klammern eine Auswahl der wohl jeweils wichtigsten Techniken):

- Frontends (Angular, React oder Vue.js)
- Backends (ASP.NET Core, Node.js, PHP)
- Kommunikation zwischen Front- und Backend (HTTPS/REST, WebSockets, gRPC)
- Datenbanken (NoSQL- oder SQL-Datenbanken)
- Authentifizierung (OAuth 2.0 und OpenID Connect)
- Tests (Karma, Jasmine, Selenium)
- Dokumentation (Swagger, JSDoc)
- Design-Tools (Adobe Experience Design, Figma, Storybook)

Auch auf der Infrastrukturseite ist einiges im Fluss: In Unternehmen bleiben Serverracks zunehmend leer und Anwendungen werden in die Cloud gehoben (vorrangig Amazon Web Services, Google Cloud Platform und Microsoft Azure). Anwendungen laufen in portablen Containern (Docker, Kubernetes). Infrastruktur lässt sich mit Code vollautomatisch aufsetzen (Terraform).

Die Webentwicklung hat demnach Berührungspunkte mit praktisch allen Disziplinen der Programmierung, und umgekehrt hat fast jede Anwendung heute eine Anbindung mit dem Web. In der Webentwicklung kann man sich wahlweise einen Schwerpunkt suchen oder als Full-Stack-Entwickler die komplette Plattform im Blick behalten. Webentwickler sind fast immer mehrsprachig unterwegs: Im Frontendbereich trifft man typischerweise die Hypertext Markup Language (HTML) für die Präsentation, Cascading Stylesheets (CSS) für das Styling und JavaScript für die Logik an.

_Wie du fast alles ins Web portieren kannst

Zumindest bis jetzt. Denn die Sprachsituation im Web hat WebAssembly (wasm) ordentlich aufgemischt. Hierbei han-

delt es sich um einen Binärcode für das Web, der durch die JavaScript-Runtime des Zielbrowsers direkt ausgeführt wird, ohne – wie JavaScript – erst geparst und kompiliert werden zu müssen. Je nach Anwendungsfall lässt sich eine nahezu native Performance erreichen. Mit wasm können Entwickler Quelltext von so gut wie jeder Programmiersprache, mit Sicherheit aber von C, Go, Rust, Java oder C# aus, auch für das Web kompilieren. Damit lassen sich Bestandteile von Altanwendungen ins Web migrieren oder komplett neue Apps auf Basis alternativer Sprachen entwickeln. Voraussetzung dafür ist allerdings, dass nur Funktionen vorliegen, die man auch in JavaScript ausführen könnte. Insbesondere ist kein wahlfreier Zugriff auf native Schnittstellen möglich.

In Kombination mit WebGL, einer Schnittstelle für 3-D-Inhalte im Web sowie der Gamepad-API, die Webanwendungen auf angeschlossene Controller zugreifen lässt, ist WebAssembly außerdem für Spiele oder aufwendige 3-D-Anwendungen geeignet. Sowohl die Unity- als auch die Unreal Engine können nach wasm kompilieren. Alle genannten Techniken werden von den vier großen Browsern Google Chrome, Microsoft Edge, Mozilla Firefox und Apple Safari von Haus aus unterstützt.

Wie sehr kann das Web native Ansätze ersetzen?

Das Web ist nicht nur für Informationen gut geeignet, sondern im letzten Jahrzehnt auch als Anwendungsplattform erstarkt. Heute ist es für eine Vielzahl von Programmen interessant. Anwendungen wie Slack, Visual Studio Code oder Skype basieren bereits auf Webtechnologien, benötigen heute für den vollen Funktionsumfang jedoch noch native Wrapper. Das Ziel ist, dass Entwickler die überwiegende Mehrzahl von Anwendungen direkt im Web implementieren können. Programme, die eine tiefere native Integration voraussetzen, etwa Treiber, Systemprogramme, Virenscanner oder Cloud-Sync-Clients dürften jedoch auch weiterhin eher zu nativen Techniken greifen.

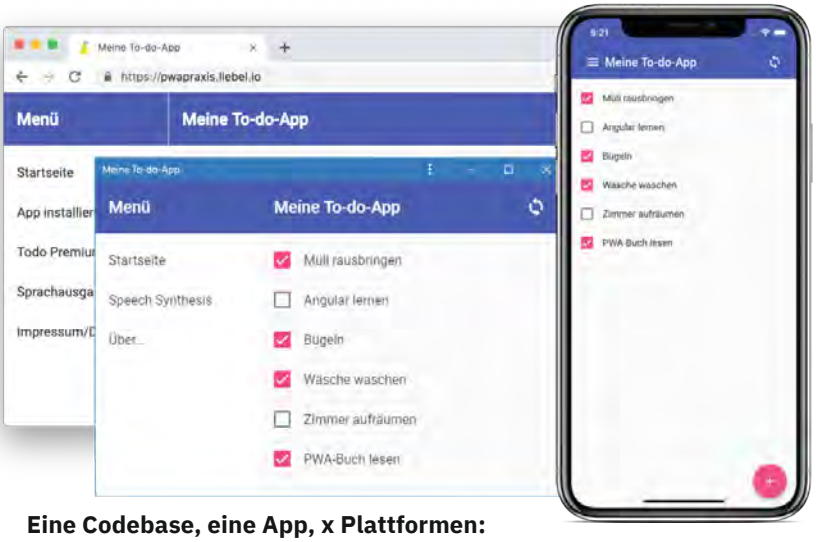
_Progressive Web Apps: ein webbasiertes Anwendungsmodell

Schon früh versuchten Entwickler, auch umfangreiche Anwendungen ins Web zu bringen (Rich Internet Applications). Da das Web anfangs noch nicht mächtig genug war, ging man es an, fehlende Fähigkeiten über Plug-in-Schnittstellen dorthin zu bringen: Java-Applets, Adobe Flash oder Silverlight waren solche Versuche, über die Webanwendungen auf bestimmte native Schnittstellen zugreifen konnten. Das Problem bei diesen Ansätzen ist, dass sie die Installation der jeweiligen Programmierplattform auf dem Gerät voraussetzten. Das war aber nicht überall möglich, iOS hat keine der genannten Plug-in-Techniken je unterstützt. HTML5 hat dann zunehmend viele Schnittstellen mit nativer Power im Web eingeführt, die Browser direkt implementieren: Dazu gehören zum Beispiel die Wiedergabe von Video- und Audio-Inhalten, die hardwarebeschleunigte Darstellung von 2-D- und 3-D-Inhalten oder das Abrufen des aktuellen Standorts für standortbasierte Dienste – all das ohne Plug-in. Diese Schnittstellen machten das Web als Anwendungsplattform attraktiv.

Progressive Web Apps (PWA) heben den Gedanken der webbasierten Anwendungsentwicklung auf die nächste Stufe: Webanwendungen sollen nicht nur Zugriff auf mächtige Features haben, sondern auch offline funktionieren und so aussehen wie native Anwendungen – und das plattformübergreifend, auf Mobil- und Desktop-Plattformen. Zentrale technische Mittel für PWA sind der Service Worker und das Web App Manifest: Ein Service Worker ist ein JavaScript-Schnipsel, der für eine bestimmte Website installiert wird. Der Service Worker bekommt einen Cache zur Seite gestellt, in dem er Dateien ablegen kann. Bei der Installation legt der Service Worker eine Kopie der Anwendungsquelldateien im lokalen Zwischenspeicher ab. Somit lässt sich die PWA künftig aus dem Cache laden. Damit wird die Anwendung nicht nur offlinefähig, sondern startet auch noch besonders schnell. Außerdem lässt sich der Service Worker losgelöst von der Website ausführen und damit zum Beispiel Push-Benachrichtigungen entgegennehmen oder Daten im Hintergrund synchronisieren.

Um schneller auf die Progressive Web App zugreifen zu können, haben Anwender die Möglichkeit, über den Browser eine Verknüpfung auf dem Home-Bildschirm beziehungsweise der Programmliste des Betriebssystems anlegen zu lassen. Hier kommt das Web App Manifest ins Spiel. Es definiert das Aussehen von auf dem Gerät installierten An-

Unreal Engine im Browser: Beeindruckende Zen-Garden-Demo mit WebGL und WebAssembly (Abb. 1)



Eine Codebase, eine App, x Plattformen: Progressive Web Apps machen's möglich (Abb. 2).

wendungen, darunter den Namen der Anwendung, das darzustellende Symbol sowie den Anzeigemodus. Auf Wunsch lässt sich die Anwendung im Standalone-Modus starten. Die App läuft auf Mobilgeräten dann vollflächig, auf Desktop-Betriebssystemen in einem eigenen Fenster.

Seit 2018 lassen sich Progressive Web Apps auf allen relevanten Mobilplattformen (iOS, Android) und Desktop-Betriebssystemen (Windows, macOS, Linux) installieren und offline ausführen. Der Vorteil der Cross-Plattform-Entwicklung auf Basis von Webtechniken gegenüber nativen Ansätzen liegt darin, dass Programmcode nur einmal zu schreiben ist und sich danach auf einer Vielzahl von Plattformen ausführen lässt. Daher braucht es nur noch ein Entwicklungsteam für alle Zielplattformen. Features stehen gleichzeitig und in gleicher Güte überall zur Verfügung. Durch die kleinere Codebase ergibt sich zudem eine geringere Fehleranfälligkeit. Das PWA-Modell eignet sich insgesamt vor allem für kleinere und mittelgroße Unternehmen.

_Die Macht des Kugelfisches

Und damit noch nicht genug. Im Rahmen seines Web Capabilities Project, besser bekannt unter dem Codenamen Project Fugu, möchte Google die Lücke zwischen Progressive Web Apps und nativen Anwendungen durch das Einführen vieler weiterer Webschnittstellen schließen. Auch die Beitragenden zum Webbrowser-Unterbau Chromium, Intel und Microsoft haben sich der Mission angeschlossen. Die Liste der geplanten APIs ist lang: So sollen PWAs zum Beispiel auf das native Dateisystem oder umfassend auf die Zwischenablage zugreifen können. Damit ließen sich im Web etliche Anwendungen umsetzen, für die es heute noch nativer Apps oder Wrapper wie Apache Cordova oder GitHub Electron bedarf: Bild- und Videobearbeitungsprogramme, integrierte Entwicklungsumgebungen oder Büroanwendungen. Darüber hinaus sind eine Autostart-Funktion, das Steuern von USB-Geräten, das Übernehmen von per SMS eingesandter Einmalpasswörter oder das Steuern der Touch-Bar unter macOS geplant. Die Schnittstellen gehen dabei über die im Web vorgesehenen Standardisierungsprozesse und werden direkt in Chromium und darauf basierenden Browsern wie Chrome, Edge oder Samsung Internet implementiert. Damit können Webentwickler in nächster Zeit viele weitere Schnittstellen im Web erwarten. Offen bleibt, welche der Fugu-Schnittstellen auch Mozilla und Apple implementieren. Im Falle der Web Share API, über die sich Inhalte aus dem Browser heraus über den nativen Teilen-Dialog streuen lassen, hat das immerhin schon geklappt.

_Das braucht es, um in die Webentwicklung einzusteigen

Einer der Grundgedanken im Web ist, dass alle Schnittstellen kostenfrei, also ohne Lizenzen und Ähnliches, verwendbar sind. Auch die Spezifikationen sind öffentlich einsehbar, ohne dass eine Gebühr dafür zu zahlen oder ein Abo abzuschließen ist. Es stehen zudem kostenfreie Entwicklungsumgebungen wie Microsofts Visual Studio Code bereit. Anwendungen und Bibliotheken legen oftmals auch ihren Quelltext offen. Damit können sich Entwickler aus aller Welt an der Weiterentwicklung von Software beteiligen oder von der Arbeit anderer lernen. Viele Cloud- und Hosting-Anbieter haben kostenfreie Abonnements, sodass nicht einmal für die Bereitstellung von Webanwendungen etwas zu zahlen ist. Die Hürden für den Einstieg als Webentwickler sind gering. Weiterhin musst du dir nicht alles selbst erarbeiten. Es gibt massenweise Lehrmaterial, auf das du zurückgreifen kannst. So finden sich auf YouTube viele kostenfreie Tutorials zum Erlernen von Webtechniken, es gibt Bücher, Blogbeiträge und Online-Lernplattformen. Als Orientierungshilfe kann dir schließlich der Lernbereich des Mozilla Developer Network dienen (https://developer.mozilla.org/de/docs/Learn).

Nahezu alle Programmiersprachen können ins Web, Webanwendungen können auf jede Plattform und der Feature-Umfang wird zukünftig nur besser. Das Web funktioniert gut als Anwendungsplattform und seine Zukunftsaussichten sind blendend – und damit vielleicht auch deine? Die beste Zeit, um Webentwickler zu werden, ist jedenfalls jetzt.

Christian Liebel

ist Consultant bei der Thinktecture AG in Karlsruhe. Dort entwickelt er moderne Cross-Plattform-Apps auf Basis von Angular und .NET Core. Christian ist begeistert von den neuen Möglichkeiten der Anwendungsentwicklung, die sich mit HTML5 und JavaScript erschließen.

Hast du Bock, der Mensch im Mittelpunkt zu sein?

ARTIFICIAL INTELLIGENCE

- Erzeugen von Nutzen aus Daten
- Nutzung von Frameworks wie Tensorflow
- Datenanalyse mit Python und R
- Analysen, Vorhersagen von Maschinenausfällen

INFRASTRUCTURE AS CODE

- Standardisierung von Deployment- und Konfigurationsprozessen
- Automatisierung von IT-Services
- Roll-Out von Container-Plattformen auf Knopfdruck
- Einsatz von Werkzeugen wie Ansible oder Terraform

DEVELOPMENT

- Individualentwicklung mit 365°-Ansatz:
- Idee → Konzeption → Umsetzung → Support
- agile Methoden und DevOps-Toolboxen
- C#, .Net, Java, Ruby, Go, C++, Python – was sprichst du?
- kontinuierliches Deployment in Dev-, Stage- und Prod-Umgebungen

CI/CD

- Kontinuierliche Integration und Deployment von Applikations- und Infrastrukturcode
- Vom SourceCode bis zur Produktion automatisiert
- CI/CD Tools wie Git, Atlassian, Artifactory implementieren und nutzen
- Etablierung von Test- Compliance- und Security-Aspekten in der CI/CD-Chain

Als erfolgreicher IT-Dienstleister modernisieren wir die IT unserer Kunden vom Server bis zur Cloud. Wir stecken unser ganzes Know-how und viel Begeisterung in unsere Projekte, um die besten Lösungen zu erzielen. Wer unsere wertschätzende Atmosphäre einmal erlebt hat, möchte sie nicht mehr missen. Heute arbeiten 1.600 Menschen an 23 Standorten bei SVA, und wir wachsen weiter – nachhaltig und mit Verstand.

kununu TOP COMPANY www.kununu.com VON MITARBEITERN EMPFOHLEN!

kununu OPEN COMPANY www.kununu.com WIR LEBEN OFFENHEIT!

job@sva.de

www.sva.de/unternehmen/karriere_uebersicht

SVA

>Smart Contracts in Rust

Lars Hupel

Die öffentliche Kryptoplattform Ethereum hat das Konzept der Smart Contracts popularisiert. Doch die hierfür gängige Programmiersprache Solidity leidet unter systematischen Problemen, die des Öfteren zu spektakulären Sicherheitslücken führen. Eine neue Generation von Sprachen schickt sich an, diese Mankos auszugleichen. Besonders interessant ist dabei die – derzeit noch experimentelle – Möglichkeit, WebAssembly als Bytecode-Format für Smart Contracts zu benutzen. Passend dazu eignet sich Rust als Hochsprache, die WASM-Code erzeugen kann.

Ethereum ist eine erstmals 2015 von Vitalik Buterin, Gavin Wood und Jeffrey Wilcke vorgestellte Kryptowährung und Applikationsplattform. Die zeitweilig genutzte markige Bezeichnung als „World Computer“ deutet darauf hin, dass die Kryptowährung nur eine Nebensache ist und der Hauptfokus den Smart Contracts gilt. Im Gegensatz zu Bitcoin (und Derivaten), in denen nur eine primitive Stack-basierte Skriptsprache zur Verfügung steht [1][2], verfügt Ethereum über eine vollwertige virtuelle Maschine, die eine spezielle Assembler-Sprache auszuführen vermag [3].

Ein Smart Contract ist dabei vergleichbar zu einem Objekt in gängigen Programmiersprachen – eine Ansammlung an aufrufbaren Methoden –, das in der Blockchain gespeichert ist und seinen eigenen Zustand verwaltet. Während sich der Zustand durch Methodenaufrufe ändern kann, bleibt der Code stets fix.

_Programmierung von Smart Contracts

Prinzipiell kann jeder Mensch so einen Smart Contract auf der Ethereum-Blockchain aufrufen, solange er der Transaktion genügend Honorar mitschickt: das sogenannte Sprit. Es wird in Ether, der eingebauten Kryptowährung von Ethereum, gemessen. Er sorgt dafür, dass die Ausführung von Code endlich ist, das heißt niemand gratis Berechnungen ausführen kann. Ganz ähnlich wie ein Notar, der schließlich Gebühren für das Ausführen von Verträgen verlangt. Die Ethereum-VM versteht eine Reihe von Assemblerbefehlen, sodass Verträge auf der untersten Ebene etwa so aussehen:

```
PUSH1 0x80
PUSH1 0x40
MSTORE
CALLVALUE
DUP1
ISZERO
```

Solchen Code möchte aber niemand von Hand schreiben. Daher gibt es Programmiersprachen, die sich mehr oder weniger an bekannte Sprachen anlehnen, aber eben zu Ethereum-Bytecode kompilieren. Der Platzhirsch ist die Sprache Solidity, deren Syntax an JavaScript angelehnt ist. Auffällig ist zunächst, dass Solidity im Gegensatz zu JavaScript ein Typsystem besitzt. Außerdem sind bestimmte Methoden

Listing 1: Ein Beispiel-Contract in Solidity

```
pragma solidity >=0.4.22 <0.6;

contract Wallet {
    uint256 balance;
    address payable owner;
    constructor () public {
        balance = 0;
        owner = msg.sender;
    }
    function addfund() payable public returns (uint256) {
        require (msg.sender == owner);
        balance += msg.value;
        return balance;
    }
    function withdraw() public {
        require (msg.sender == owner);
        selfdestruct(owner);
    }
}
```

speziell annotiert. Im obigen Beispiel steht etwa `payable` dafür, dass sich mit der Methode `Ether` (abzüglich Sprit) in den Vertrag einzahlen lässt.

_Solidity als Standard

Ganz genau messbar ist es zwar nicht, aber Solidity hat sich mittlerweile als Standard für die Programmierung auf Ethereum durchgesetzt. Das zeigt sich auch daran, dass ein Tooling-Ökosystem um die Sprache herum entstanden ist. Allerdings ruft die Vielzahl an öffentlichen Verträgen mit teils hohen Kontoständen auch kriminelle Akteure auf den Plan. Ein Smart Contract, einmal programmiert und ausgerollt, kann keine Bugfixes mehr enthalten, was es umso wichtiger macht, dass sie hohen Qualitätsansprüchen genügen.

Doch Solidity fällt dabei als nicht besonders solide auf. Eine Forschergruppe an der University of Texas hat in einer Untersuchung [4] 44 Fehlerklassen festgestellt, von denen fünf auf Solidity zurückgehen. In der Vergangenheit wurden einige spektakuläre Fehler in Solidity-Verträgen ausgenutzt, um acht- bis neunstellige Dollarbeträge abzuzweigen.

Während sich Probleme in Programmiersprachen oftmals durch eine neue Version ausgleichen lassen, wiegen Designfehler in der Ethereum-VM deutlich schwerer. Aufgrund von Abwärtskompatibilität lässt sich Laufzeitverhalten nur schwer ändern, denn die gesamte Basis an Smart Contracts muss lauffähig bleiben.

Alles in allem erinnert das Problem stark an die Debatte, die die Community der Systemprogrammierer schon seit geraumer Zeit führt: Liegen Sicherheitslücken an schlampiger Programmierung oder sind die Ursachen in der Programmiersprache zu finden? Aus dieser Beobachtung ist die Programmiersprache Rust geboren, die etliche Sicherheitsprobleme durch besseres Sprachdesign beseitigen möchte.

_Eine vielseitige Sprache

Auf das Tapet der Systemprogrammierung kam Rust erst verhältnismäßig spät. Graydon Hoare gestaltete für Mozilla eine Programmiersprache mit dem Ziel, eine neue, sicherere Browser-Engine zu schaffen. Mittlerweile haben größere in Rust entwickelte Module Einzug in Firefox gehalten. Mozilla sponsort deswegen die Weiterentwicklung, und die Community hilft kräftig mit. Auf GitHub ist Rust eine der am stärksten wachsenden Programmiersprachen in den letzten zwei bis drei Jahren.

Das Versprechen von Rust ist es, ähnlich wie C++ durch manuelle Speicherverwaltung und Abstraktionen ohne Laufzeitkosten hohe Performance zu garantieren, gleichzeitig aber durch ein starkes Typsystem gängige Fehler zu verhindern. Beispielsweise ist es in Rust unmöglich, in einem parallelen Programm einen Data Race zu erzeugen: Der Compiler verbietet kurzerhand, dass zu einem Objekt gleichzeitig mehrere schreibbare Zeiger existieren. Andererseits stellen die Standardbibliothek und viele Pakete Abstraktionen bereit, mit denen Parallelismus generell einfacher zu handhaben ist. Mit der rayon-Bibliothek lässt sich zum Beispiel folgender Code schreiben:

```
let mut arr = [0, 7, 9, 11];
arr.par_iter_mut().for_each(|p| *p -= 1);
println!("{:?}", arr);
```

Der „mutable parallel iterator“ erlaubt Veränderungen pro Array-Eintrag, aber unterbindet, dass sich diese Zeiger außerhalb der Iteration verwenden lassen.

_Rust für Smart Contracts

Die Vorteile von Rust haben auch die Ethereum-Entwickler erkannt. Unter der EWASM-Flagge (Ethereum flavored WebAssembly) läuft ein Standardisierungsprozess für eine zweite, inkompatible Version der Ethereum-VM auf Basis des WebAssembly-Bytecodeformats („Phase 2“). WebAssembly ist ein vom W3C ursprünglich für die Browser-Plattform vorangetriebene Spezifikation, die sich aber auch außerhalb des Webs Beliebtheit erfreut. Die Vorteile von WASM als Low-Level-Format für die Ethereum-VM sind eindeutig: Zum einen ist es das praxisorientierte Werk erfahrener Sprachdesigner und nicht eine proprietäre Nischenlösung. Zum anderen steht via LLVM eine reichhaltige Basis an Programmiersprachen bereit, die nach WASM übersetzen können, so auch Rust.

Was wäre also zweckmäßiger, als Rust, eine sichere Programmiersprache, mit der soliden Basis von WebAssembly zu kombinieren, um Smart Contracts zu entwickeln, die gegebenenfalls Millionen von Krypto-Tokens verwalten?

_Eine Geldbörse in Rust

Der Vertrag für eine einfache Geldbörse, der oben in Solidity wiedergegeben ist, lässt sich auch in Rust formulieren. Als Erstes fällt auf, dass man in Rust die Schnittstelle von der Implementierung trennen muss (s. Listing 2).

Der Code definiert ein Trait; ein Interface in Rust, das man später implementieren kann. Die Besonderheit besteht darin, dass alle Traits in Rust implizit einen Typparameter haben (`Self`). Ähnlich wie in Python ist das Objekt, auf dem eine Methode aufgerufen wird, explizit mitzuübergeben (`self` vom Typ `Self`). Die Annotation am Trait (`eth_abi`) sorgt dafür, dass der Rust-Compiler automatisiert ABI-Definitionen erzeugt. Unter ABI (Application Binary Interface) versteht man die Konventionen, mit denen Funktionsaufrufe im kompilierten Bytecode stattfinden.

Für Ethereum ist das notwendig, da die EVM keine Methoden vorsieht, sondern jeder Smart Contract nur einen einzigen Einstiegspunkt definiert. Stattdessen bildet man einen Hash aus gewünschtem Funktionsnamen und -parametern und übergibt diesen an den Smart Contract, der dann üblicherweise per switch-ähnlichem Statement an die richtige Stelle springt. EWASM hat diese Konvention von Solidity übernommen. Die Annotation erzeugt den nötigen Boilerplate, damit sowohl Aufrufer als auch aufgerufener Vertrag dieses Hashing nicht umständlich per Hand implementieren müssen. Ein Baustein bleibt aber noch übrig, nämlich der Einsprungpunkt für den Vertrag (dazu später mehr). Im zweiten Schritt definiert man dann die tatsächliche Datenstruktur, die dem Vertrag zugrunde liegt. Im Regelfall kann diese leer ausfallen:

```
pub struct WalletContract;
```

Zunächst sieht das nicht intuitiv aus. Wo genau soll denn der Vertrag nun speichern, wer die Eigner sind und wie viele Ether sie momentan abgelegt haben? Dazu müssen wir kurz auf das Speichermodell von Ethereum eingehen.

_Storage und Felder

In Solidity muss man sich um Storage keine Gedanken machen, denn die Programmiersprache suggeriert das Verwalten von abstrakten Objekten. Definiert man in einem Vertrag eine Reihe von Objektfeldern, dann verhalten sie sich so, wie man es von gängigen Programmiersprachen gewohnt ist. Allerdings handelt es sich dabei nur um eine Abstraktion. Tatsächlich verwaltet die Ethereum VM für persistenten Speicher nur eine Menge von Registern, und zwar 2256 Stück zu je 256 Bit Breite. Was wie eine unvorstellbare Menge erscheint, wird schnell dadurch relativiert, dass das Schreiben und Lesen in ein persistentes Register teuer ist, das heißt einen hohen Sprit-Preis hat. Deswegen wird bei der Programmierung peinlich genau darauf geachtet, möglichst wenig Speicher zu verschwenden. In Konsequenz heißt das auch, dass Ethereum-Nodes die Register komprimiert speichern können und letztlich nur wenig Plattenplatz benötigt wird. Solidity versucht demzufolge, die abstrakten Variablen auf ein möglichst effizientes Registerlayout abzubilden. Dazu gibt es verschiedene Strategien, zum Beispiel mehrere 4-Byte-Werte in ein einziges Register zu packen. Trickreich wird es, wenn dynamische Strukturen wie Arrays mit flexibler Größe oder Hashtables abgelegt werden sollen. Solidity nutzt dafür Hashingverfahren und versteckt das als Implementierungsdetail vor den Nutzern. Die Details lassen sich aber in der Dokumentation nachlesen [6].

_Speicherverwaltung von Rust

Rust hingegen hat ganz andere Ansprüche an Programmierer. Code soll sich ohne „Schnickschnack“ möglichst direkt auf Speicherlayout abbilden lassen. Diese Einstellung kommt aus den C- und C++-Welten, in denen manuelle Speicherverwaltung Usus ist. Das bisher noch experimentelle Ether-

Listing 2: Schnittstelle für den Wallet-Vertrag

```
#[eth_abi(WalletEndpoint, WalletClient)]
pub trait WalletInterface {
    fn constructor(&mut self);

    #[constant]
    fn owner(&mut self) -> Address;
    #[constant]
    fn balance(&mut self) -> U256;
    fn addfund(&mut self) -> bool;
    fn withdraw(&mut self) -> bool;
}
```

eum-SDK für Rust lässt einen mit der Registerallokation weitgehend allein, sodass man die Adressen selbst ausrechnen muss. Das ist der Grund, warum der struct zum Contract leer geblieben ist: Es gibt keine Notwendigkeit dafür, irgendwelche Werte im Stack abzulegen. Stattdessen deklariert man sich zwei globale Konstanten (Listing 3).

Würde man das automatisieren wollen, müsste man sich ein Makro schreiben, das ähnlich wie `eth_abi` die Stuktur des Vertrags analysiert und Register passend alloziert. Der große Vorteil von Rust-Makros gegenüber dem C-Präprozessor ist, dass man damit ASTs manipulieren kann, statt grobkörnig Text zu ersetzen. Der dritte Schritt ist die Implementierung der tatsächlichen Logik des Vertrags (Listing 4).

Die Abläufe sind ähnlich zum Solidity-Pendant, aber es ist deutlich zu sehen, dass das SDK weniger Hilfestellung leistet. Insbesondere muss man häufig zwischen verschiedenen Zahlentypen konvertieren (Hashes, Unsigned Integer, Byte-Arrays). Der Registerzugriff läuft hier über spezielle Funktionen, die die pwasm_ethereum-Bibliothek bereitstellt.

Listing 3: Statische Variablen für die Registeradressen

```
lazy_static! {
    static ref OWNER_KEY: H256 =
      H256::from([2,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1]);
    static ref BALANCE_KEY: H256 =
      H256::from([3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1])
}
```

Intern handelt es sich dabei um dünne Wrapper über EWASM-Primitive, die also in der VM implementiert sind.

_Call & Deploy

Als letzten Schritt definieren wir noch die Einstiegspunkte für den Vertrag für die beiden Fälle des initialen Deployments und normaler Aufrufe (Listing 5).

Die beiden Funktionen sind bei den meisten Smart Contracts identisch. Die einzige Aufgabe besteht darin, die eingehenden Argumente zu verarbeiten und an die automatisch (per `eth_abi`) generierten `dispatch-` und `dispatch_ctor-Methoden` weiterzuleiten. Sie kümmern sich um die Auswahl der korrekten Smart-Contract-Methode.

Listing 4: Implementierung des Wallet-Vertrags

```
impl WalletInterface for WalletContract {
    fn constructor(&mut self) {
        let sender: [u8; 32] = H256::from(pwasm_ethereum::sender()).into();
        pwasm_ethereum::write(&OWNER_KEY, &sender)
    }

    fn owner(&mut self) -> Address {
        H256::from(pwasm_ethereum::read(&OWNER_KEY)).into()
    }

    fn balance(&mut self) -> U256 {
        pwasm_ethereum::read(&BALANCE_KEY).into()
    }

    fn addfund(&mut self) -> bool {
        let sender = pwasm_ethereum::sender();
        if sender != self.owner() {
            false
        }
        else {
            let new_balance: [u8; 32] = (self.balance() + pwasm_
ethereum::value()).into();
            pwasm_ethereum::write(&BALANCE_KEY, &new_balance);
            true
        }
    }

    fn withdraw(&mut self) -> bool {
        let sender = pwasm_ethereum::sender();
        if sender != self.owner() {
            false
        }
        else {
            pwasm_ethereum::suicide(&sender)
        }
    }
}
```

_Programmierkomfort?

Rust hat sich im Laufe der Zeit zu einer durchaus komfortablen Programmiersprache mit herausragender Tooling-Unterstützung gemausert, wobei das strikte Typsystem einige Eingewöhnung abverlangt. In den Beispielen des Artikels zeigt sich allerdings, dass das EWASM-SDK für Rust noch nicht ausgereift ist. Diese Begrenzungen sind allerdings nicht systematisch, sondern nur auf die derzeitige Experimentierphase zurückzuführen. In Zukunft könnten sich weitere Bibliotheken und Frameworks herausbilden, mit denen Smart Contracts ähnlich komfortabel wie in Solidity zu programmieren sind. Doch bis dahin ist es noch ein weiter Weg. So fehlt es noch an vielen Dingen, etwa einer breiten Implementierung (bislang unterstützt nur der Parity-Client EWASM).

Einige Vorteile zeichnen sich aber ab. Sowohl Rust als auch WASM sind Techniken mit Zukunft. Das starke Typsystem lässt sich dafür nutzen, Verträge sicherer zu gestalten und gängige Sicherheitslücken zu tilgen. Mit WASM als Kompilierziel sind polyglotte Sprach- als und Ausführungsszenarien denkbar: Der gleiche Code ließe sich dann klassisch auf Client oder Server, aber auch mit minimalen Änderungen auf der Blockchain auszuführen. Den gesamten Rust-Code findet man auf GitHub [7].

Listing 5: Aufruf- und Deploymentfunktion

```
#[no_mangle]
pub fn call() {
    let mut endpoint = WalletEndpoint::new(WalletContract{});
    pwasm_ethereum::ret(&endpoint.dispatch(&pwasm_ethereum::input()));
}

#[no_mangle]
pub fn deploy() {
    let mut endpoint = WalletEndpoint::new(WalletContract{});
    endpoint.dispatch_ctor(&pwasm_ethereum::input());
}
```

Literatur

[1] Mirko Dölle; Bitcoins doppelt sicher – Multisignatur-Bitco in-Wallets als Diebstahlschutz, in: c't 4/2020

[2] Lars Hupel; Adressen und Transaktionen in Kryptowährungen – Teil 1: Bitcoin (www.innoq.com/de/articles/2019/05/kryptowaehrungen-transaktionen-teil-1/)

[3] Hajo Schulz; Vertrag denkt mit – Smart Contracts in der Ethereum-Blockchain, c't 23/2017

[4] Huashan Chen, Marcus Pendleton, Laurent Njilla, Shouhuai Xu; A Survey on Ethereum Systems Security: Vulnerabilities, Attacks and Defenses (arxiv.org/pdf/1908.04507.pdf)

[5] The State of the Octoverse (octoverse.github.com)

[6] Layout of State Variables in Storage, in Solidity-Dokumentation (solidity.readthedocs.io/en/v0.6.3/miscellaneous.html)

[7] GitHub zum Artikel (github.com/larsrh/ewasm-example)

Lars Hupel

ist Consultant bei INNOQ in München und bekannt als einer der Gründer der Typelevel-Initiative, die sich der Entwicklung von typgetriebenen Scala-Bibliotheken in einer einsteigerfreundlichen Umgebung verschrieben hat. Er spricht oft auf Konferenzen und ist im Open-Source-Umfeld in Scala unterwegs. Außerdem programmiert und redet er gern über Haskell, Prolog und Rust.

VATTENFALL

Will you join us at the forefront of innovative IT solutions? Together we can solve the future challenges of the energy sector.

Start your climate smarter career now!
https://careers.vattenfall.com/de

>CODEQUALITÄT LEHREN UND LERNEN

Linus Dietz, Simon Harrer

Dieser Artikel fasst die Lehren aus über sechs Jahren Programmierausbildung an der Universität zusammen und empfiehlt Maßnahmen, um vor dem Berufseinstieg für eine bessere Ausbildung zu sorgen und eine sinnvolle Weiterbildung zu etablieren.

2011 hatte die Universität Bamberg die Programmierausbildung überdacht. Auf der Erstsemestervorlesung „Einführung in der Informatik“ aufbauend, sollten zwei weiterführende, praktische Programmierkurse geschaffen werden. Als junge Doktoranden hatten die Autoren dieses Artikels die Chance, ein neues Lehrkonzept [1] zu entwickeln, das den Studierenden nicht nur weiterführende Programmierkonzepte beibrachte, sondern den Fokus auf das Schreiben qualitativ hochwertigen Codes legte.

_Ein auf Codereviews aufbauendes Lehrkonzept

Das Lehrkonzept besteht aus einer Plenarübung im Rechnerraum, die zwischen dem Vermitteln neuer Programmierkonzepte und deren sofortiger Anwendung in kleinen Übungsbeispielen alteriert. Die Lösungsvorschläge werden im Plenum interaktiv begutachtet und so lange unter Moderation der Dozenten refaktoriert, bis es keine Verbesserungsvorschläge mehr gibt und die Vor- und Nachteile aller Alternativen abgewogen sind. Die Diskussionen sind essentiell, um den Studierenden nicht nur ein Gespür für Verbesserungen zu geben, sondern auch um die Erwartungshaltung in Bezug auf Codequalität in den Programmieraufgaben zu setzen.

Der zweite Pfeiler der Lehrveranstaltungen besteht aus mehreren Programmierprojekten, die innerhalb von zwei Wochen in Kleingruppen zu lösen sind. Die Dozenten bewerten die Projekte, die circa fünf bis 15 Java-Klassen mit bis zu 2000 Codezeilen umfassen, schriftlich hinsichtlich funktionaler Korrektheit, aber auch Architektur und Codequalität. Dabei teilen sie nicht nur die erreichte Punktzahl mit, sondern gehen wie in einem Codereview detailliert auf die Codequalität und Verbesserungsmöglichkeiten mit Verweis auf die Codestelle ein. Neben dem individuellen Feedback gibt es jeweils eine Sammlung gehäuft auftretender Fehler, die mitsamt ihrer Verbesserung an alle im Kurs verschickt werden. Aus der Sammlung der typischen Fehler entstand mit der Zeit das Buch „Java by Comparison“ [2].

Die mündliche Prüfung am Ende des Semesters fragt schließlich den erlernten Stoff anhand des Codes aus den Projekten ab. Die Studierenden sind aufgefordert, in der IDE den eigenen Code zu erklären und kritisch zu bewerten. Zusätzlich bekommen sie unbekannte Codebeispiele vorgelegt, die sie wie in einem Codereview beurteilen sollen.

Im Folgenden stellen die Autoren nun die Maßnahmen zum Erreichen einer besseren Codequalität vor, wie wirksam sie waren und wie sie sich auch in Unternehmen für die gezielte Weiterbildung von Berufseinsteigern nutzen lassen können.

_Erfahrungen bei der Fachliteratur

Wie an einer Universität üblich haben wir Fachbücher wie „Clean Code“ [3] oder „Effective Java“ [4] in großzügiger Anzahl über die Bibliothek angeschafft, in der Lehrveranstaltung empfohlen und in deren Verlauf immer wieder darauf verwiesen. Ausgeliehen haben sie allerdings nur wenige. Über die Jahre war klar: Nur die Besten haben die Fachliteratur freiwillig studiert, der Rest gab sich mit den Vortragsinhalten zufrieden. Aus Dozentensicht war das trotz des geringen Aufwands unbefriedigend, da wir ja auch die Breite erreichen wollten.

In der Industrie sind unsere Erfahrungen ähnlich: Die Besten bilden sich freiwillig, oft auch in der Freizeit, selbstständig weiter. Diejenigen, die dazu nicht bereit sind oder es aus privaten Gründen einfach nicht möglich machen können, fallen zurück. Das bedeutet, dass die Anschaffung von Büchern in der Abteilungsbibliothek, die Bereitstellung von E-Books im Intranet oder das Abo einer Online-Lernplattform eben nicht ausreichen.

In unseren Kursen hatten wir auch die Möglichkeit, Pflichtlektüren als Hausaufgabe zu geben, die dann gemeinsam im Plenum besprochen wurden - zum Beispiel ein Kapitel aus Martin Fowlers Refactoring-Buch [5] oder ein Artikel zum Thema Concurrency [6]. Dadurch dass die Inhalte von überschaubarem Umfang waren und zudem genau zum Stoff passten, haben die meisten sie gelesen und die fachliche Diskussion war dementsprechend direkt von Anfang an auf einem hohen Niveau. In der Ausbildung ist eine solche Pflichtlektüre mit Nachbesprechung in der Veranstaltung ein geeignetes Mittel, um ein wichtiges Thema voranzutreiben. Das ist auch eine Empfehlung, die in Unternehmen funktionieren kann, wenn es gelingt, sowohl für das Vorbereiten als auch für die Besprechung Arbeitszeit vorzusehen.

> **Wenn eine Metrik zum Ziel wird, ist sie keine gute Metrik mehr.**
>
> Charles Goodhart

_Statische Codeanalyse

Statische Codeanalyse verspricht ein unvoreingenommenes, vollständiges und vor allem günstiges Codereview, das Code-Smells automatisch und zuverlässig aufdeckt. Sie verhindert, dass unausgereifter Code in den Produktivbetrieb kommt. Von diesen Versprechungen geleitet haben sich entsprechende Werkzeuge stark verbreitet.

In der Lehre haben wir das auch ausprobiert. Wir stellten den Studierenden die einschlägigen Open-Source-Werkzeuge vor und erklärten die Einbindung in die IDEs. Das kostete Zeit, die wir in der Hoffnung auf weniger Korrekturaufwand gerne investierten. Das Ergebnis blieb weit hinter den Erfahrungen zurück. Zum einen zeigte sich, dass viele sowohl die Codeanalysewerkzeuge als auch die Standardwarnungen der IDE komplett ignorierten. Des Weiteren stieg der Betreuungsaufwand während der Programmieraufgaben, da immer wieder Rückfragen kamen, wie denn Meldung „X“ zu beseitigen wäre. Bei näherer Betrachtung waren das meist false-positives beziehungsweise komplett irrelevante Regeln, die getrost hätten deaktiviert werden können. Im schlimmsten Fall sahen wir obskure Workarounds für relativ einfache Funktionen, um die Warnungen loszuwerden.

Die Einführung statischer Codeanalysewerkzeuge war also mit einem mittleren Aufwand verbunden, brachte aber kaum etwas. Teils wurden die Werkzeuge ignoriert, teils gingen insbesondere die motivierten Studierenden über große Längen, um alle Hinweise systematisch zu eliminieren, was wiederum nicht förderlich für den Lernprozess war. Hier greift das Goodhart'sche Gesetz: „Wenn eine Metrik zum Ziel wird, ist sie keine gute Metrik mehr.“

Im Unternehmenskontext gibt es häufig folgendes Muster: Jemand, dem Codequalität wichtig ist, stellt stichprobenhaft fest, dass sie nicht ausreichend ist. Da die Person nicht die Zeit hat, alles selbst zu reviewen, sorgt sie für die Einführung von Codeanalysewerkzeugen. Um alle Fehlerquellen abzudecken, werden dann viele der im Werkzeug vorhandenen Regeln aktiviert und als Vorgabe ausgegeben. Infolgedessen geht die Produktivität des Teams nach unten: Es wird etwa der Getter des Feldes „firstname“ mit „Gets the firstname“ dokumentiert. Nur hat das nichts mit Codequalität zu tun, sondern ist eine Arbeitsbeschaffungsmaßname.

Dass Unternehmen statische Codeanalyse „erfolgreich“ einsetzen, möchten wir nicht in Abrede stellen. Aus unserer Erfahrung ist sie allerdings kein geeignetes Mittel, um ein Gespür für hochqualitativen Code zu erlernen. Für Studierende, aber auch für Berufseinsteiger läuft der Einsatz statischer Codeanalyse zumeist auf das unreflektierte Befolgen von Regeln hinaus, statt die Dinge zu hinterfragen. Selbst bei einer sinnvoll konfigurierten Regelauswahl tritt oftmals ein ähnlicher Effekt wie bei der Fachbuchanschaffung ein: Den Guten hilft es, noch einen Flüchtigkeitsfehler zu tilgen, die Schwächeren werden frustriert und ignorieren die Werkzeuge in Gänze. Hilfe könnte sein, die Regeln gemeinsam im Team zu besprechen und zu diskutieren – und dabei eben ein Gespür zu schaffen, wann man nach Benedikt von Nursia eher die Regel beugen soll als den Menschen.

_Pair Programming und Mob Programming

Wenn mehrere Entwickler in einem Projekt zusammenarbeiten, ist der Drang da, die Arbeit irgendwie aufzuteilen. Naiverweise könnte man N Entwicklern jeweils 1/N der Funktionalität zur Implementierung geben und nach einer gewissen Zeit die Einzelteile zu einem Produkt zusammensetzen. Klassisches „Divide and Conquer“ anhand definierter Schnittstellen. So ähnlich haben viele unserer Teams die Projekte bearbeitet. Leider haben die Teile nach gut einer Woche Programmierung nicht immer so zusammengepasst wie geplant. Auch wenn das Zusammenstöpseln zur Abgabe noch geklappt hat, zeigte sich bei der Prüfung, dass jeder nur den eigenen Teil gut konnte. Die Versuche der Studierenden, sich den Code gegenseitig „vorzustellen“, haben oft nicht gereicht. Ganz anders war das bei denen, die den Code gemeinsam entwickelt haben.

Beim Pair Programming bedient die eine Person das Keyboard, während die andere über die nächsten Schritte nach-

denkt. Diese Art, ein Programm zu entwickeln, ist für Neulinge keine leichte Sache. Sie ist aber der beste Weg, um eine hohe Codequalität zu erreichen. Nicht nur, dass vier Augen mehr sehen als zwei; um auf eine gemeinsame Lösung zu kommen, muss diskutiert und abgewogen werden. Das erfordert Übung und eine hohe Sozialkompetenz. Am Anfang ist für viele der soziale Druck hoch, wenn sie sich oft vertippen oder ihnen die Funktionsnamen der Standardbibliothek nicht mehr einfallen. Hier wollen wir nicht darauf eingehen, wie Pair Programming funktionieren kann [7], sondern dazu ermutigen, es einfach auszuprobieren.

Mob Programming [8] geht sogar noch einen Schritt weiter. Hier sitzen mindestens drei Personen vor einem Rechner und erreichen dadurch eine noch bessere Qualität. Wir haben das als didaktisches Mittel in der Lehrveranstaltung bei der Besprechung der Übungen genutzt, sprich mit allen Studierenden vor dem Beamer. Es ist ein großer Aufwand, der sich dann rechtfertigen lässt, wenn die Qualität und der Wissenstransfer maximiert werden sollen.

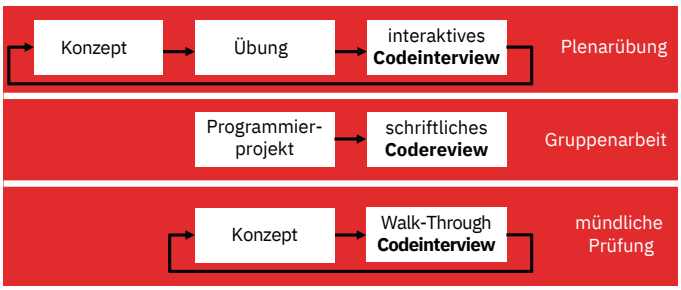
Studierende, die wirklich gemeinsam die Projekte bearbeitet hatten, lieferten nicht nur bessere Ergebnisse, sondern brachten auch in der mündlichen Prüfung die besseren Argumente. Unternehmen können daraus lernen, indem sie vor allem ihren neuen Entwicklern die Möglichkeit geben, zu zweit oder in Gruppen zu programmieren. So können diese schnell eingelernt werden und früh zum Produktivcode beitragen. Insbesondere das Pairing von Berufseinsteigern mit erfahrenen Kräften maximiert den Wissenstransfer und kann ein traditionelles Onboarding größtenteils ersetzen. Die Motivation ist dank schneller Fortschritte höher, der Bildung von Wissenssilos wird entgegengewirkt. Deswegen die Empfehlung an Young Professionals: Macht so viel Pair und Mob Programming wie möglich, um von den erfahrenen Kollegen zu lernen.

>So viel Pair und Mob Programming wie möglich, um von Kollegen zu lernen.

_Best Practices mit Codereviews

Codereviews sind das zentrale Element in den Lehrveranstaltungen, da wir überzeugt sind, dass die gemeinsame Diskussion das geeignete Mittel ist, um Codequalität zu erlernen. Wie eingangs beschrieben, bestanden die Lehrveranstaltung, die Korrektur der Hausaufgaben, die Sprechstunden und Tutorien und auch die Abschlussprüfung aus Codereviews. Das schult das Auge für Code-Smells und verbessert die Argumentationsfähigkeit.

Der Erfolg von einem Codereview hängt daran, dass Lernende verstehen, warum ihre Implementierungsvariante ein Problem darstellt. Denn nur wenn sie dieses einsehen, können sie eine Lösung als Verbesserung annehmen. Das bedeutet, dass man den Code nicht nur knapp kritisieren sollte, sondern durchaus die Hälfte des Texts über das Problem



Das didaktische Konzept basierte auf Codereviews.

schreiben kann. Auf der Lösungsseite schlägt man idealerweise eine konkrete Verbesserung in Form von Code vor und beschreibt deren Vorteile knapp.

Unternehmen raten wir deshalb, Ähnliches zu tun. Einsteiger brauchen Feedback, mit dem sie etwas anfangen können, um sich zu verbessern. Die Abnahme von Code wird viel einfacher, wenn sie in kleinen Häppchen passiert und früh die Kriterien für die Codequalität diskutiert werden. Hierbei ist Pairing optimal, aber auch schriftliche Codereviews in einem Pull-Review-Workflow sind wertvoll, da er üblicherweise von jemandem verfasst wird, der an der ursprünglichen Entwicklung unbeteiligt war. So bekommt man eine weitere Perspektive auf den Code. Wenn man nur begrenzte Zeit hat, sollte man ökonomisch vorgehen: zuerst die schlimmen Probleme angehen, danach die weniger gewichtigen.

_Das Ziel ist eine Lernkultur

Universitäten sind ein Ort des Lernens. Es sollte aber jedem klar sein, dass es mit einem Abschluss nicht getan ist, sondern es im Berufsleben mit dem Lernen genauso weitergeht, nur dass es vielleicht nicht mehr so klar ist, wer die Dozenten sind. Als (Young) Professionals sind alle gleichermaßen Dozenten und Lernende. Deshalb ist es wichtig, dass jeder einen Lernanspruch an sich selbst hat sowie das Lernen im Unternehmen wertgeschätzt und entsprechende Maßnahmen gefördert werden. Durch das Vorleben dieses Anspruches in der Lehrveranstaltung konnten wir unsere Erwartungen an die Studierenden setzen und waren dementsprechend zufrieden mit deren Lernerfolg. Dazu benötigten wir jedoch viel Zeit für Diskussionen in und nach der Lehrveranstaltung und noch mehr zeitlichen Aufwand für die schriftlichen Reviews der Hausaufgaben.

Der Aufwand war ebenso groß bei den Studierenden, die sich regelmäßig darüber beschwert haben. Rückblickend hat sich aber gezeigt, dass wir viele richtige Entscheidungen getroffen haben. Dafür sprechen nicht nur die sechs Nominierungen und schließlich der Gewinn des Preises für gute Lehre, sondern auch, dass diejenigen, die bei uns im Kurs sehr gut waren, das Wissen später auch im Berufsleben rüberbrin-

gen konnten. Für die Leser empfehlen wir: Schafft eine Lernkultur und greift zum beschriebenen Werkzeugkasten, der bei weitem nicht abschließend ist. Code-Retreats, Hackathons, persönliches Mentoring, Katas et cetera sollten ihn ergänzen. Der langfristige Erfolg hängt jedoch nicht an punktuellen Maßnahmen oder einem elaborierten Styleguide für guten Code, sondern von dem täglichen Anspruch an das eigene Handwerk ab. Wer neu im Team ist, der schlägt dem Team-Lead Verbesserungen vor und findet Gleichgesinnte, um die Entwicklungsprozesse fortwährend zu verbessern.

Literatur

[1] Linus Dietz, Johannes Manner, Simon Harrer, Jörg Lenhard; Teaching Clean Code. 1st Workshop on Innovative Software Engineering Education, 2018

[2] Simon Harrer, Jörg Lenhard, Linus Dietz; Java by Comparison; The Pragmatic Bookshelf 2018

[3] Robert C. Martin; Clean Code; Prentice Hall 2008

[4] Joshua Bloch; Effective Java; Addison-Wesley Professional 2018

[5] Martin Fowler; Refactoring; Addison-Wesley Professional 1999

[6] Igor Sorokin, Alex Miller; Core Java Concurrency (dzone.com/refcardz/core-java-concurrency)

[7] Kent Beck; Extreme Programming Explained: Embrace Change; Addison-Wesley Professional 2005 (2. Aufl.)

[8] Mark Pearl; Code with the Wisdom of the Crowd; The Pragmatic Bookshelf 2018

Linus Dietz
forscht an der Technischen Universität München zu Mobilität und Empfehlungsdiensten für Reisende.

Dr. Simon Harrer
ist seit zwei Jahren Senior Consultant bei INNOQ und sorgt dank Remote Mob Programming von Zuhause aus für zufriedene Kunden.

Gemeinsam haben sie aus ihrer Erfahrung in Open-Source-Software und durch die Lehre an der Universität Bamberg das Buch „Java by Comparison“ verfasst und sind regelmäßige Redner in User Groups und Softwarekammern.

>Inbetriebnahme von ML-Modellen

Nico Axtmann

Werkzeuge wie DVC und Cortex, die auf Operationalisierung von KI-Projekten ausgelegt sind, sollen Entwicklern beim Deployen von Modellen in Produktion helfen

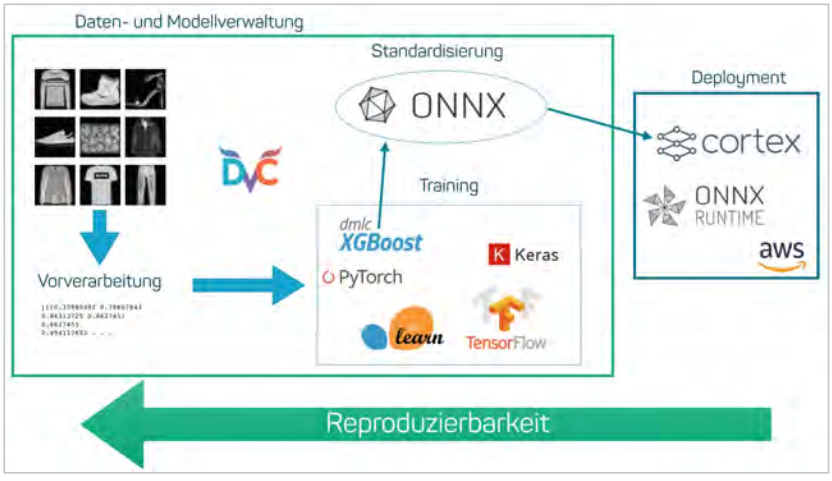
Obwohl ein großer Hype um Machine Learning und KI existiert, landet bei den meisten Unternehmen nur ein Bruchteil der entwickelten Modelle in der Produktion. Die Werterzeugung mit datengetriebenen Entscheidungen wird bei den Unternehmen in den nächsten Jahren zunehmend im Fokus stehen. Viele beginnen KI- und Datenstrategien zu implementieren. In den vergangenen Jahren sind zahlreiche Deep-Learning-Frameworks wie TensorFlow, PyTorch oder MxNet entstanden und gewachsen.

Insgesamt hat der große Anteil an Open-Source-Software die Entwicklung der Modelle stark vereinfacht. Heute erweitern viele Werkzeuge das Ökosystem, die mehr Struktur und Stabilität in die Entwicklung und das Deployment der KI-Applikationen bringen. Der Fokus der neuen Tools erstreckt sich von der Daten- und Modellverwaltung bis hin zum Deployment der Modelle auf unterschiedlichen Cloud-Plattformen (s. Abb. 1). Ziel ist es, Hilfestellung zu geben, um Modelle in Produktion zu bekommen.

_Das Vorhaben – Vogelperspektive

Dieser Artikel zeigt, wie sich KI-Projekte mit DVC, ONNX (Open Neural Network Exchange) und Cortex von der Entwicklung bis hin zur Produktion reproduzierbar und skalierbar umsetzen lassen, unabhängig vom verwendeten Deep-Learning-Framework. Unterschiedliche Frameworks kommen für das Trainieren von Modellen zum Einsatz, die man anschließend in einer homogenen Produktionsumgebung als REST-API in Betrieb nutzt.

Als Datenbasis dient Fashion MNIST von Zalando [a]. Interessierte können die Daten als gezipptes NumPy-Array herunterladen. Für den Artikel hat der Autor die Daten in ein Python-Skript eingelesen und anschließend nach ihrem Label (Ordner) als PNG-Datei abgespeichert. Fashion MNIST repräsentiert verschiedene Artikel aus dem Modebereich mit über 60.000 Graustufenbildern. Die Bilder haben jeweils



Daten- und Modellverwaltung, Training und Deployment mit verschiedenen Open-Source-Tools wie DVC, ONNX und Cortex (Abb. 1)

eine Größe von 28 x 28 Pixel. Die im Datensatz angegeben Trainings- und Testdaten kann man herunterladen. Im Anschluss lässt sich die unter Abbildung 2 dargestellte Verzeichnisstruktur erstellen. In den Verzeichnissen `train` und `test` gibt es jeweils die Unterverzeichnisse `Ankle_boot`, `Bag`, `Coat`, `Dress`, `Pullover`, `Sandal`, `Shirt`, `Sneaker`, `Trouser` und `Tshirt_top`. In den Verzeichnissen sind die dazugehörigen Bilder als PNG-Datei hinterlegt.

_Data Version Control

In Softwareprojekten gehört die Codeverwaltung mit Werkzeugen wie Subversion und Git zum Alltag. Ein wichtiger Aspekt für die Nachvollziehbarkeit von KI-Projekten ist die Reproduzierbarkeit der Experimente. Dabei spielt neben der Codeverwaltung auch die Daten- und Modellverwaltung eine wichtige Rolle. Für große Datenmengen ist Git nicht geeignet, da es darauf spezialisiert ist, Textdateien zu versionieren. Abhilfe hierfür schafft Data Version Control (DVC) [b], eine quelloffene Kommandozeilenanwendung für die Verwaltung von Daten und Modellen, die sich ähnlich wie Git bedienen lässt.

DVC ist ein Python-Paket, das man mit dem Paketmanager pip installiert und in Kombination mit Git verwendet. Dabei werden zuerst Git und im Anschluss DVC initialisiert. Nach der Initialisierung legt das System im Pendant zum Ordner `.git/` den Ordner `.dvc/` an. In diesem Verzeichnis finden sich alle relevanten Metadaten von DVC.

```
git init
dvc init
dvc add research/data/raw
git add Research/data/.gitignore research/data/raw.dvc
```

```
research
├── data
│   └── raw
│       ├── test
│       └── train
├── model
└── src
```

Verzeichnisstruktur des Projekts (Abb. 2)

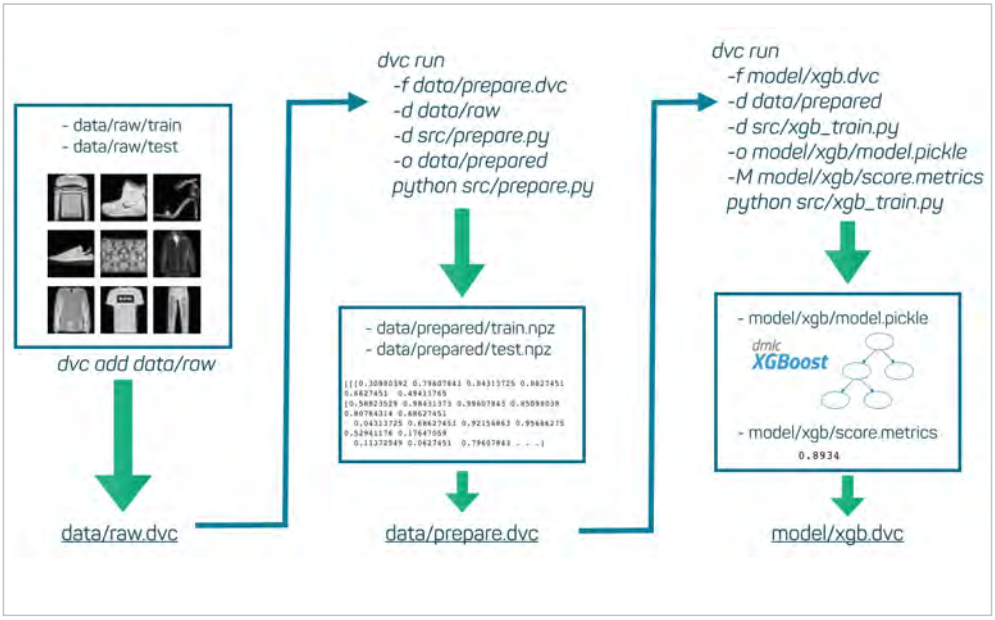
In DVC fügt der Befehl `dvc add` Daten hinzu. Für jeden Ordner und jede Datei berechnet das Tool ein md5-Hash, der im `.dvc/`-Verzeichnis hinterlegt wird.

```
- md5: 570cd340b97f7b4629aa3670a1694a62
wdir: ..
outs:
- md5: f98b70d0adcf1eb0d30a5b95ec8900d7.dir
  path: research/data
  cache: true
  metric: false
  persist: false
```

DVC verwendet den Hash, um Veränderungen in den einzelnen Dateien festzuhalten. Weiterhin generiert DVC eine Gitignore-Datei, die sicherstellt, dass keine Daten aus dem

hinzugefügten Verzeichnis im Git Repository landen. Zu guter Letzt erzeugt dvc add die Datei research/data/raw.dvc. Die DVC-Datei umfasst die relevanten Metainformationen. Im Anschluss übernimmt Git die Versionierung.

Zu diesem Zeitpunkt liegen alle Daten lokal auf dem Rechner. Das kann in einem größeren Projekt die Zusammenarbeit erschweren, da sich Veränderungen am Datensatz nicht zentral erfassen lassen. Weiterhin hat die Verwaltung der Daten auf einem Rechner den Nachteil, dass beim Ausfall des Rechners ein Datenverlust entstehen kann. Um dem vorzubeugen, bietet es sich an, die Daten zentral zu persistieren und dadurch eine Redundanz zu gewährleisten. Ähnlich wie bei Git lassen sich die Daten mit DVC zentral in einem Remote-Speicher hinterlegen. DVC unterstützt viele Speichermöglichkeiten. In diesem Fall fiel die Wahl auf Amazon S3. DVC speichert die Metainformationen über den Remote-Speicher in der Datei `./dvc/config`. Der Befehl push veranlasst das Hochladen der Dateien.



Verzeichnisstruktur des Projekts (Abb. 3)

```
dvc remote add -d myremote s3://XXXXX/fashiondvc
git commit .dvc/config -m „Remotespeicher Konfiguration“
dvc push
```

DVC führt die Experimente mit dem Befehl dvc run aus. Dieser dient als Schnittstelle, um Befehle auszuführen. Über verschiedene Flags werden Ein- und Ausgaben und die generierte DVC-Datei für die Reproduzierbarkeit angegeben (s. Abb. 3):

- -d: Abhängigkeiten
- -o: Ausgaben
- -M: Metriken
- -f: DVC-Datei

Kommt `dvc run` erneut zum Einsatz und bleiben die Abhängigkeiten unverändert, erspart sich DVC durch den Vergleich der DVC-Datei den Rechenaufwand.

Vor dem Training eines Modells ist die Vorbereitung der Rohdaten zwingend notwendig. Dafür speichert das Tool die Bilddaten als NumPy-Array ab und skaliert die Pixel zwischen 0 und 1. Diese Schritte erfolgen in der Prepare-Stage. Dazu legt das System die Datei src/prepare.py an und ruft sie anschließend mit dem folgenden Befehl auf:

Listing 1: Datenvorbereitung, Trainig und Testing

```
from pathlib import Path
import numpy as np
import imageio

mapping = {"Tshirt_top": "0", "Trouser": "1",
"Pullover": "2", "Dress": "3", "Coat“": "4",
"Sandal": "5", "Shirt": "6", "Sneaker": "7",
"Bag": "8", "Ankle_boot": "9"}

def create_images_labels(input: Path):
    labels = []
    images = []
    for image_path in input.glob("*/*.png"):
        parent_name = image_path.parent.name
        label = int(mapping[parent_name])
        image = imageio.imread(image_path)
        images.append(image)
        labels.append(label)
    images = np.array(images)
    labels = np.array(labels)
    scaled = images / 255.0
    reshaped = scaled.reshape(len(scaled), 28, 28)
    return reshaped, labels

data_path = Path("./data")
data_raw_path = data_path.joinpath("raw")
train_input = data_raw_path.joinpath("train")
test_input = data_raw_path.joinpath("test")

train_images, train_labels = create_images_labels(train_input)
test_images, test_labels = create_images_labels(test_input)
output_path = data_path.joinpath("prepared")

np.savez(f"{output_path}/train.npz", images=train_images, labels=train_labels)
np.savez(f"{output_path}/test.npz", images=test_images, labels=test_labels)
```

Listing 2: src/xgb_train.py

```
import numpy as np
from pathlib import Path
import pickle

prepared_train_file = Path("./data/prepared/train.npz")
train_data = np.load(f"{prepared_train_file}")
X_train = train_data["images"]
X_train = X_train.reshape(len(X_train), -1)
y_train = train_data["labels"]

xgbc = XGBClassifier()
xgbc.fit(X_train, y_train)

prepared_test_file = Path("./data/prepared/test.npz")
test_data = np.load(f"{prepared_test_file}")
X_test = test_data["images"]
X_test = X_test.reshape(len(X_test), -1)
y_test = test_data["labels"]

xgb_folder = Path("./model/xgb")
if not xgb_folder.exists():
    xgb_folder.mkdir()

model_out = xgb_folder.joinpath("model.pickle")
with model_out.open("wb") as fd:
    pickle.dump(xgbc, fd)

scored = xgbc.score(X_test, y_test)
score_out = xgb_folder.joinpath("score.metrics")
with score_out.open("w") as fd:
    fd.write(f"{scored:.2f}")
```

```
dvc run -f data/prepare.dvc -d src/prepare.py -d data/raw -o
data/prepared python src/prepare.py
```

`src/prepare.py` führt die Datenvorbereitung durch und erstellt für das Training und Testing die dazugehörigen Numpy-Arrays mit den Labels (s. Listing 1).

Nach der Vorbereitung der Rohdaten lässt sich ein Baseline-Modell mit XGBoost trainieren. Es dient als Benchmark, um weitere Experimente in der Zukunft zu bewerten. Die Trainings- und Testdaten lassen sich aus dem vorherigen Schritt laden, um den Klassifizierer zu trainieren. Um Modelle zu einem späteren Zeitpunkt miteinander vergleichen zu können, speichert das Tool zusätzlich die Genauigkeit als Metrik ab. DVC kann die Metriken zu den Experimenten ebenfalls verwalten, indem Entwickler eine Datei mit dem Flag -M angeben:

```
dvc run -f model/xgb.dvc -d src/xgb_train.py -d data/prepared -o model/xgb/ ↵
                model.pickle -M model/xgb/score.metrics python src/xgb_train.py
```

`src/xgb_train.py` lädt die vorbereiteten Daten, trainiert einen XGB-Klassifizierer, speichert Modell und Ergebnisse nach `model/xgb` (s. Listing 2).

_Open Neural Network Exchange (ONNX)

Nachdem das Training beendet ist, könnte man das Baseline-Modell deployen. Dazu müssen Entwickler eine Applikation bauen, die neue Daten mit dem Modell inferiert. Dazu ruft XGBoost die predict-Methode auf. Darüber hinaus ist die Entwicklung, Versionierung und Verwaltung weiterer Modell-Experimente mit DVC möglich. Vielleicht liefert ein Deep-Learning-Verfahren bessere Ergebnisse als der XGBoost-Ansatz. Bei diesen Experimenten stellen sich unter anderem folgende Fragen:

Teilausschnitt der Frameworks mit ONNX Support (Abb. 4)

- TensorFlow, PyTorch oder MxNet – welches dieser Frameworks sollte man wählen?
- Was passiert, wenn ein besseres Modell gefunden wurde?
- Ist für jedes Modell-Framework die Entwicklung einer neuen Applikation nötig?

Sinnvoll wäre eine Produktionsumgebung, die mit standardisierten Modellen funktioniert. Dadurch wären alle Frameworks mit der Umgebung kompatibel, die den Export des Modells in das standardisierte Format implementieren.

Open Neural Networks Exchange (ONNX) ist eines der Projekte, die sich mit der Standardisierung von Machine-Learning- und Deep-Learning-Modellen beschäftigt. Das Projekt wurde 2017 von Microsoft, Facebook und Amazon ins Leben gerufen. Die Idee hinter ONNX ist, dass man ein Modell zwischen einem Framework A und B problemlos austauschen kann. Abbildung 4 zeigt einen Teilausschnitt des Frameworks. Der Standard besteht aus einem erweiterbaren Graphen, definierten Datentypen und festgelegten Operatoren und Funktionen. Durch eine Import- und Exportfunktion zu bestehenden Frameworks lassen sich die Architektur und Gewichte eines Modells austauschen. Die Kompatibilität zwischen den Frameworks bezeichnet man Fachjargon als Interoperabilität.

Listing 3: ONNX-Konvertierung des XGBoost-Modells

```
import onnxmltools
from onnxmltools.convert.common.data_types import FloatTensorType

initial_type = [(‚feature_input‘, FloatTensorType([1, 784]))]
xgb_onnx = onnxmltools.convert.convert_xgboost(xgb, initial_types=initial_type)
with open("model/xgb/model.onnx", "wb") as file:
    file.write(xgb_onnx.SerializeToString())
```

Listing 4: Klasse ONNXPredictor

```
import numpy as np
class ONNXPredictor:
   def __init__(self, onnx_client, config):
       self.client = onnx_client

   def predict(self, payload):
       model_input = payload["image"]
       model_input = np.array(model_input) / 255.0
       prediction = self.client.predict(model_input)
       return prediction
```

Die Konvertierung des XGBoost-Modells nach ONNX benötigt das Paket `onnxmltools`. Darüber hinaus ist vor der Konvertierung die Definition des Eingangsvektors für das Modell notwendig. `src/xgb_train.py` wird um die folgenden Zeilen ergänzt, um das ONNX-Modell ebenfalls abzulegen. Zusätzlich wird der dvc run-Befehl mit dem Flag `-o model/xgb/model.onnx` erweitert:

```
dvc run -f model/xgb.dvc -d src/xgb_train.py -d data/prepared -o model/xgb/ ↵
              model.pickle -o model/xgb/model.onnx -M model/xgb/score.metrics ↵
                                                    python src/xgb_train.py
```

Listing 3 bildet die Erweiterung von src/xgb_train.py um die ONNX-Konvertierung des XGBoost-Modells ab.

Ein weiteres Projekt, das der Standard entwickelt hat, ist die ONNX Runtime, eine Laufzeit für die Inferenz (s. Abb. 5). Mit dieser Technologie lassen sich ONNX-Modelle innerhalb einer Umgebung ausführen. Dadurch ist es möglich, die Modelle nach dem Training ohne große Anpassungen direkt in den produktiven Betrieb mit der gleichen Komponente zu übernehmen. Nachdem `onnxruntime` über pip installiert wurde, lässt sich die Inferenz ausführen.

```
import onnxruntime as rt
import numpy as np

session = rt.InferenceSession(""model/xgb/model.onnx")
input_name = session.get_inputs()[0].name
session.run([], {input_name: np.array(np.random.rand(1, 28*28), dtype=np.float32)})
```

Dazu wird zuerst das Modell geladen und im Anschluss mithilfe der Metadaten der Eingangsvektor definiert. Für den Input ist es wichtig, dass die Daten als NumPy-Array mit dem Datentyp Float32 vorliegen.

_Das Werkzeug Cortex

Nachdem das Daten- und Modellmanagement, die Standardisierung und die Inferenz der Modelle behandelt wurde, steht als Nächstes die Inbetriebnahme des Modells an. Die Inferenz des Modells soll über eine REST-API erreichbar sein. Dazu eignet sich der Einsatz des Cloud-Infrastruktur-Werkzeug Cortex, das Modelle skalierbar auf AWS deployt. Das Projekt unterstützt ONNX-Modelle und führt diese mit der ONNX Runtime aus.

Cortex ist ein ML-Deployment-Werkzeug für Inferenz. Mithilfe der Open-Source-Plattform können Entwickler Modelle als REST-API auf einem Kubernetes-Cluster bereitstellen. Derzeit unterstützt Cortex hierfür nur die Services von AWS. In der Zukunft soll Cortex aber auch für andere Kubernetes-Provider verfügbar sein [c]. Für das Deployment verwendet das Tool die Amazon Services S3 und Elastic Kubernetes Service (EKS). S3 ist ein Objektspeicher, auf dem sich die trainierten Modelle ablegen lassen. EKS ist ein vollständig verwalteter Kubernetes-Service.

Entwickler können das Toolkit über die Kommandozeile installieren [d]. Es benötigt Zugriff auf eine AWS-Umgebung für das Erstellen der Ressourcen. Dafür sind die AWS Credentials eines Benutzers notwendig. Um die Komplexität beim Erstellen des Benutzers nicht zu sprengen, erhält er in diesem Beispiel die AdministratorAccess-Policy. An dieser Stelle sei erwähnt, dass in einem produktiven Umfeld davon abgeraten wird, einem Account mehr Berechtigungen zu erteilen als notwendig sind. Die minimalen Berechtigungen können Interessierte hier nachlesen [e].

Cortex installieren

Die Installation erfolgt anschließend mit dem Befehl `cortex cluster up`. Sie kann bis zu 15 Minuten dauern, da währenddessen verschiedene Ressourcen wie das EKS Cluster, EC2-Instanzen, die zugehörigen Netzwerkeinstellungen und der S3 Bucket angelegt werden. Bevor das System die Installation bestätigt, kommt es zu einer Auflistung aller Kosten, die Cortex erzeugt (s. Tabelle).

Nachdem die Installation abgeschlossen ist, bedarf es der Durchführung einiger Aufgaben. Zunächst muss man das ONNX-Modell auf dem gewählten S3-Bucket ablegen. Dabei lässt sich das Modell entweder über die AWS-Konsole oder direkt über die Kommandozeile speichern. In einem weiteren Schritt wird die Klasse ONNXPredictor erstellt (s. Listing 4).

Sie erhält über den Konstruktor den `ONNXClient` [f]. Dieser verwaltet die ONNX Runtime und enthält den Boilerplate-Code für die Datentransformationen und für das Laden des Modells. Die Klasse wird beim Hochfahren des Pods ein-

Training mit XGBoost, Standardisierung mit ONNX und Inferenz mit der ONNX Runtime (Abb. 5)

mal geladen – ein Pod ist eine deploybare Einheit in Kubernetes. In diesem Fall ist es eine laufende Instanz des Modells. Ein Request reicht den Payload an die `predict`-Methode weiter. Sie implementiert die Logik für die Entscheidungsfindung. Die Datei cortex.yaml hält die Konfigurationen fest und listet alle Modell-Instanzen auf, die auf dem Cluster eingerichtet werden sollen. Die Unterscheidung erfolgt über den Namen. Die Datei verbindet das Skript mit dem Modell-Pfad auf S3 und erstellt dadurch ein Deployment.

```
- name: fashion-classifier
  predictor:
    type: onnx
    path: predictor.py
    model: s3://cortex-XXXX/model.onnx
  compute:
    cpu: 1
```

Außerdem können Entwickler weitere Konfigurationen wie die Anzahl an Instanzen festlegen. Der Befehl `cortex deploy` legt das Modell auf dem Cluster an. Details über die Konfiguration des Deployments, den Status und die Endpoint-Url werden mit dem von dem Befehl `cortex get fashion-classifier` eingesehen. Weiterhin können Entwickler die Log-Nachrichten der API mit `cortex logs fashion-classifier` abrufen. Der Befehl `get` macht die Adresse des Endpoints ausfindig. Daraufhin lässt sich das Deployment mit dem Request testen:

```
import requests
import imageio
image = imageio.imread("coat_image.png")
url = "http://XXX.eu-west-1.elb.amazonaws.com/fashion-classifier"
r = requests.post(url, json={"image": image.reshape(-1).tolist()})
```

Über `cortex cluster` down lässt sich das Tool wieder de-

installieren. Neben dem schnellen Deployment hat Cortex weitere hilfreiche Features für den operativen Betrieb im Gepäck, die zum größten Teil durch den Einsatz von Kubernetes hinzukommen. Dazu gehören unter anderem Rolling Updates, Autoscaling und Monitoring der Applikationen. Das Monitoring sammelt neben Metriken zum Netzwerkverkehr die über die Verteilung der Vorhersagungen. Diese können darüber aufklären, wie das Modell im Live-Betrieb performt. An dieser Stelle sei erwähnt, dass sich der gewählte Ansatz mit Cortex aus Sicht der Betriebskosten nur eignet, wenn die Applikation konstant aufgerufen wird. Das Kubernetes-Cluster und die beteiligten EC2-Instanzen verursachen laufende Kosten, die erst mit ausreichend hoher Grundlast wirtschaftlich rentabel sind. Der Einsatz von Cortex lohnt sich, sobald genügend viele Inferenzanfragen auftreten, um die operativen Kosten zu rechtfertigen.

Fazit

Die drei Werkzeuge DVC, ONNX und Cortex konzentrieren sich auf die Operationalisierung von KI-Projekten. Dieser Stack bietet eine einfache Möglichkeit, um ML-Modelle in Produktion zu bringen. Bei klassischen Softwareprojekten ist die Codeverwaltung das A und O, während bei KI-Projekten die Versionierung von Daten und Modellen genauso wichtig ist. DVC ist ein praktisches Werkzeug für die Daten- und Modellverwaltung. Es ermöglicht die Reproduzierbarkeit von Experimenten und hilft beim schmerzfreien Datenaustausch. Dadurch kann man innerhalb eines Projekts besser kollaborieren. Weiterhin lassen sich verschiedene Modell-Ansätze und Experimente individueller erproben. ONNX und Cortex unterstützen die Inbetriebnahme der Modelle. Durch die ONNX Runtime können Entwickler kompatible Modelle innerhalb einer Produktionsumgebung problemlos deployen. Data Scientists haben dank der Standardisierung der Modelle die Möglichkeit, ihr präferiertes Deep-Learning-Framework zu verwenden, sofern dieses den Export nach ONNX unterstützt.

Cortex hilft beim einfachen Deployment der Modelle und verkürzt damit die Zeit von der Modellentwicklung bis zur Produktion. Data Scientists müssen sich dadurch weniger Gedanken um Infrastruktur oder Operations machen. Durch das Zusammenspiel von DVC, ONNX und Cortex gelangen Modelle auf einem sehr schnellen Weg in die Produktion und lassen sich reproduzieren.

Kosten durch Cortex

AWS Ressourcen	Kosten pro Stunde
EKS Cluster	$0,10
20 GByte EBS volume for the operator	$0,003
T3.Medium instance for the operator	$0,0456
NAT Gateway	$0,048
2 Elastic Load Balancers	$0,056
50 GByte EBS Volume for APIs	$0,008
M5.Large instance for APIs	$0,107

Onlinequellen

[a] Fashion MNIST von Zalando: github.com/zalandoresearch/fashion-mnist#get-the-data
[b] Data Version Control (DVC): dvc.org
[c] Kubernetes und Cortex: www.cortex.dev/cluster-management/aws-credentials
[d] Installation: www.cortex.dev/install

Nico Axtmann

entwickelt als Machine Learning Engineer datengetriebene Produkte und Lösungen. Derzeit konzentriert er sich vor allem auf die Inbetriebnahme von Modellen in produktiven Umgebungen.

>Miniaturwelt

Alexandra Waldherr

Quantencomputer beschleunigen die Analyse komplexer Systeme. Sie eignen sich damit für Kryptografie im Bereich der Telekommunikation ebenso wie für das Berechnen und Simulieren von Quantensystemen.

Quantencomputer bieten der Forschung eine Möglichkeit, Quantensysteme zu berechnen und zu simulieren. Für das Verständnis von Quantum Computing ist es zunächst wichtig, den Quantencomputer als radikal andere Hardware und als Backend anzuerkennen. Die Branche arbeitet immer noch mit Gates, allerdings materialisieren sich hier sogenannte Pauli-Gates physisch nicht als Flip von 0 und 1 in der CPU, sondern beispielsweise als Laser- oder Mikrowellenimpulse, die Energieniveaus eingefangener Elementarteilchen (Qubits) rotieren können.

Damit stellen Quantencomputer ein derzeit noch fragiles Experiment dar. Es geht um Transformationen, Kalkulationen und Umformungen sowie um das Nachstellen und Verstehen molekularer Prozesse, aber nicht um das langfristige Speichern von Information. Auch können Qubits nicht kopiert oder eingefügt werden.

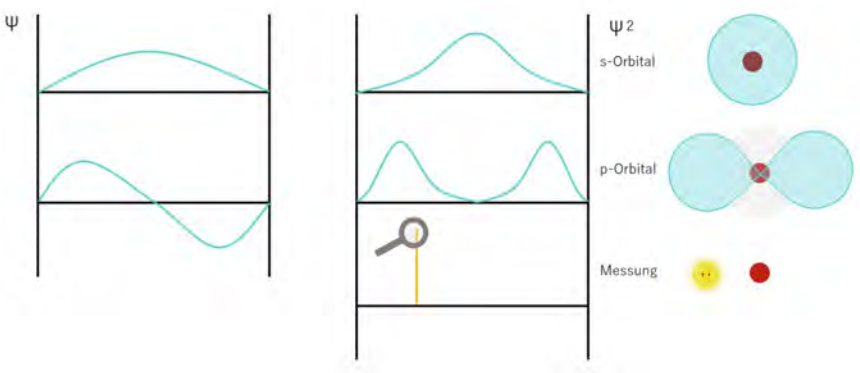
Quantencomputer bieten kein Copy-and-Paste

Bevor der Artikel den Qubits anhand eines anschauliches Beispiels ein Gesicht geben wird, lohnt es sich zu verstehen, warum Qubits so viel bessere Kalkulationen ermöglichen und wie das quantenmechanische Auswertungsverfahren dazu eigentlich funktioniert.

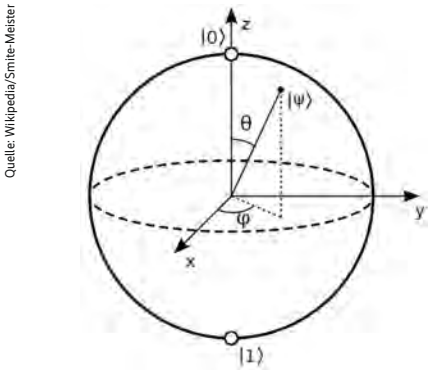
Grob gesagt gibt es ab einer gewissen Größe ein Limit, wie weit sich Energieniveaus oder andere physikalische Zustände voneinander unterscheiden können. Daher kommt auch das Wort „Quanten“, was so viel heißt wie „quantisierte Teilchen“. Das Kernproblem in dieser Miniaturwelt tritt dann zutage, wenn mehr Energie als für den Grundzustand notwendig vorhanden ist, aber zu wenig, um das nächste Level zu erreichen. Hier handelt es sich um quantisierte Niveaus; bildlich gesprochen kann man zwischen Erdgeschoss und erstem Stock nicht einfach aus einem Aufzug aussteigen.

Ein Mensch kann sich natürlich real in einem Aufzug zwischen Stockwerken befinden, einem quantisierten Teilchen ist das verboten. In der Forschung wird es an einer der zwei Stellen auftauchen müssen. Haben Teilchen beispielsweise 20 Prozent mehr Energie, befinden sich 80 von 100 weiterhin im Grundzustand. Hat ein Teilchen aber 80 Prozent mehr Energie, wird ein Beobachter es zu 80 Prozent im metaphorischen ersten Stock wiederfinden.

Insgesamt fasst die Wissenschaft alle möglichen Positionen in einer Wellenfunktion Ψ zusammen. Aus der Physik ist bekannt, dass sich Wellenfunktionen nicht real messen lassen. Um die real messbare Wahrscheinlichkeit zu jeder dieser Positionen zu erhalten, müssen Anwender diese Funktion daher zunächst quadrieren. Als anschauliches Beispiel betrachte man Elektronen in einem Atom: Aus der Schrödinger-Gleichung erhalten Anwender pro Energieniveau eine Funktion und jede entspricht quadriert als Ψ^2 – oder bildlicher als Orbitale – den wahrscheinlichen Aufenthaltsorten von Elektronen (Abb. 1). Das Elektron ist überall zu einer gewissen Wahrscheinlichkeit. Das nennt sich Superposition und birgt die extreme Macht der Quantencomputer, viele Zustände parallel abzutasten. Je größer $\Psi(x)^2$, desto wahrscheinlicher ist ein Messergebnis an Position x. Allerdings kollabiert mit der Messung sofort die Wellenfunktion und als Qubit müsste solch ein Teilchen erneut in die richtige Superposition gebracht werden. Und genau hier versteckt sich das momentane Kernproblem im Quantencomputing, nämlich die Error Rate und die Dekohärenzzeit verfügbarer Quantencomputer. Wenn der Quantencomputer nicht die endgültige



Die Abbildung zeigt die niedrigsten Energiezustände pro Schale in einem Atom. Das Quadrat der Wellenfunktion beschreibt den Orbitalbereich. Bei einer Messung fällt diese komplett zusammen, das Elektron ist lokalisiert (Abb. 1).



Quelle: Wikipedia/Smite-Meister

Die abstrakte Manipulation des Zustandes der Bloch-Sphäre drückt man über θ und φ aus. Manche Rotationen sind besonders wichtig (Abb. 2).

Messung durchführt, sondern eine Umweltstörung zuvorkommt, stimmen die erhaltenen Wahrscheinlichkeiten nicht mehr und das System liefert einen Fehler. Derzeit liegen die Dekohärenzzeiten zum Einsatz kommender Superconducting-Schaltkreise bei einigen Nano- bis Mikrosekunden und die Systeme bräuchten mindestens 100 physische Elementarteilchen, um einen ausreichend verlässlichen Qubit darstellen zu können.

Bloch-Sphäre: Visualisierung hilft zu verstehen

Die Frage ist, wie man nun ein Qubit in Superposition bringt. Dazu müssen Anwender einen Quantum Circuit mit diversen Gates erstellen, die den Quantenzustand (entspricht der Wellenfunktion Ψ) verändern. Abstrakt lässt sich das mithilfe der Bloch-Sphäre beschreiben. Auf dieser Kugel würden Nord- und Südpol den klassischen Bits 0 und 1 entsprechen, nur dass das Qubit hier alle möglichen Zustände dazwischen bis zur Messung ebenso abdecken kann. Zwei wichtige Konzepte lassen sich mithilfe der Sphäre illustrieren (Abb. 2):

- Die Position auf der Bloch-Sphäre gibt den Quantenzustand an und wird als Zustandsvektor bezeichnet. Die zwei Komponenten a, b sind komplexe Zahlen, wobei a^2 die Wahrscheinlichkeit angibt, Zustand |0> zu erhalten, und b^2 die Wahrscheinlichkeit, Zustand |1> zu erhalten. Visuell über die Bloch-Kugel: Je näher am Nordpol, desto eher |0> und am Äquator 50 : 50 für |0> : |1>.

$$\Psi = a * |0\rangle + b * |1\rangle$$

- Mathematisch lassen sich die Komponenten in Matrixschreibweise angeben. Die Gates müssen diese Matrix manipulieren, um auf eine Veränderung des Zustandsvektors hinzuwirken. Folglich lassen sich die Gates mathematisch ebenso in Matrizenform ausdrücken, wobei die Argumente die Winkel beschreiben, um die man den Zustandsvektor rotieren möchte. Der grobe Ablauf für jedes Programm mit Quantencomputern ist simpel:

1. Qubits und klassische Bits initialisieren (Quantum and Classical Register);
2. mit Operationen in Form der üblichen Quantum Gates einen Quantum Circuit bauen;
3. den Quantum Circuit an einen Simulator oder ein echtes Quantum-Backend schicken;
4. Ergebnisse analysieren.

Selbe Prinzipien in der Softwareentwicklung

Die genannten Programmiersprachen verwenden die gleichen Gates und ähnliche Bezeichnungen, da auch die Softwareentwicklung auf denselben quantenmechanischen und mathematischen Modellen aufbaut. Stereotypisch ist sicherlich das Hadamard-Gate, das ein Qubit in Superposition bringt. Mithilfe des CX-Gates lassen sich Qubits verschränken. Anwender können allein mit diesen zwei Gates bereits Bell-Paare „bauen“, was für eine erste Auswertung von Messdaten wichtig ist. Die fundamentalen Gates seien hier kurz aufgelistet (weitere siehe Tabelle). Leser, die einen tieferen Einblick in Circuits suchen, sollten dazu einen Blick in den Quantum Algorithm Zoo von Stephen Jordan werfen, der als Forscher in Microsofts Quantum Systems Group beschäftigt ist.

Unitary-Gates: Das elementare Einheitsgatter ist das U3-Gatter, das Anwendern alle Rotationen in alle Raumrich-

Praxisbeispiel – erste Schritte auf IBMs Quantencomputer

Ein gefürchteter Anwendungsfall für Quantum Computing ist der Shor-Algorithmus, der die weitverbreitete RSA-Kryptografie per Primfaktorenzerlegung einfach knacken könnte. Weitere Anwendungsfälle liegen unter anderem im Bereich der sogenannten Postquantenkryptografie. Hier kann man beispielsweise die fragile Superposition von Qubits nutzen, um einen Lauscher in der Leitung zu identifizieren; angelehnt an das BB84-Protokoll.

Um das Beispiel praktisch selbst nachzustellen, empfiehlt es sich, sich zunächst auf der Quantum-Experience-Seite von IBM zu registrieren. Das System stellt alles Nötige für die Entwicklung und Simulation entsprechender Experimente bereit. Rechts am Bildschirm stehen alle verfügbaren Backends gelistet: Melbourne ist der leistungsstärkste frei verfügbare Quantencomputer mit derzeit 15 Qubits.

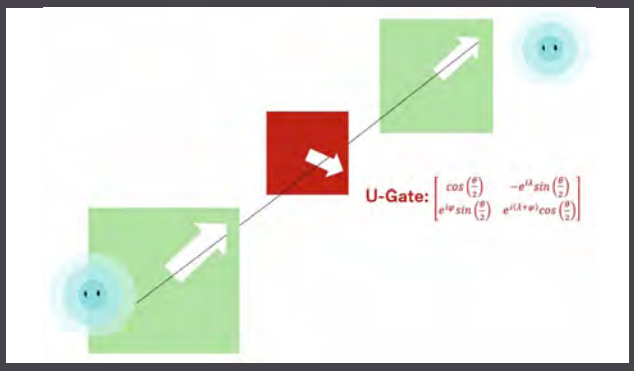
Unter „My Account" befindet sich ein Anmeldetoken. Man kann innerhalb der Plattform arbeiten, im hier gezeigten Beispiel kommt Qiskit per Installation über `pip install qiskit` zum Zug (Listing 1).

```
from qiskit import *
from qiskit.tools.visualization import plot_histogram
from qiskit.tools.visualization import plot_bloch_multivector

IBMQ.save_account('#longtokenhere')
IBMQ.load_account()
```

Listing 1: Verbindung zu Qiskit und QExperience

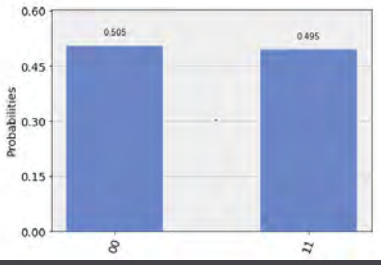
```
#circuit = QuantumCircuit(Nr. Qubits, Nr. klassische Bits)
circuit = QuantumCircuit(2, 2)


#Bell-Pair Entanglement
circuit.h(0)          #Superposition (control qubit)
circuit.cx(0,1)       #Verschränkung der zwei Qubits


#Barriere ("Übertragung")
circuit.barrier()
#circuit.u3(20, 50, 70, 1)  #-> Eve
circuit.barrier()


circuit.measure(0,0) #Alice
circuit.measure(1,1) #Bob

circuit.draw()
```

Listing 2: Code eines vereinfachten Algorithmus zum Versenden verschränkter Teilchen



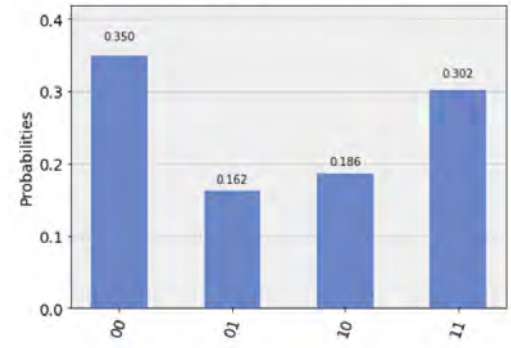
Die Bezugsbasis beim Erstellen und Messen eines Systems sollte übereinstimmen oder zumindest bekannt sein, da sonst die Polung der Bloch-Sphären nicht gleich ist (Abb. 3).

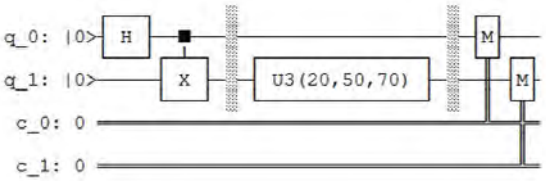
```
#refresh account and load provider
IBMQ.load_account()
provider = IBMQ.get_provider('ibm-q')

#use least occupied quantum computer
qucomp = provider.get_backend('ibmq_london')
results = execute(circuit, backend=qucomp, shots = 500).result()

plot_histogram(results.get_counts(circuit))
```



Die Ergebnisse sind nun |00> und |11> im Verhältnis 0,5 : 0,5. In den meisten Runs ist eine Störung der Hardware derzeit nicht unüblich; bei unterschiedlichen Ergebnissen mit |01> oder |10> liegt das Problem bei der äußerst großen Empfindlichkeit der Quantenhardware auf Umwelteinflüsse (Abb. 5).

```
#refresh account and load provider
IBMQ.load_account()
provider = IBMQ.get_provider('ibm-q')

#use least occupied quantum computer
qucomp = provider.get_backend('ibmq_london')
results = execute(circuit, backend=qucomp, shots = 500).result()

plot_histogram(results.get_counts(circuit))
```





Das „lauschende“ U3-Gate bewirkt eine totale Verfälschung, und sobald Alice und Bob auch nur einen Teil der Ergebnisse vergleichen, merken sie, dass unerwartet viele |01>- und |10>-Zustände vorhanden sind und die übertragenen Daten nicht genutzt werden sollten (Abb. 6).

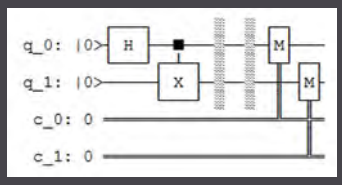
1. Circuit bauen und Gates einfügen: Das Experiment benötigt zunächst einen Quantum Circuit mit zwei Qubits und zwei klassischen Bits. Letztere sind essenziell, um die finalen Informationen verlässlich zu speichern. Ein Hadamard-Gate bringt dann das erste Qubit in Superposition zwischen |0> und |1>. Anschließend sind beide Qubits parallel zu verschränken (das heißt, beide sind im selben Zustand), um sie in Superposition von |00> und |11> zu bringen. Mittels Barrieren muss sichergestellt sein, dass die Qubits von der Initialisierung (Senden) zur Messung (Empfangen) abgetrennt sind (Listing 2).

2. Schematische Darstellung auswerfen: Mit dem Befehl `cirquit.draw()` lässt sich der programmierte Circuit als Ablaufschema ausgeben (Abb. 4).

3. Circuit an den Simulator oder Quantencomputer schicken: Hier können Anwender*innen jetzt auf einen von IBMs Quantencomputern zugreifen und den Algorithmus dort 500 Mal laufen lassen, um anschließend die Wahrscheinlichkeit mittels Histogramm auswerten zu können. Es liegen nur die Zustände |00> und |11> vor (Abb. 5).

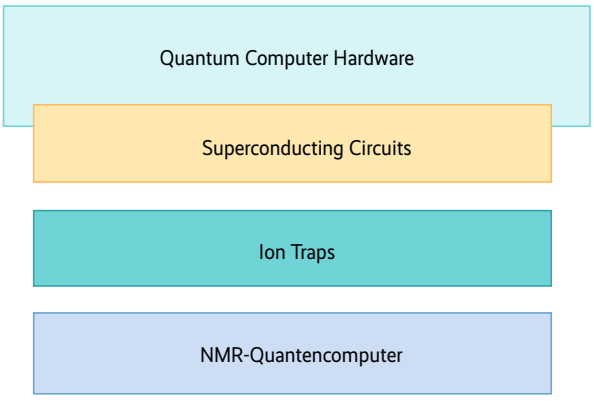
Die Ergebnisse sind nun |00> und |11> im Verhältnis 0,5 : 0,5. In den meisten Runs ist eine Störung der Hardware derzeit nicht unüblich; bei unterschiedlichen Ergebnissen mit |01> oder |10> liegt das Problem bei der äußerst großen Empfindlichkeit der Quantenhardware auf Umwelteinflüsse (Abb. 5).4. Spion (Eve) einschalten: Jetzt kommt der Kniff bei Quantensystemen: Solange nicht bekannt ist, in welcher Orthonormalbasis gemessen wird (vereinfacht gesagt ist die Ausrichtung beider Pole der Bloch-Sphäre nicht bekannt), können Dritte

Das Circuit-Schema repräsentiert den ersten, unbeeinflussten Kommunikationsweg (Abb. 4).



diese nur willkürlich aussuchen. Somit ist bei der Messung mit einem „neuen“ Nord- bzw. Südpol die Entfernung des Zustandsvektors zu |0> bzw. |1> komplett unterschiedlich, wodurch die Superposition falsch kollabiert. Die erhaltenen Qubits sind nicht mehr gleich und verhalten sich in verschiedenen Bezugsbasen bei Messungen von Alice und Bob unterschiedlich. Um diesen Fall nachzustellen, verschiebt man den Beginn des Kommentars zwischen den zwei `cirquit.barrier()` nur noch vor `-> Eve.cirquit.u3(20,50,70,1)` aktiviert einen Lauscher als Zwischensubstanz mit komplett randomisierter Messbasis. Die ersten drei Zahlenwerte lassen sich willkürlich ändern, das vierte Argument gibt an, dass es sich um den zweiten Qubit handelt und im übertragenen Sinne somit die Leitung zu Bob abgehört wird.

Das U3-Gate rotiert zwar das Qubit, aber es läuft auf dasselbe hinaus, als wäre die Bezugsbasis des Messapparates falsch eingestellt. Wer jetzt den Circuit noch mal von Anfang an neu baut und an das gewählte Backend schickt, kann eine drastische Änderung erkennen (Abb. 6). Achtung: Alle Befehle inklusive der Circuit-Definition anfangs sind noch einmal auszuführen! Setzt man den Circuit nicht auf 0, würden die neuen Gates einfach hinten an den bereits bestehenden Circuit angehängt.



Derzeit verfolgt die Forschung drei Ansätze für die Entwicklung von Quantencomputern: Miniaturschwingkreise, die ebenso zwei Energieniveaus einnehmen können, sind industriell am meisten gefragt. Ion Traps manipulieren einzelne Atome, sind aber großteils noch in Entwicklung. NMR-Quantencomputer waren die ersten Modelle, skalieren derzeit aber schlechter als andere Systeme (Abb. 7).

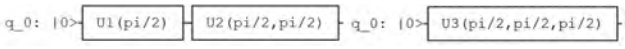
tungen und um jeden Winkel ermöglicht. Davon werden viele vereinfachte Gates für spezielle Fälle abgeleitet, was die Bestimmung notwendiger Parameter erleichtert.

Matrix-U3-Gate:

$$\begin{bmatrix} cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi}sin\left(\frac{\theta}{2}\right) & e^{i(\lambda+\varphi)}cos\left(\frac{\theta}{2}\right) \end{bmatrix}$$

Unitary-Gates:

```
circuit.u3(theta, phi, lambda, qubit)
circuit.u2(theta, phi, qubit)
circuit.u1(lambda, qubit)
```



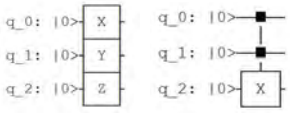
Pauli-Gates: Es gibt drei Pauli-Gates für alle Raumachsen: Sie tauschen die Zustände der x-, y- oder z-Achse, zum Beispiel zwischen |0> und |1> (X-Gate), was dem typischen NOT-Gate entspricht. Daher ist das CX-Gate (controlled NOT) beim Quantum Computing analog zu CNOT. Das CCX-Gate (controlled-controlled-NOT) ist als Toffoli-Gate bekannt.

Pauli-Gates:

```
circuit.x(qubit)
circuit.y(qubit)
circuit.z(qubit)
```

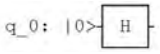
CX-Gate: `circuit.cx(ctr qubit, target qubit)`

Toffoli-Gate: `circuit.ccx(ctr1, ctr2, target qubit)`



Hadamard-Gate: Das Hadamard-Gate rotiert um die x- und z-Achse und setzt das Qubit von der Anfangsposition |0> auf den „Äquator" zu Position |+>. Damit könnte Anwender eine 50:50-Superposition zwischen |0> und |1> einstellen. Um sich alle Gates vorzustellen, bietet sich in Qiskit der `statevector_simulator` an, der die Ergebnisse gut vergleichbar auf der Bloch-Sphäre visualisiert.

Hadamard-Gate: `circuit.h(qubit)`



Je nach Anwendungsgebiet können die umgesetzten Systeme unterschiedliche Algorithmen lösen. Es gibt Ansätze zu topologischen und universellen Quantencomputern (vergleiche Abbildung 7). Wenn von Systemen mit über 1000 Qubits gesprochen wird, wie sie derzeit die Firma D-Wave produziert, handelt es sich um adiabatische „Quantencomputer", die spezifisch für spezielle Optimierungsprobleme gebaut wurden. Es stellt sich nicht die Frage, ob Quantencomputer viel leisten können, sondern wann die Technologie stabil genug ist, die Systeme groß genug sind und wann es genug Softwareentwickler dafür gibt. Der enorme Vorteil von Quantencomputern ist die vereinfachte und beschleunigte Analyse komplexer Systeme. Daher gibt es viele Überlegungen zu Anwendungen für Finanzgeschäfte oder gekoppelt mit dem Training und der Optimierung von künstlicher Intelligenz. Adiabatische Quantencomputer sowie der VQE-Algorithmus (Variational Quantum Eigensolver) erzielen bereits große Fortschritte.

Mehr noch als nur für Endkonsumenten steigt auch die Relevanz der Technologie für das Sicherheitsbedürfnis innerhalb von Unternehmen. Je größer und verlässlicher die Quantensysteme werden, desto eher besteht die Chance, mit dem Shor-Algorithmus die derzeitige RSA-Kryptografie auf Eis zu legen. Ebenso spannend ist die Entwicklung im Bereich der Telekommunikation. Micius, ein Satellit für ein Pilotprojekt zur Quantenkryptografie, schickt bereits kontinuierlich über unseren Köpfen verschränkte Photonen zwischen Österreich und China hin und her. Das Ziel ist ein Quantenschlüsselaustausch (QKD – Quantum Key Distribution), der in komplexerer Weise mit dem BB84-Protokoll das oben demonstrierte Schema praktisch zwischen Weltall und Erde umsetzt. Dadurch können gemeinsame geheime Schlüssel unglaublich sicher ausgetauscht

EDDI WILL CHANGE IT.

Wir sind der zentrale IT-Experte des EDEKA-Verbunds. Als Partner und Visionär der Digitalisierung gehen wir voran, um die Transformation im Verbund mitzugestalten und proaktiv weiterzuentwickeln. Aus dem laufenden Betrieb heraus oder durch Inspiration von außen: Wir machen möglich, was unmöglich schien und beeinflussen so maßgeblich die Zukunft für den Einzel- und Großhandel sowie für die EDEKA-Zentrale und ihre Tochtergesellschaften. **EDDI, THAT'S IT.**

WIR SUCHEN DICH: SITE RELIABILITY ENGINEER (M/W/D)

Als SRE bist du in gleich zwei Welten zu Hause: Entwicklung und Administration. Du bewegst dich schnell zwischen beiden hin und her und sorgst so dafür, dass optimal zusammengearbeitet wird. Dazu kümmerst du dich um Design, Entwicklung, Test und Betrieb von verteilten Anwendungen – und gestaltest so mit uns die IT für den Handel von morgen!

DEINE AUFGABEN BEI EDDI

- Du wirkst mit an der Integration von verteilten Anwendungen in die Webauftritte der EDEKA und unser Enterprise Content Management System.
- Du arbeitest aktiv am Aufbau unserer Plattform auf Azure.
- Du bringst neue Technologien und Frameworks aus dem Cloud-, Web-, API- und Java-OpenSource-Umfeld ein.
- Du etablierst DevSecOps-Maßnahmen zur Absicherung unserer CI-/CD-Strecke und der Cloud-Landschaft.
- Du erstellst Datadog-Dashboards und -Alerts für den Blick hinter die Kulisse vielfältiger Microservices.
- Du automatisierst alle Tätigkeiten im Infrastrukturumfeld wie z. B. Kubernetes, Datenbanken und Suchmaschinen.

WAS EDDI DIR BIETET

- Arbeit in einem eigenverantwortlichen und selbstorganisierten Team
- Stetige Entwicklungs- und Weiterbildungsmöglichkeiten im laufenden Projekt
- Möglichkeit zur Teilnahme an Entwickler-Konferenzen
- Moderne Arbeitsbedingungen und einen sicheren Arbeitsplatz
- Flexible Arbeitszeiten
- 30 Tage Jahresurlaub
- Familienservice und Fitness-Studio
- Vermögenswirksame Leistungen sowie Zuschuss zu Kantine und ÖPNV

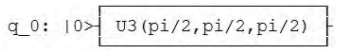
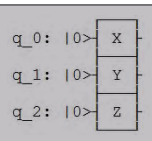
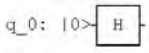
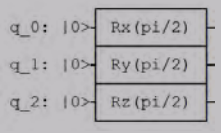
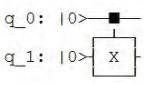
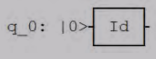
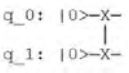
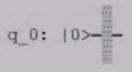
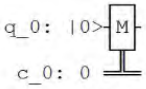
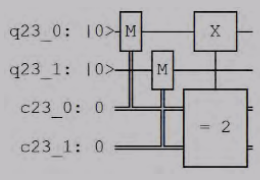
Entwickle mit uns die digitale Zukunft des EDEKA-Verbunds:

JETZT CODE SCANNEN UND DIREKT BEWERBEN!

Mehr Infos auf digital.edeka/karriere.

EDEKA EDDI

Die wichtigsten Quantengatter (Quantum Gates), deren Befehl in Qiskit inklusive Darstellungsweise im Circuit

Operation	Circuit	Zweck
U3-Unitary-Gate	circuit.u3(theta,phi,lambda,qubit) 	Manipulation des Zustandsvektors in allen Raumrichtungen
Pauli-Gates	circuit.x(qubit) circuit.y(qubit) circuit.z(qubit) 	tauscht Zustände durch Rotation um x-, y- oder z-Achse (Bloch-Kugel)
Hadamard-Gate	circuit.h(qubit) 	50:50-Superposition, Anfangszustand: Antizustand
Rotationsgatter	circuit.rx(angle, qubit) circuit.ry(angle, qubit) circuit.rz(angle, qubit) 	50:50-Superposition, Anfangszustand: Antizustand
CNOT-Gate	circuit.cx(ctr,qubit) 	Pauli-X-Gate, wenn der Kontroll-Qubit sich in Zustand |1> befindet → wird verschränkt! (nicht nur CX, auch CH)
Identity-Gate	circuit.iden(qubit) 	entspricht Multiplikation mit der Identitätsmatrix, oftmals Kontroll-Gate (Wert bleibt gleich)
Swap-Gate	circuit.swap(qubit1,qubit2) 	tauscht die Zustandsvektoren zweier Qubits
Barriere	circuit.barrier() 	verhindert explizit, dass benachbarte Gates gemeinsam ausgeführt werden
Measurement	circuit.cx(qubit, bit) 	misst den Zustand des Qubits und speichert das Ergebnis
c_if-Operation	circuit.gate(qubit).c_if(creg,value) 	Operation, wenn die klassischen Bits zusammen == Wert ergeben

werden, was speziell eben für Telekommunikation zentraler Bestandteil des Geschäftsfeldes ist. Bisher wurden für die Forschung unter anderem Bilder von Erwin Schrödinger sicher ver- und entschlüsselt. Gleichzeitig eröffnen Quantencomputer der Forschung eine Möglichkeit, Quantensysteme zu berechnen und exakt zu simulieren. Das verspricht Antworten auf Fragen wie: Welche Moleküle können sich im Körper an Enzyme binden oder Krankheiten heilen? Wie sind wichtige Proteine in unseren Zellen gefaltet? Lassen sich Datenbanken noch schneller durchsuchen? Welche Konfiguration sollen Materialien besitzen, die die Menschheit auf den Mars bringen? Welche Ansätze können Ökosysteme stabilisieren?

Je besser die Forschung die Systeme versteht, desto innovativere Quantencomputer können entstehen (vergleiche Abbildung 4). Schon heute zeigt sich, wie interdisziplinär die Quantenmechanik eigentlich ist. Quantencomputer mögen auf den ersten Blick abstrakt und problematisch klingen, bieten aber fast auf jedem technischen, wirtschaftlichen und naturwissenschaftlichen Feld einen Anwendungsfall.

Quellen

Weiterführende Information zum Quantum Development Kit und Circuits: ix.de/zawx

Alexandra Waldherr

hat bereits am CERN, CeMM und Freudenberg Sealing Technologies mitgewirkt. Ihre Faszination gilt der Physik und Chemie des Kleinen; ihre Diplomarbeit hat sie der umweltfreundlichen Herstellung neuartiger fluoreszierender Pigmente gewidmet.

Developer-Recruiting war noch nie so einfach!

Bekommen Sie Zugang zu Europas größter Developer-Communtiy – WeAreDevelopers ist der einfachste Weg ihren nächsten Software Developer, DevOps Engineer and Engineering Leader einzustellen.

In nur 5 Minuten registrieren & loslegen:
wearedevelopers.com/business

1,2mio+
Besucher*innen pro Jahr

30k+
registrierte Entwickler*innen auf Jobsuche

<10
Tage durchschnittlich bis zur ersten Bewerbung

>WARUM DER NEUSTART

Thomas Limbüchler

Graue Büros, trister Alltag und steile Hierarchien – das war einmal. Heute haben Developer bei der Jobsuche die Wahl.

Nicht lange ist es her, dass sich noch hartnäckige Vorurteile über das Privat- und Berufsleben von Programmierern hielten: Lange Arbeitstage in grauen, engen Büros, in der Freizeit wird weiter programmiert, und an die Sonne kommen Menschen mit diesem Beruf nicht allzu häufig. Dazu kommen steile Hierarchien und wenig Entscheidungskraft. Kurz gesagt: Die Befehle kommen von oben, die Developer führen einfach nur durch – und das in nicht enden wollenden Tages- und Nachtstunden vor dem Bildschirm.

Konferenzen wie die enterJS sind eine gute Gelegenheit, über den Tellerrand zu schauen und neue Unternehmen kennenzulernen.

Sollten diese Vorurteile jemals der Realität nahegekommen sein, so hat sich das mittlerweile drastisch geändert. Junge Menschen – Generation X, Y, Z und wie sie alle heißen – legen Wert auf einen Arbeitsplatz, der sich wie ein verlängerter Arm für die Freizeitgestaltung anfühlt. Viel Freiheiten, aufregende Aufgaben, Abwechslung und Entscheidungs- sowie Entwicklungsmöglichkeiten. Dieses Gesamtpaket ist für Developer ähnlich bedeutend wie das Gehalt – das zwar laut einer Umfrage von WeAreDevelopers noch immer das entscheidendste Kriterium für die Jobwahl ist (78 % der Befragten), aber dicht gefolgt wird von Flexibilität (69 %), Entscheidungsfindungen und Konfliktlösungen im Unternehmen 56 %) und den zwischenmenschlichen Beziehungen (53 %).

Immer mehr junge Developer wechseln in den letzten Jahren bereits nach kurzer Zeit ihren Job, weil sie schnell feststellen, dass sie darin nicht ihre Erfüllung finden werden. So auch Michael, ein junger Softwareentwickler, der nach Abschluss des Bachelor-Studiums erst einmal eine Stelle in einem Großhandelsunternehmen angenommen hat. Das Gehalt war gut, ebenso wie

Zeitgemäß lernen und studieren - flexible und individuelle Wege in die Medienwirtschaft

im Gespräch mit Martin Endel - Culture & Product Development SAE Institute | Germany, Switzerland, Austria

Während an staatlichen Schulen und Einrichtungen noch von Digitalisierung gesprochen wird, Konzepte erarbeitet werden wie Lernen künftig gut gelingen kann, gibt es bereits erfolgreiche Modelle.

Um zu verstehen, was zeitgemäßes Lernen ausmacht, blicken wir zunächst auf die Bedürfnisse unserer Studierenden. Denn erfolgreiches Lernen ist nur gewährleistet, wenn **individuelle Bedürfnisse** genauso Berücksichtigung finden wie **unterschiedliche Lerntypen**.

Seit über 40 Jahren bildet das SAE Institute weltweit erfolgreich für die Medienbranche aus, und das nach einem äußerst praxis-orientierten Ansatz. Die Arbeit an Projekten und in Gruppen stellt den Kern des Lernens dar, die notwendige Theorie erhält den nötigen Raum und bildet knapp ein Viertel des Lernstoffes. So wird Gelerntes direkt in der praktischen Anwendung umgesetzt, erprobt und wesentlich vertieft und gefestigt.

Blicken wir nun auf die Methoden der Wissensvermittlung, so lassen sich insbesondere in Bezug auf die theoretischen Vorlesungen die Vorzüge der Digitalisierung für alle vorteilhaft zu Nutze machen. Alle Themen werden zu festen Zeiten morgens und abends live in Campus übergreifenden **Online-Vorlesungen** abgehalten. Studierende kommen in den Genuss unterschiedlicher Dozent*innen, so dass auch unterschiedliche Perspektiven eingenommen werden können. Dabei wird auch der Austausch unter den Studierenden im Fachbereich gefördert.

Über die live Vorlesungen hinaus steht jede Vorlesung im Nachgang als Aufzeichnung für die gesamte Studiendauer zur Verfügung, so dass diese zeitlich flexibel nachbereitet und sich optimal auf Prüfungen vorbereitet werden kann.

Betrachtet man die unterschiedlichen Lebenssituationen, in denen sich die Studierenden befinden, so wird man schnell feststellen, dass diese **Zeit- und Ortsunabhängigkeit** einen immensen Vorteil gerade für die **berufliche Weiterbildung** oder das berufsbegleitende Lernen und Studieren darstellt. Die fest einzuplanende Zeit am Campus beläuft sich meist nur auf einen festen Tag in der Woche.

Am Campus entfaltet sich dann das gesamte Potential. In intensiven und **kleinen Lerngruppen** wird von erfahrenen Dozent*innen aus der Medienwirtschaft gelernt. Darüber hinaus ermutigen wir zur Zusammenarbeit in peer-learning Groups - gemeinsam sowie fachbereichsübergreifend und somit interdisziplinär - an Projekten für das eigene **Portfolio** zu arbeiten; dies entsteht also bereits während des Studierens und fördert so einen schnellen Berufseinstieg nach dem Abschluß.

“Der offene Austausch untereinander und das praktische Lernen auf Augenhöhe in der Gemeinschaft stellt für uns den unumstößlichen Kern dar. So schulen sich neben den technisch-handwerklichen Fertigkeiten auch maßgebliche Softskills, die unerlässlich für den Erfolg im Berufsleben sind und zur optimalen Berufsvorbereitung gehören”

weitere Informationen unter:
www.sae.edu/bildungsangebot

das restliche Gesamtpaket – man hat es ihm ermöglicht, berufsbegleitend weiter zu studieren. Sein erklärtes Ziel ist es schließlich, eines Tages auch den Master-Titel in der Tasche zu haben.

Für das Unternehmen ist das wiederum eine Gelegenheit, einen talentierten Programmierer früh an sich zu binden. Ein beliebtes Modell, das nicht nur im DACH-Raum Anwendung findet. In der praktischen Umsetzung gibt es dabei aber oft ein Problem: Die angebotenen Jobs sind häufig so monoton, dass sie schon nach kurzer Zeit keine Weiterentwicklungsmöglichkeiten mehr bieten. So auch bei Michael, der bereits vor Abschluss des Master-Studiums innerlich mit seinem Job abgeschlossen hat. Wenn da nicht a) das Schuldgefühl gegenüber jenem Unternehmen wäre, das ihm das duale Studium ermöglicht hat, und b) die Angst überwiegen würde, als Job-Hopper abgestempelt zu werden, sodass kein neuer, ähnlich gut dotierter Beruf mehr in Sicht wäre.

_Keine Weiter' entwicklung' in Sicht?

Zugegeben: Das Beispiel von Michael ist ein fiktives. Und doch entspricht es ganz oder zumindest in Teilen der Situation zahlreicher Developer in Deutschland, Österreich und ganz Europa. Laut Umfragen sind zwei Drittel der deutschen Entwicklerinnen und Entwickler offen für neue Herausforderungen. Das Traurige daran: Viele bleiben letztlich doch in „ihrem" Unternehmen und bauen sich dort über Jahre hinweg eine Komfortzone auf. Zunächst ist der Grund dafür das falsche Ehrgefühl oder die Angst, nichts Besseres zu finden – irgendwann ist der einzige Grund zu bleiben dann nur noch der, dass man jetzt eh schon so lange hier ist.

So stagnieren die Karrieren vieler Programmierer, nur weil sie sich selbst nicht die Chance geben, das Segel zu setzen und neue Ufer zu erforschen.

Karriere-Reboot in Zahlen

Viele Programmierer sind in ihrem derzeitigen Job nicht nur glücklich, das belegen zahlreiche Umfragen. Die gute Nachricht: Wer eine neue Stelle sucht, wird schnell fündig.

Offen für einen Jobwechsel ...

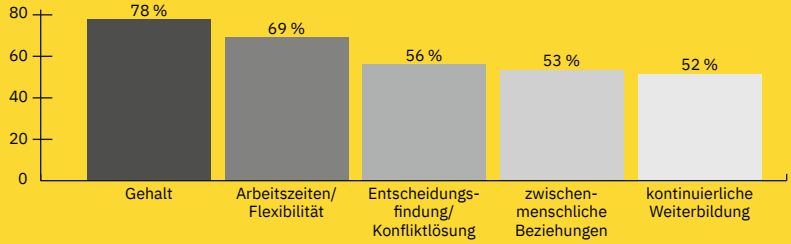
... sind aktuell rund zwei Drittel aller Softwareentwickler. Demnach ist derzeit nur ein Drittel nicht offen für den Jobwechsel. Unter den Wechselwilligen sind bereits 15 Prozent aktiv auf der Suche nach einer neuen Arbeitsstelle.

Gründe für den Jobwechsel ...

... gibt es viele: Hier steht ein zu niedriges Gehalt ganz oben auf der Liste. Rund jeder Zweite findet, die Bezahlung sei zu schlecht. Jeder Dritte fühlt sich dem Job emotional nicht (mehr) verbunden aufgrund der Entscheidungsfindung und Konfliktlösung im Unternehmen, oftmals aufgrund von Versäumnissen der Führungskraft. Ebenso jeder Dritte beurteilt ein schlechtes Arbeitgeberimage als treibenden Faktor für den Jobwechsel. Demgegenüber sind flexible Arbeitszeiten nur für jeden Vierten Grund, den Arbeitsplatz zu wechseln.

Prioritäten für bei der Jobwahl ...

... sind ziemlich eindeutig: Für 78 Prozent der Softwareentwickler ist das Gehalt ausschlaggebend bei der Entscheidung, ob Jobs für sie infrage kommen. Weitere Gründe stehen in Zusammenhang mit der Arbeitszeit und Flexibilität (69 Prozent), Aspekte wie Entscheidungsfindung und Konfliktlösung in den Unternehmen (56 Prozent), den zwischenmenschlichen Beziehungen (53 Prozent) sowie der kontinuierlichen Weiterbildung (52 Prozent).



„Ortswechsel? Sofort!" ...

... antwortet rund ein Drittel junger Talente, die einem Jobwechsel ins Ausland begeistert gegenüberstehen. Die maßgeblichen Faktoren bei einer Relocation sind Weiterentwicklung und ein Gehalt, die denselben Lebensstandard sichern. Berlin, Wien, New York, Barcelona und London waren die Top-Wahl für Städte, in denen Entwickler einen Umzug erwägen würden.

Verbesserungswürdig ...

... sind die Arbeitsverhältnisse speziell für Frauen. Jede zweite Frau gab an, im beruflichen Umfeld bereits mit einem schwerwiegenden Konflikt konfrontiert gewesen zu sein, und 60 Prozent waren der Ansicht, dass das Problem nicht gelöst wurde.

Quelle: Developer Report 2019 von WeAreDevelopers und MindTake Research

Der komplette 160-seitige Report mit weiteren Ergebnissen zur demografischen Verteilungen (wie Alter, Geschlecht etc.) sowie sozialen und persönlichen Prioritäten ist bei WeAreDevelopers (www.wearedevelopers.com/survey) erhältlich.

Denn auch andere Unternehmen bieten eine tolle Unternehmenskultur mit diversen Benefits, netten Kollegen und gutem Gehalt. Vor allem aber bieten sie die Chance, durch neue Erfahrungen Neues zu lernen.

Softwareentwicklung zählt zu den aufregendsten beruflichen Tätigkeiten unserer Zeit, Developer sind gefragt wie nie. Ein kleiner Insider-Tipp: Vom Fachkräftemangel sind zunehmend solche Arbeitgeber betroffen, die vor allem von gut ausgebildeten, technisch versierten Mitarbeitern abhängig sind. Aufgrund der fortschreitenden Innovation entstehen zunehmend schneller neue Aufgabengebiete, für die qualifizierte Developer aus jeder fachlichen Richtung gesucht werden.

Nie waren die Möglichkeiten besser!

Der Markt ist so unterbesetzt, dass selbst kleine Startups ihre Suche nach Developern auf ganz Europa ausweiten. Bei diesem Überangebot an Arbeitgebern gibt es für jeden Entwickler die Möglichkeit, einen Job zu finden, in dem Weiterentwicklung und Wohlfühlatmosphäre gelebt werden.

_Firmen bewerben sich bei Entwicklern

Wem das Verfassen von Bewerbungsschreiben zu mühsam ist, der kann es sich direkt sparen. Unternehmen sind auf der Suche nach zahlreichen Entwicklern – die Arbeitnehmer sind dabei nicht länger in der Rolle des Bittstellers. Job-Plattformen haben diesen Trend erkannt und gehen in ihrem Angebot auf die neuen Gegebenheiten ein. Sie verbinden Arbeitgeber mit potenziellen Kandidaten – vom Junior zum Senior, vom Spezialisten zum erfahrenen Allround-Developer. Die Plattformen bringen den Unternehmen die Anliegen der Entwickler näher und unterstützen sie dabei, ihre Zielgruppen besser zu verstehen. Das kommt auch den Developern entgegen. Sie geben auf der Plattform ganz einfach ihre Skills und Präferenzen an – natürlich ganz anonym – und erhalten anschließend mit dem intelligenten Matching-Algorithmus Jobvorschläge von Unternehmen, die den eigenen Anforderungen entsprechen. Und wenn die Developer doch lieber selbst aktiv werden möchten, verwenden sie ihre Daten einfach, um jederzeit nach passenden Arbeitgebern zu suchen – völlig kostenlos.

_Nicht mehr länger warten

Zu viele Karrieren geraten ins Stocken, zu viele berufliche Träume platzen, weil man ja doch „irgendwie zufrieden" beim aktuellen Arbeitgeber ist. Developer haben die unglaubliche Chance, ihren Arbeitsplatz selbst auszusuchen und ihn entsprechend mitzugestalten. Es liegt aber an ihnen, diese Chance auch zu nutzen.

Thomas Limbüchler

ist Developer Relations Strategist bei WeAreDevelopers und beschäftigt sich unter anderem mit Veränderungen, die gerade und in der Zukunft in der Karrieren von Software Developer passieren. Die Heise Medien sind seit 2019 mit 10 Prozent an WeAreDevelopers beteiligt.

Erfolg vorprogrammiert

Smart freelancing – Etengo schafft neue digitale Projekt-Perspektiven

Die digitale Revolution ist in vollem Gange und krempelt die Arbeitswelt um. Vor allem für DevOps-Entwickler brechen sonnige Zeiten an. Ihre Kompetenz zur Entwicklung digitaler Produkte und Services wird immer erfolgskritischer. Experten sehen die Bedeutung von Software- und Anwendungs-Entwicklung auf dem Vormarsch, da Qualifikationen in diesem Bereich die stärksten Treiber für Agilität und Innovationskraft in der Digitalisierung sind. Das betrifft insbesondere die Kernkompetenzen der DevOps-Entwickler oder -Engineers (Development, Entwicklung und Operations). Die Kombination dieser Fertigkeiten innerhalb eines Aufgabenprofils macht diese Fachkräfte, insbesondere als flexible Freiberufler, so attraktiv für Unternehmen. Denn damit lassen sich nicht nur qualitativ hochwertige Entwicklungsergebnisse erzielen, sondern vor allem Anwendungsleistungen massiv beschleunigen. Hinzu kommt die verbesserte Koordination in unterschiedlichen Teams, die DevOps innerhalb ihres Kompetenztableaus mitbringen. Viele Unternehmen legen aufgrund dieser Expertise sogar ihre Investitionsentscheidungen – beispielsweise im Bereich Einkauf von Cloud-Anwendungen – in die Hände ihrer DevOps. So eine aktuelle Erhebung des Identitätsanbieters Auth0 unter rund 350 weltweit tätigen Führungskräften. SaaS-Komponenten wie digitale Zahlungsabwicklungen oder Authentifizierungen sind wichtige Bausteine für DevOps, um Web-Anwendungen effizienter und schneller zu machen. 76 Prozent der Entscheider gaben an, über die Einbindung der DevOps mehr Zeitersparnis und Agilität über SaaS-Komponenten zu erreichen. DevOps haben also ein großes Mitspracherecht bei Investitionsentscheidungen.

Lust auf mehr Mitsprache? Lust auf Zugang zu interessanten Projekten? Lust auf faire Verhältnisse bei der Projekthonorierung?

SMART FREELANCING von ETENGO.
Hier geht´s zu unseren aktuellen Projekten:
www.etengo.de/it-projektsuche

Impressum We Are Developers!

Redaktion: iX | heise Developer
Telefon: 0511 5232-387, **E-Mail:** post@ix.de

Leitung: Alexander Neumann, Oliver Diedrich (verantwortlich für den Textteil)

Autoren dieser Ausgabe:
Nico Axtmann, Lutz Dietz, Simon Harrer, Lars Hupel, Christian Liebel, Thomas Limbüchler, Antti Pitkänen, Alexandra Waldherr

DTP-Produktion:
Lisa Hemmerling Heise Medienwerk, Rostock

Korrektorat:
Lara Bögner, Heise Medienwerk, Rostock

Titelbild:
© Stefan Wieland

Verlag
Heise Medien GmbH & Co. KG,
Postfach 61 04 07, 30604 Hannover; Karl-Wiechert-Allee 10, 30625 Hannover;
Telefon: 0511 5352-0, Telefax: 0511 5352-129

Geschäftsführer:
Ansgar Heise, Dr. Alfons Schräder

Mitglieder der Geschäftsleitung:
Beate Gerold, Jörg Mühle

Verlagsleiter:
Dr. Alfons Schräder

Anzeigenleitung (verantwortlich für den Anzeigenteil):
Michael Hanke (-167), E-Mail: michael.hanke@heise.de, www.heise.de/mediadaten/ix

Leiter Vetrieb und Marketing:
André Lux

Druck:
Dierichs Druck + Media GmbH & Co. KG, Frankfurter Straße 168, 34121 Kassel

Die Inserenten

Die hier abgedruckten Seitenzahlen sind nicht verbindlich.
Redaktionelle Gründe können Änderungen erforderlich machen.

DEVELOPER-KONFERENZEN + WORKSHOPS 2020

» Continuous Lifecycle »

Continuous Delivery Day: 24.11.2020
Online

betterCode()

PHP 8: 26.11.2020
Online

heise MacDev
Die Entwicklerkonferenz von Mac & i

SwiftUI, XCode, AR & Co.: 02.12.2020
Online

betterCode()

.NET 5.0: 03.12.2020
Online

[Container Conf]

Container Deep Dive: 09.12.2020
Online

betterCode()

Product Owner Day: 10.12.2020
Online

Veranstalter:

iX heise Developer dpunkt.verlag

Weitere Informationen unter:

www.heise.de/developer

“As Software Engineers, we bring chips to life.”

Software Engineers at Infineon dedicate their skills to developing products that count, such as security software and systems for self-driving cars. Without them, chips would never be as powerful as they are. You could say that their designs give chips a brain.

Do you want to know how Software Developer Roland keeps the bad guys of the cyber world in check and why Ruth thinks it’s a good thing when Softare Engineers are lazy?
Find employee stories, a contact person and open jobs at:
www.infineon.com/software-engineering

#beInfineon

Your impact: Write great code for tiny things.

Zertifikat seit 2010
audit berufundfamilie

Infineon