Ausgabe 03/2022

# Java aktuell

iJUG Verbund
www.ijug.eu

**Security**
TLS, Machine Learning, Keycloak

**Cybersicherheit**
Ethik im Fokus

**GraphQL**
-Anwendungen mit Spring

## Security

Java aktuell
Praxis. Wissen. Networking. Das Magazin für Entwickler
D: 4,90 EUR | CH: 9,80 CHF | A: 5,60 EUR | Benelux: 5,80 EUR
4 191978 304903 03

CloudLand

www.cloudland.org

# DAS CLOUD NATIVE FESTIVAL
# 4 TAGE – 4 THEMEN

## DAS EVENT DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY

Weitere Informationen

**29.JUNI BIS 2.JULI 2022**

im Phantasialand in Brühl

#CloudLand2022

Eventpartner: Heise Medien

# Liebe Leser/innen der Java aktuell,

freut euch in dieser Ausgabe auf wissenswerte Fachartikel unserer Autoren zum Thema Security. Den Anfang macht Nils Bokermann, der uns zeigt, welche Möglichkeiten und Mechanismen verwendet werden können, um die Sicherheit von HTTPS und dem darunterliegenden TLS-Zertifikat zu erhöhen. Im Anschluss zeigt Daniel Etzold anhand von Beispielen Sicherheitslücken im Machine Learning auf. Etwa, wie Bilderkennungssysteme systematisch getäuscht werden können. Wenn wir an Keycloak denken, denken wir auch gleichzeitig an Niko Köbler. Der Keycloak-Experte präsentiert in seinem Artikel, wie Keycloak IAM und SSO erfolgreich eingesetzt werden können. Mit Corinna Wiedenmann nehmen wir uns kurz eine Pause von technischen Artikeln und widmen uns den ethischen Aspekten in der Cybersicherheit.

Im folgenden Artikel von Nils Hartmann stellt dieser das neue Projekt "spring-graphql" vor, mit dessen Hilfe die Entwicklung von GraphQL-APIs in Spring ermöglicht wird. In der letzten Ausgabe der Java aktuell 2/22 hat Matthias Eschhold uns gezeigt, wie wir durch Clean Architecture eine flexiblere Anwendungsarchitekur erreichen. Dabei erklärte er uns die Basis für mehr Flexibilität. In dieser Ausgabe findet ihr ab Seite 38 den zweiten Artikel seiner dreiteiligen Reihe "Flexibilität auf dem nächsten Level".

Nicht nur Programmiersprachen entwickeln sich stets weiter, sondern auch die Mittel und Wege, diese zu erlernen. Johannes Schildgen stellt einige neue Möglichkeiten dazu vor. Zum Abschluss nimmt uns Edwin Günthner mit auf eine Zeitreise durch die Organisation in der Softwareentwicklung, mit dem Ziel, aufzuzeigen, wie Techniken auf große Teams skaliert werden können.

Wir wünschen euch viel Spaß beim Lesen!

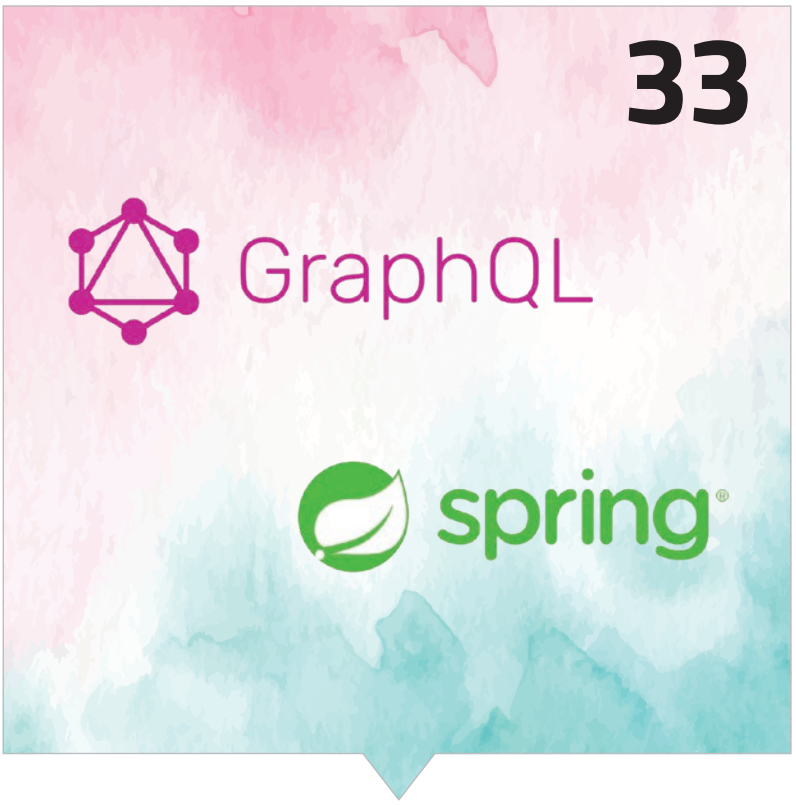
Eure

**Lisa Damerow**
Redaktionsleitung Java aktuell

# Inhalt


*Welche Möglichkeiten gibt es, die Sicherheit von HTTPS/TLS zu erhöhen?*


*Ethische Aspekte in der Cybersicherheit*




*So können Sie mit mit Spring GraphQL-APIs entwickeln*


*Agilität gestern und heute*

# Das Java-Tagebuch

Andreas Badelt, stellv. Leiter der DOAG Java Community


*Das Java-Tagebuch gibt einen Überblick über die wichtigsten Geschehnisse rund um Java.*

## 23. Dezember 2021

### Java-Community on Twitter

Java bekommt zu Weihnachten einen eigenen Community-Account auf Twitter, ins Leben gerufen von Arun Gupta und Markus Eisele. Der dürfte neben dem offiziellen Java-Handle von Oracle mit einer halben Million Followern auch Potenzial haben. Fröhliches Zwitschern *[1]*!

## 30. Dezember 2021

### Temurin: Pakete für Linux

Eclipse Temurin („das neue AdoptOpenJDK") baut jetzt auch RPM- und DEB-Pakete für die Installation auf Linux-Systemen *[2]*. Mir reicht sdkman, aber ich bin auch kein Linux-Admin. Ein weiteres „Produkt" sind JREs. Bislang war gar nicht geplant, sie für Java-Versionen ab 17 weiter zu liefern, da der Support im Upstream-Projekt OpenJDK eingestellt wurde (mit der Begründung, dass sich mit jlink ja jederzeit das eigene, passende JRE erstellen lässt). Laut Temurin-Blog war es „eine der längsten Diskussionen, die wir im Projekt hatten". Aber jetzt wurde – auch auf Grundlage vieler Anfragen der Temurin-Nutzer – entschieden, weiterhin „General Purpose JREs" als Teil von Temurin bereitzustellen.

## 5. Januar 2022

### The State of Valhalla

„The State of Valhalla" heißt der Titel, den der zuständige „Java Language Architect" Brian Goetz für seine Blog-Serie auf den OpenJDK-Seiten gewählt hat. Nein, nordische Götter haben sich nicht das Konzept „Staat" zu eigen gemacht – Valhalla ist das bereits 2014 offiziell gestartete Projekt zur Modernisierung der Sprache, das insbesondere Objekte und primitive Datentypen miteinander versöhnen möchte. Zitat Goetz zur Entscheidung im Jahr 1995: „It was not yet known how to get away with ‚everything is an object' and still offer reasonable numeric performance. It didn't seem so bad at the time, and we've been able to accomplish great things despite it, but it is an ongoing tax on developers, library designers, and users."

Im ersten Eintrag beschreibt Goetz die Probleme und ihre gemeinsame Wurzel: Objekt-Identität. Dann geht er auf die grobe Einteilung der geplanten Lieferungen von Valhalla ein: Zuerst Value Objects (also Instanzen ohne Identität, deren Felder ausschließlich final deklariert sind), dann Primitive Classes („*codes like a class, works like an int*"), gefolgt von der Migration und der konsistenten Behandlung bestehender Primitives (Wrapper-Klassen). Schließlich universelle und zuletzt spezialisierte Generics für alle Typen. Der zweite Blog-Eintrag behandelt die beabsichtigten Änderungen am Java-Language-Modell und der letzte die Sicht der JVM auf die neuen Typen. Ein bisschen zu viel, um es hier detaillierter zu beschreiben, aber guter Lesestoff für eine ruhige Stunde *[3]*.

## 15. Januar 2022

### Lückensuche im MicroProfile

In der MicroProfile Google Group läuft eine längere Diskussion: Wie lässt sich für MicroProfile – beziehungsweise Quarkus oder andere Implementierungen – ein schneller Überblick über verfügbare Integrationen erreichen, wie ihn Spring Boot schon lange bietet? Wie stehen MicroProfile/Quarkus im Vergleich überhaupt da? Gibt es größere Lücken? Am Ende erstellt Reza Rahman als pragmatischen Startpunkt ein Google Doc mit ersten Details, das – so die Hoffnung – per „Crowd Sourcing" reifen soll und dann vielleicht demnächst als Grundlage für eine Erweiterung des MicroProfile-Starters dient. Zum Anschauen und vielleicht auch direkten Ergänzen geht's hier entlang *[4]*.

## 21. Januar 2022

### Neue APIs für MicroProfile

Es gibt unterschiedliche Meinungen in der MicroProfile Working Group, wie mit dem Metrics-API weiter verfahren werden soll. Behalten? Schließlich ist das Feedback nicht so schlecht, das API fügt sich gut in die anderen MicroProfile-Spezifikationen ein und es hat schon viele Nutzer, die sonst migrieren müssten. Oder wegwerfen – und in direkter Zusammenarbeit mit Implementierungen wie Micrometer und auch OpenTelemetry etwas Neues schaffen? Die Tendenz scheint zu „behalten" zu gehen.

Gleichzeitig wird aber auch schon länger eine Annäherung an Micrometer diskutiert, die in der nächsten Version Metrics 5.0 umgesetzt werden soll. Das könnte am Ende auch zu „Breaking Changes" führen (MicroProfile ist da ja verglichen mit Jakarta etwas „freier", was je nach Kontext gut oder schlecht sein kann).

Eine andere lebhafte Diskussion dreht sich darum, eine „Data Access"-Abstraktionsschicht zu schaffen – für SQL, NoSQL, und mehr... Es könnte auf eine weitere Kooperation mit Jakarta hinauslaufen, da es dort mit Jakarta Data bereits ein ähnliches Proposal gibt.

Nachtrag: Otavio Santana steckt hinter Jakarta Data und hat jetzt einen Artikel dazu auf Medium veröffentlicht *[5]*.

## 25. Januar 2022

### GraalVM 22

Die neue GraalVM-Version 22 wirbt unter anderem mit noch einmal kleineren nativen Images. Die mit 21.3 gekommene neue Garbage-Collector-Policy ist jetzt per Default aktiviert und soll für deutlich weniger Memory-Bedarf sorgen. Außerdem gibt es einen in die verschiedenen Schritte gegliederten Build-Output mit Details und Statistiken zu Native Images. Der Compiler kann jetzt selbstständig von Just-in-Time-Compilation in den Ahead-of-Time-Modus umschalten, wenn keine ausreichenden Profil-Informationen vorliegen, sodass möglichst immer noch eine gute Optimierung erreicht wird. Dieses Verhalten lässt sich laut Release-Notes nicht beeinflussen oder abschalten, da es „im Kern des Compilers" liegt. Eine weitere Neuerung, die jedoch unabhängig vom Release ist, sind die „GitHub Actions for GraalVM", die das Aufsetzen der Workflows mit der GraalVM und ihren Tools erleichtern sollen – wenn man denn GitHub nutzt.

## 31. Januar 2022

### MicroProfile-6.0-Ideen

Details des für Juni geplanten MicroProfile-6-Releases stehen fest: Alle für das Release aktualisierten Spezifikationen sollen grundsätzlich auf Java 11 als Source-Level umsteigen (bislang ist es immer noch 8). Telemetry 1.0 kommt als neue Spezifikation hinzu. Hier wird es etwas kompliziert: MicroProfile Telemetry soll auf OpenTelemetry basieren, aber (zumindest vorerst) nur den Tracing-Teil von dessen API enthalten. Im selben Zug wird MicroProfile OpenTracing aus der Plattform-Spezifikation entfernt (aber gegebenenfalls als „Standalone Specification" weitergeführt), da ja auch OpenTracing von dem umfassenderen OpenTelemetry abgelöst wird. Für Metriken hat MicroProfile aber, wie schon erwähnt, eine eigene Spezifikation, die als Metrics 5.0 im neuen Plattform-Release 6 landen soll. Da wird sich in Zukunft noch einiges bewegen.

Was soll's noch in Release 6 geben? OpenAPI 3.1 und JWT 3.0 stehen auf der Liste. Außerdem stehen zwei bisherige „Standalone Specifications" vor der Aufnahme: GraphQL – in Version 2.0 – und eventuell auch Reactive Messaging 3.0.

Update: Kurz bevor ich dieses Tagebuch schließe, kommt wieder Bewegung in die Diskussion um Metrics. Ein Vorschlag mit drei Optionen liegt vor. Option 1: Metrics 5.0 schreibt Micrometer als verpflichtende Implementierung vor. Option 2: Es schreibt Micrometer und OpenTelemetry als verpflichtende „Doppel-Implementierung" vor, sodass Nutzer wählen können. Option 3: Metrics 5.0 trifft, wie es eigentlich bei Spezifikationen sein sollte, keine Annahmen über die Implementierung. Über die Optionen soll zeitnah in der MP Google Group abgestimmt werden – mehr dazu dann im nächsten Tagebuch. Wer die Details zu den Vorschlägen mit Pros und Cons lesen möchte, findet sie im Google Doc *[6]*.

## 3. Februar 2022

### Open Source und Sicherheit

Nach dem log4shell-Desaster ist gerade die Bereitschaft wieder erhöht, in die Sicherheit von Open-Source-Software zu investieren. Die aus der Linux Foundation hervorgegangene Open-Source-Security-Foundation hat sich nach Verhandlungen im Weißen Haus 5 Mio. US-Dollar Anschubfinanzierung für ihr Projekt Alpha-Omega gesichert (wobei das Geld von Google und Microsoft kommt, nicht aus dem Weißen Haus, aber sei's drum). Alpha ist der Projektteil, der sich auf „die kritischsten OSS-Projekte fokussieren" soll, um diese gezielt zu beraten und in ihren sicherheitsrelevanten Aufgaben zu unterstützen. Omega soll für den „long tail" von OSS-Projekten da sein, die nicht individuell unterstützt werden können. Dabei sollen mindestens 10.000 Projekte weitgehend automatisiert gescannt und im Bedarfsfall kontaktiert werden.

In dem Zusammenhang ist vielleicht auch die Meldung zum Machine-Learning-gestützten Analyse-Werkzeug CodeGuru von AWS interessant. Das soll jetzt unter anderem potenzielle Log-Injections, die ja bei log4shell ausgenutzt wurden, in Java- und Python-Code automatisch erkennen. Wenn ich das dann wiederum mit neueren Berichten über den Code-Generator AlphaCode von Googles DeepMind-Gruppe kombiniere, frage ich mich, ob es nicht auch längst Schadcode-Generatoren gibt, die ML-gestützt lernen, wie ihr Code nicht von Analyse-Werkzeugen erkannt wird…

## 16. Februar 2022

### Jakarta-EE-10-Status

Nichts wesentlich Neues von Jakarta EE 10. Die Einzelspezifikationen sind noch nicht alle fertig, wenn auch meist nur noch Kleinigkeiten fehlen. Aber zur Plattform-Spezifikation sind noch eine ganze Reihe von Tickets offen, unter anderem zur Definition des neuen Core-Profile. In Quartal 1 wird das Release sicher nicht mehr kommen. Im Hintergrund treibt die Jakarta EE Ambassadors um, wie die „Adoption" von Jakarta gesteigert werden kann und Nutzer zumindest schon mal zur Migration auf die aktuelle Version motiviert werden können – unter anderem mit Aufklärung über hilfreiche Werkzeuge: wie dem Eclipse Transformer, der EE-8-in EE-9-Applikationen umwandeln kann (und umgekehrt); enthalten etwa in Payara oder Open Liberty, um EE-8- und -9-Applikationen direkt gemeinsam


deployen zu können, aber auch als Maven Goal „erhältlich". Oder, noch einen Schritt davor, die automatisierte Migration von Java EE zu Jakarta EE, wie sie etwa IDEA bietet.

## 22. Februar 2022

### Quarkus 2.7 und die Superhelden

Quarkus ist vor Kurzem in Version 2.7 freigegeben worden. Eine kleine Verbesserung für die CLI: Diese lässt sich jetzt direkt über homebrew, sdkman, JBang und vermutlich bald noch weitere Paket-Manager installieren; der „Dev Mode" hat jetzt ein interaktives Terminal. RESTEasy Reactive unterstützt jetzt die Kotlin-Serialisierung. Und viele weitere kleine Features. Was vielleicht spannender ist: Die „Quarkus Superheroes": eine aus mehreren Microservices bestehende Demo-Applikation, die die Quarkus-Features, aber auch die Konzepte dahinter, umfassend demonstrieren soll. Quasi der Cargo Tracker von Quarkus *[7]*.

## 24. Februar 2022

### JDK 18 auf der Zielgeraden

Java 18 hat den „Final Release Candidate"-Status erreicht und dürfte pünktlich am 22. März verfügbar sein. Die Features hatte ich schon aufgezählt – bis auf zwei: Die Abkündigung (Deprecation) von finalize()-Methoden, die aufgrund von Performance- und Security-Problemen in einer zukünftigen JDK-Version komplett verschwinden sollen; und ein neues Service-Provider-Interface für die Auflösung von Internet-Adressen (unter anderem weil die aktuell blockierenden Betriebssystem-Aufrufe ein Problem für das neue, leichtgewichtige Thread-Modell von Project Loom sind). Alles hier zu sehen *[8]*.

Eine Sache hätte ich fast vergessen, die steht da nämlich nicht: Mit JDK 17 wurde ja auch der Security Manager abgekündigt – eine eigene Diskussion. Erst mit 18 wurde aber die Property „ java.security.manager" auf den Default „disallow" gesetzt, das bedeutet, soll der Security Manager weiter genutzt werden, muss dieses Flag explizit beim Starten der JVM gesetzt werden. Was zumindest dazu führt, dass das entsprechende Programm korrekt starten kann, aber nicht unbedingt heißt, dass alles weiterläuft wie vorher: Die Implementierung des Security Manager wird sukzessive zurückgebaut, sodass erwartete Checks gegebenenfalls gar nicht mehr durchgeführt, sondern einfach ignoriert werden.

## 3. März 2022

### Jakarta Commons bei Eclipse angekommen

Das „neue" Jakarta Commons, als Sammlung von Basis-APIs und Werkzeugen für Jakarta EE, ist jetzt offiziell von der Eclipse Foundation akzeptiert worden.

### Referenzen:

[1] *https://twitter.com/i/communities/1471178821906821122*
[2] *https://blog.adoptium.net/2021/12/eclipse-temurin-linux-installers-available/*
[3] *https://openjdk.java.net/projects/valhalla/design-notes/state-of-valhalla/01-background*
[4] *https://groups.google.com/g/microprofile/c/IxAHXuSATP8*
[5] *https://medium.com/xgeeks/learn-to-access-java-database-with-jakarta-data-17fc3e2549b0*
[6] *https://docs.google.com/document/d/1VtlwZw6QIWckWHOvBsXNuj77xXtM07f4x64y2WnwjUk*
[7] *https://github.com/quarkusio/quarkus-super-heroes*
[8] *https://openjdk.java.net/projects/jdk/18/*

**Andreas Badelt**
stellv. Leiter der DOAG Java Community
*andreas.badelt@doag.org*

Andreas Badelt ist stellvertretender Leiter der DOAG Java Community. Er ist seit dem Jahr 2001 ehrenamtlich im DOAG e.V. aktiv, zunächst als Co-Leiter der SIG Development und später der SIG Java. Seit 2015 ist er stellvertretender Leiter der neugegründeten Java Community innerhalb der DOAG. Beruflich hat er seit dem Jahr 1999 als Entwickler und Architekt für deutsche und globale IT-Beratungsunternehmen gearbeitet und ist seit dem Jahr 2016 als freiberuflicher Software-Architekt unterwegs („www.badelt.it").

# Markus' eclipse-Corner

Es ist irgendwie surreal. Lisa Damerow wartet wie immer auch auf meine Kolumne, den Abgabetermin habe ich mal wieder verpennt. Ich sitze, wie auch sonst immer, gemütlich mit dem Laptop am Fenster, vor mir dampft ein Tässchen Espresso vor sich hin, ein bisschen sieht es aus wie das Java-Logo. Ich versuche über den Stand bei der Eclipse Foundation zu texten, aber irgendwie fällt mir nichts ein. Die Sonne scheint mir ins Gesicht, draußen flattern Vögel, ab und zu kommt ein Auto vorbei. Ich höre den Pizzabäcker nebenan quer über die Straße mit den Nachbarn quatschen. Das internationale Flair dieser urbanen Gegend hat mich schon vor 20 Jahren begeistert, als ich kurz nach dem Studium hierhergezogen bin. Viele Informatiker gibt es hier, in direkter Nähe trifft sich die örtliche Linux User Group, es ist ein Studentenviertel, mit vielen „Altstudenten" wie mir.

Solche Viertel gibt es überall auf der Welt, und es gab sich auch in ukrainischen und russischen Großstädten. Auch dort scheint die Sonne, auch dort schrieben bis vor Kurzem viele ausgezeichnete Journalisten und Informatiker ihre Kolumne oder hackten an ihrem aktuellen Open-Source-Projekt – einige davon sogar gemeinsam mit mir. Mit ihnen war ich regelmäßig, teilweise täglich, in Kontakt über E-Mail und Twitter. Und bin es teilweise noch immer.

Doch seit einigen Wochen ist alles anders. Putin brachte den Krieg in die Ukraine, und mit diesem Terror, Leid, Vertreibung, Tod. Ich weiß, ihr wollt nicht auch noch in diesem Magazin vom Krieg lesen, die Medien überschwemmen einen sowieso schon mit diesem Thema.

Aber, auch wenn die Ukraine weit entfernt scheint, für mich und viele andere, die in die Projekte bei der Eclipse Foundation involviert sind, ist die Ukraine in diesen Tagen sehr, sehr nah.

Krieg ist etwas Schreckliches, und die Weltgemeinschaft muss alles tun, um das Kämpfen zu beenden. Es ist schon fast grotesk, in diesen Tagen eine fröhliche JavaLand feiern zu wollen, oder über den Projektfortschritt bei Jakarta zu berichten.

Doch würde es den Menschen, meinen Freunden und Projektpartnern auf beiden Seiten der Frontlinie, etwas bringen, es nicht zu tun? Vermutlich nicht.

Vielleicht ist es sogar eher ein Zeichen: Seht her, wir lassen uns unsere liberale Lebensweise, unsere Offenheit, unsere Werte und Überzeugung, nicht nehmen!

Ich verurteile niemanden, nein, ich kann sogar wirklich sehr gut nachfühlen, dass man sich in diesen Tagen absichtlich mal zwei, drei Tage aus der realen Welt ausklinkt und in die JavaLand-Bubble, in die Arbeit oder in ein Open-Source-Projekt flüchtet.

Man kann sich nicht immer nur rund um die Uhr mit dem Krieg beschäftigen. Es hilft ja keinem. Insofern berichte ich nun also schweren Herzens, gelähmt durch die schrecklichen Bilder und Nachrichten, die von allen Seiten einprasseln, dass Jakarta EE 10 durchaus auf gutem Wege ist, aber sich Teile davon – wie üblich – um Monate verzögern.

Allem voran JAX-RS 3.1, bei dem die extra dafür bezahlten Mitarbeiter von Oracle, IBM und Red Hat in den letzten Wochen und Monaten leider, sagen wir mal, „sehr zurückhaltend" mit eigener Arbeitsleistung waren, gleichwohl aber fordernd gegenüber anderen Projektteilnehmern auftreten. Es freut mich daher insbesondere, dass mit Jersey 3.1-M3 inzwischen eine erste Implementierung nach der neuen Spezifikation zertifiziert ist. Endlich! Der Weg ist nun frei zu großen Umbauten in Richtung JAX-RS 4.0, wie beispielsweise der Ersatz des proprietären CI-Frameworks durch CDI, aber auch für die Unterstützung von JDK 17 in Jakarta EE.

Es ist offensichtlich, dass man bei den großen Unternehmen inzwischen verstärkt darauf setzt, dass andere die Arbeit machen. Verstanden. Das ist einerseits schade, da uns diese Arbeitsleistung fehlt. Andererseits ist es verständlich, dass diejenigen neue Features einbauen sollen, die sie haben wollen: Wir.

Daher danke ich an dieser Stelle den iJUG-Open-Source-Stipendiaten, die sich in den vergangenen Monaten für Jakarta EE und Adoptium eingesetzt haben. Bravo!

Ein neuer Grund für die Verzögerungen bei „unseren" Projekten ist nun hinzugekommen: Die Beteiligten sind durch den Krieg schockiert, vertrieben oder gar Teil der Kampfhandlungen. Wer weiß.

Umso mehr mein Appell an alle, die das unglaubliche Privileg haben, in einer befriedeten Region der Erde zu leben und zu arbeiten: Beteiligt euch an Open Source! Unterstützt bitte diejenigen, die derzeit nicht mitprogrammieren können, in ihrem Bestreben für freie Software, indem ihr in die Projekte einsteigt und eure Arbeitsleistung einbringt.

Danke, auch im Namen all derer, die derzeit ganz andere Sorgen haben als Terminverschiebungen bei Jakarta EE.

**Markus Karg**
*markus@headcrashing.eu

Markus Karg ist Entwicklungsleiter eines mittelständischen Softwarehauses sowie Autor, Konferenzsprecher und Consultant. JAX-RS hat der Sprecher der Java User Group Goldstadt von Anfang an mitgestaltet, zunächst als freier Contributor, seit JAX-RS 2.0 als Mitglied der Expert Groups JSR 339 und JSR 370.




# TLS – und weiter?

**Nils Bokermann, Freiberufler**

*HTTPS, und damit TLS, wird überall eingesetzt, aber TLS ist mehr als nur das Zertifikat auf dem Server einspielen. Welche Mechanismen können genutzt werden, um die Sicherheit zu erhöhen und welche Trade-offs geht man damit ein?*

HTTPS und das darunter liegende TLS-Protokoll sind aus dem Internet nicht wegzudenken. Die meisten Websites benutzen diese Verschlüsselungstechnologie, von allen gängigen Browsern wird mittlerweile von HTTPS als Standard ausgegangen. Viele von uns, den Autor eingeschlossen, sind jedoch geneigt, eben das Zertifikat auf den Server zu legen und damit ist die Geschichte gegessen. Stimmt meist auch so, aber ein kleiner Blick hinter die Kulissen zeigt uns, welche Möglichkeiten im Protokoll-Stack rund um die TLS-Verschlüsselung und X.509 noch verborgen sind. Um zu verstehen, was eine entsprechende Änderung für Auswirkungen hat, ist es sinnvoll, das Protokoll und die wichtigsten Objekte zu kennen.

In diesem Artikel wird auf die hinter asymmetrischer Kryptografie stehende Mathematik und die Art der Zertifikatsvalidierung nicht eingegangen. Dies liegt in der Komplexität der verwendeten Verfahren. Es wird stillschweigend davon ausgegangen, dass eine Zertifikatsvalidierung durch das Zertifikat aus dem signierenden Schlüsselpaar möglich ist, auch ohne die genauen Zusammenhänge und Mathematik zu erläutern.

## TLS Simple

Der „simple" Fall, der im Allgemeinen gezeigt wird, ist das Zusammenspiel von Alice und Bob. Diese beiden Personas werden uns durch den Protokoll-Stack begleiten. In *Abbildung 1* wird der Ablauf des Key-Exchange zwischen Alice und Bob gezeigt. In einem ersten Schritt übergibt Alice (als Initiator der TLS-Session, Client-Seite) mit dem *Client Hello* die Verschlüsselungsalgorithmen, die von ihrer Seite aus eingesetzt werden können. Aus diesen Algorithmen kann Bob (Server-Seite) sich einen passenden Algorithmus auswählen und mit dem *Server Hello* an den Client schicken. Damit ist der Verschlüsselungsalgorithmus für den ersten Key-Exchange ausgehandelt. Um den Schlüsselaustausch fortzusetzen, schickt der Server eine weitere Nachricht mit dem Zertifikat, dem Server Key-Exchange und dem Server Hello-Done-Nachrichtentypen. Der essenzielle Bestandteil ist hierbei das Zertifikat des Servers. Mit diesem Zertifikat (technisch gesehen der Public Key mit einer Signatur) kann der Client das symmetrische *Premaster Secret* verschlüsseln und an den Server übermitteln. Mit diesem Schritt ist die asymmetrische Verschlüsselung beendet und es wird weiter mit dem symmetrischen Schlüssel verschlüsselt. Um im Weiteren mit einer größeren Schlüssellänge hantieren zu können, wird der symmetrische Schlüssel mit der *ChangeCipherSpec*-Nachricht ausgetauscht.

Für Alice und Bob ist nun ein abhörsicherer Kanal erzeugt worden. Dieser ist auch nachträglich durch einen Dritten quasi nicht zu entschlüsseln. Sowohl der Angriffsvektor des *Premaster Secret* ist durch die asymmetrische Verschlüsselung hinreichend gesichert (solange ein

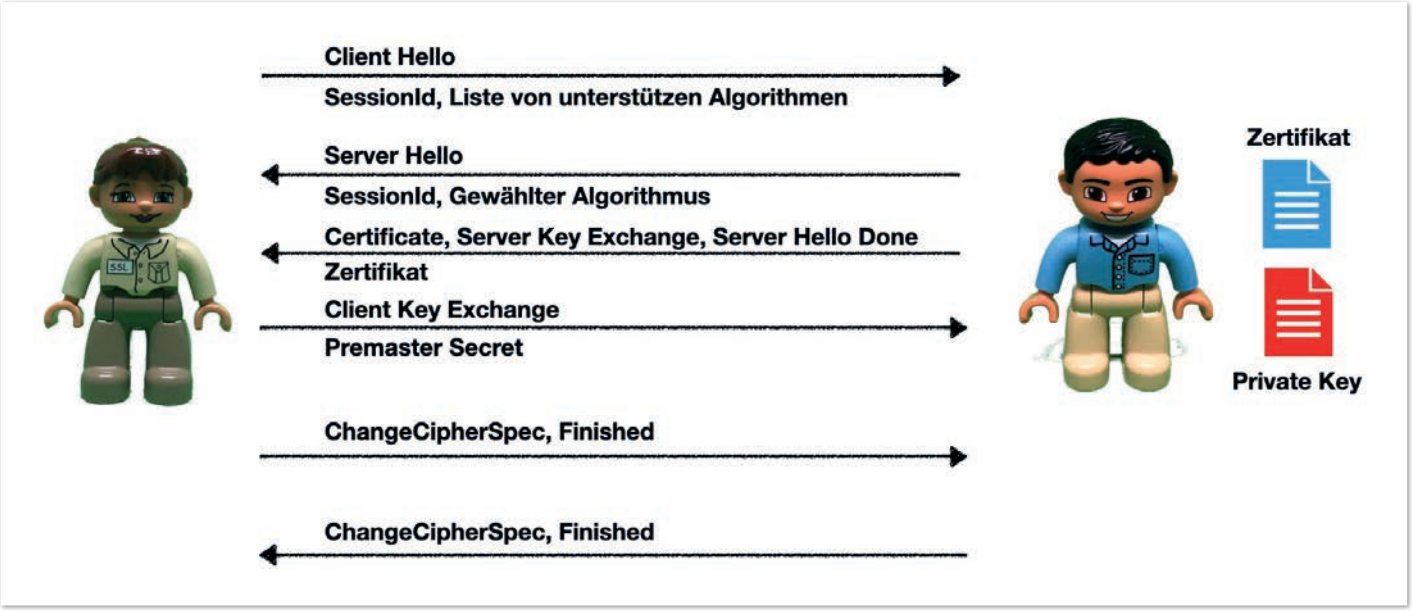



*Abbildung 1: Key-Exchange (© Nils Bokermann)*

genügend langer Schlüssel verwendet wird), als auch der Schlüsseltausch im *Change Cipher Spec* ist gut abgesichert. Beide symmetrischen Schlüssel sollten eine entsprechend hohe Entropie mitbringen, dass mit einer statistischen Analyse keine Daten gewonnen werden können, was bei dem letztlich ausgetauschtem HTTP-Stream nicht so ist.

Allerdings kann Alice bislang nicht prüfen, ob Bob auch wirklich der ist, der er vorgibt zu sein. Aus genau diesem Grund ist die Verbindung zwischen Alice und Bob durch eine Man-in-the-Middle-Attacke angreifbar.

## Zertifizierung durch Dritte

Um die Angreifbarkeit und die Überprüfung der Identität von „Bob" wird es im Weiteren gehen. Um diese Prüfung allgemein zu ermöglichen, kommt „Tom" („Trustworthy Tom") ins Spiel. Dieser vertrauenswürdige Dritte spielt die entscheidende Rolle bei der Überprüfung von Zertifikatsgültigkeiten. Wie sicher schon klar, ist das eine Zertifizierungsstelle (Certificate Authority, kurz: CA). Wenn wir uns den Prozess aus *Abbildung 2* anschauen, gleicht dieser sehr stark dem aus *Abbildung 1*.

Tatsächlich ändert sich in erster Linie die Zertifikatserstellung. Bob erstellt sein Zertifikat nicht selbst (Self-Signed), sondern kauft bei der Zertifizierungsstelle ein signiertes Zertifikat. Bei Vorliegen des Zertifikats der Zertifizierungsstelle (Root-Zertifikat) kann die Signatur von Bobs Zertifikat geprüft werden.

Weiterhin muss die Zertifizierungsstelle dafür sorgen, dass ihre (Root-)Zertifikate möglichst jedem Nutzer zur Verfügung stehen. Die

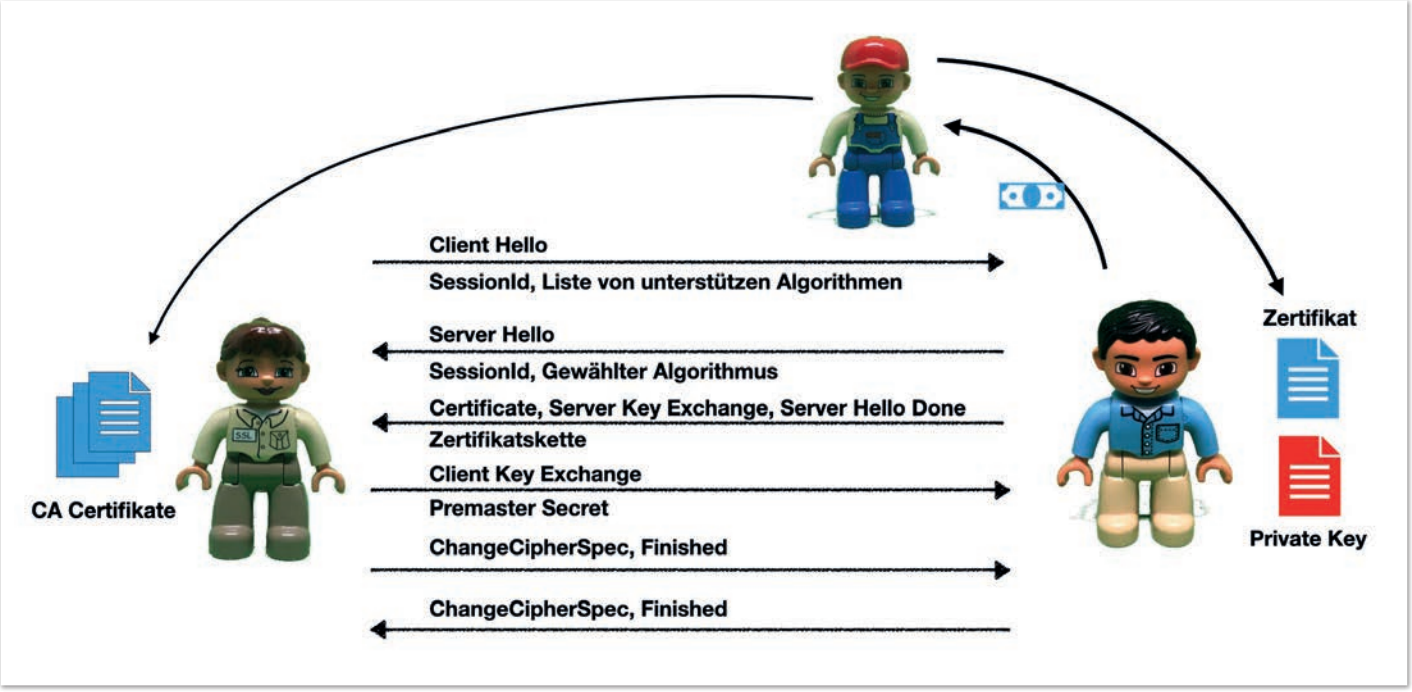



*Abbildung 2: Überprüfung mittels Zertifizierungsstelle (© Nils Bokermann)*

Zertifikatsverteilung dieser Zertifikate nimmt – je nach Betriebssystem – der Betriebssystemhersteller, Distributions-Maintainer oder der Browser-Hersteller vor.

Da die Zertifizierungsstellen aus verschiedenen Gründen nicht direkt mit dem CA-Zertifikat signieren, müssen mit der *Server-Hello*-Nachricht alle Zwischenzertifikate mitgegeben werden. Nur so ist Alice in der Lage, Bobs Zertifikat vom CA-Zertifikat aus zu überprüfen, indem die Zwischenzertifikate (Intermediate Zertifikate) vom Root-Zertifikat ausgehend bis zu Bobs Zertifikat überprüft werden.

In diesem Szenario hat Alice die Möglichkeit, über einen kryptografisch sicheren Weg festzustellen, dass Bob auch derjenige ist, der er vorgibt zu sein. Allerdings gilt diese Sicherheit nur für den Ausstellungszeitpunkt des Zertifikats und ist maximal so gut wie die Verifikation, die die Zertifizierungsstelle durchführt. Diese Verifikation ist je nach Zertifizierungsstelle verschieden implementiert. Die Spannbreite geht von einer Zertifizierung durch eine E-Mail an *admin@domain*, über HTTP-Challenges (eine bestimmte Datei muss auf dem HTTP-Server zu finden sein) bis hin zu DNS-Challenges, die einen bestimmten TXT-Record lesen wollen.

Die Absicherung erwartet nicht, dass Bob seine Identität wechseln würde, eher, dass der private Schlüssel „verloren" geht. Der Verlust wäre nicht weiter schlimm, da dann das Zertifikat auch wertlos wäre. Verlust bedeutet in diesem Kontext eigentlich den Diebstahl oder das Kopieren des privaten Schlüssels, sodass ein anderer vorgeben kann, „Bob" zu sein.

## Zertifikatsrückrufliste

Auch gegen den Diebstahl des privaten Schlüssels gibt es eine Lösung. Die Zertifizierungsstelle kann einzelne Zertifikate zurückziehen (revoke). Die zurückgezogenen Zertifikate werden in einer Liste veröffentlicht (*Certificate Revoke List*, kurz: *CRL*). Während der Validierung des Server-Zertifikats kann die CRL von der Zertifizierungsstelle abgerufen und es kann überprüft werden, ob dieses Zertifikat zurückgerufen wurde. Im Zertifikat selbst kann der Endpunkt, von dem die CRL abgerufen werden kann, hinterlegt sein. Es gibt Zertifikate, in denen kein CRL-Endpunkt gepflegt ist, da die Zertifizierungsstelle keine CRL anbietet. Die CRL selbst ist letztlich eine signierte Liste von Zertifikatsseriennummern. Durch diese Signatur ist die Integrität der Liste gegeben.

Die CRL wird allerdings nicht ständig aktualisiert. In der Regel wird die CRL einmal wöchentlich aktualisiert. Damit sind diese Informationen auch nicht notwendigerweise auf dem aktuellen Stand. Der Vorteil ist allerdings, da ein „Next Update"-Zeitstempel in der CRL gepflegt wird, dass der Server diese Information für eine ganze CA im Cache halten kann.

## Online-Abfrage

Um bei kritischeren Verbindungen aktuellere Informationen zu bekommen, hält der Werkzeugkoffer natürlich auch etwas bereit. Mit dem *Online Certificate Status Protocol (OCSP)* [1] kann der Status eines Zertifikats abgefragt werden. Der Ablauf des Key-Exchange ändert sich quasi nicht. Auf der Client-Seite wird allerdings der Zertifikatscheck erweitert. Nachdem das Zertifikat empfangen und gegen das CA-Zertifikat validiert wurde, wird eine OCSP-Anfrage an den OCSP-Endpunkt initiiert. Das Ergebnis dieser Abfrage ist der *CertStatus* mit den Ausprägungen *good*, *revoked* und *unknown*.

Mittels OCSP kann also ein relativ aktueller Status des Zertifikats ermittelt werden. Je nach Zertifizierungsstelle kann auch hier ein gewisser zeitlicher Versatz zwischen Zurückziehen des Zertifikats und der richtigen Übermittlung des Status liegen. Ebenso ist die Behandlung des *OCSPResponseStatus tryLater* sicherheitsrelevant. Dieser Status könnte von einem Angreifer eingeschleust werden und damit die eigentliche OCSP-Aussage verdecken.

Häufig werden Bedenken hinsichtlich der Privatsphäre geäußert, da die OCSP-Abfrage „im Klartext" erfolgt. Allerdings wird nur die

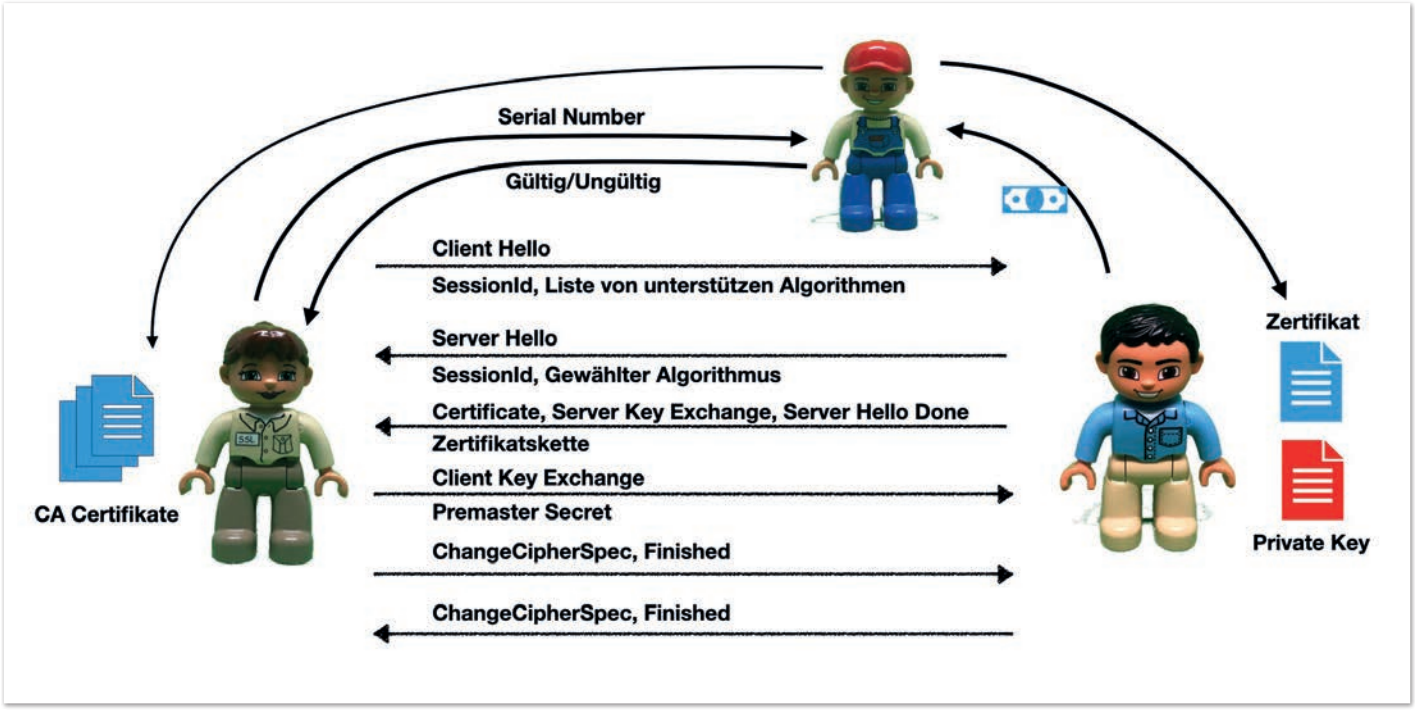



*Abbildung 3: OCSP-Validation (© Nils Bokermann)*

```
// for debugging:
System.setProperty("javax.net.debug", "all");
System.setProperty("java.security.debug", "all");

System.setProperty("com.sun.net.ssl.checkRevocation", "true");
Security.setProperty("ocsp.enable", "true");
```

*Listing 1: Setzen der System-Properties für OCSP in Java*

Seriennummer des Zertifikats übertragen. Aus Seriennummer und Zertifizierungsstelle lässt sich zwar grundsätzlich eindeutig auf ein Zertifikat schließen, allerdings lassen sich nur mit erheblichem Aufwand die Hostnamen ermitteln. Für die Zertifizierungsstelle ist das allerdings ein Einfaches, was dann jedoch die Frage nach der Vertrauenswürdigkeit der Zertifizierungsstelle aufwirft. Der entscheidende Nachteil von OCSP ist allerdings der Traffic, der auf die Zertifizierungsstelle zukommt. Man denke nur an eine Site wie Google oder Amazon.

Will man OCSP in seinem Browser aktivieren, so muss zumindest für Firefox die Konfiguration angepasst werden (`security.OCSP.require = true` in `about:config`). Weitere Informationen über die Browser-Spezifika finden sich in *[2]*. Für Java lässt sich die OCSP-Kontrolle wie in *Listing 1* erzwingen.

## OCSP-Stapling

Das Traffic-Problem des OCSP-Protokolls lässt sich vermeiden, indem man die OCSP-Abfragen dem Server überlässt und die signierte Information mit im Handshake austauscht. Wie in *Abbildung 4* zu sehen, wird der OCSP-Request vom Server aus geschickt. Der Zeitpunkt muss vor dem Senden der *Server-Hello-Done*-Nachricht liegen, ist aber weiter nicht genau spezifiziert. Damit kann der *OCSP Response* mehrfach verwendet werden und senkt somit den Traffic bei der Zertifizierungsstelle. Die Integrität der OCSP-Antwort ist auch weiterhin gegeben, da diese von der Zertifizierungsstelle signiert ist. Damit ist eine lückenlose kryptografische Nachverfolgung möglich.

Das OCSP-Stapling-RFC *[4]* stammt aus dem Jahr 2013 und ist mittlerweile in den gängigen Browsern integriert. Das heißt, aktuelle Browser unterstützen die Information, die über das Zertifikat mitgegeben wird. Die Einschränkungen der Nutzung sind allerdings die gleichen, wie in *[2]* weiter oben schon gezeigt.

Weiterhin muss auf Server-Seite ein Kanal zu der Zertifizierungsstelle geöffnet werden. OCSP-Responder sind üblicherweise nicht HTTPS abgesichert, da die Information an sich schon durch Signatur gesichert und nicht vertraulich ist. Dafür muss dann gegebenenfalls in einer Firewall wiederum eine Regel für Port 80 (HTTP) geöffnet werden.

Dies sind die Möglichkeiten, mit denen die Identitätsvalidierung erfolgen kann. Im Folgenden werden wir uns mit üblichen Use Cases beschäftigen und damit, wie diese sinnvollerweise gesichert werden können.

## Use Cases

Je nach Sicherheitsbedürfnis der Website oder des Service kann man sich nun entscheiden, welche Tiefe von Absicherung notwendig und sinnvoll ist.

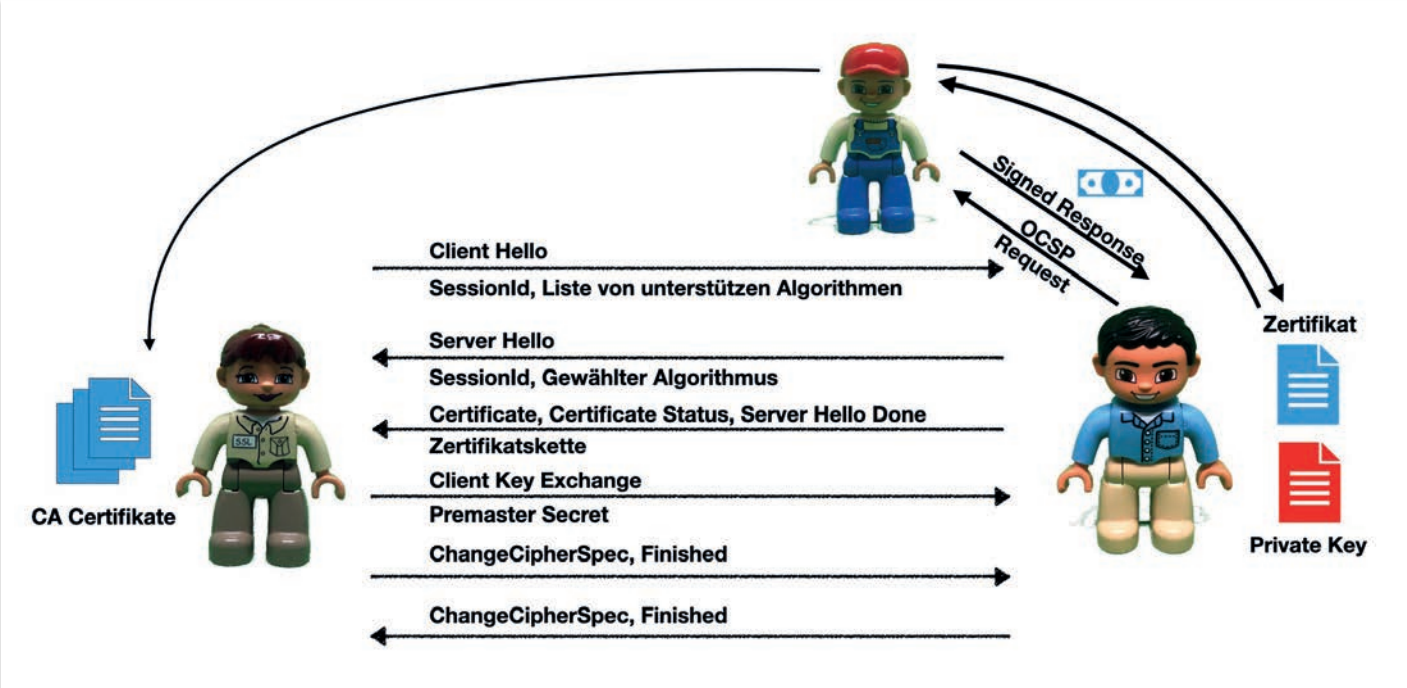



*Abbildung 4: OCSP-Stapling (© Nils Bokermann)*

```
$ openssl s_client -connect <host_no_ocsp>:443 -status

CONNECTED(00000005)
OCSP response: no response sent
[…]

$ openssl s_client -connect <host_ocsp>:443 -status

CONNECTED(00000005)
depth=2 C = US, O = DigiCert Inc, […]
verify return:1
depth=1 C = DE, O = D […]
verify return:1
depth=0 businessCategory = Private […]
verify return:1
OCSP response:
======================================
OCSP Response Data:
     OCSP Response Status: successful (0x0)
     Response Type: Basic OCSP Response
     Version: 1 (0x0)
```

*Listing 2: Überprüfen der OCSP-Information im TLS-Handshake*

Für eine klassische Webseite (statische „Werbe"-Seite) ist natürlich HTTPS heute notwendig. Da diese Information allgemein verfügbar ist, muss auch nach Verlassen des Servers nicht mehr Aufmerksamkeit gegen Veränderung gegeben werden als auf dem Server selbst. Das ändert sich deutlich, wenn der Nutzer mit der Website Daten austauschen kann, etwa bei einem Shopsystem. Gerade wenn möglicherweise auch noch Bankdaten oder Kreditkarten-Informationen übertragen werden, ist eine Überprüfung des Zertifikats unerlässlich. Ist die Website womöglich eine Bank, so ist eine sehr direkte und sichere Überprüfung sinnvoll und notwendig. Allerdings unterstützen nicht alle Banken OCSP-Stapling. Wer bei seiner Bank einmal kontrollieren möchte, siehe *Listing 2*.

Biete ich meinen Nutzern keine Website, sondern eine Schnittstelle (etwa ein REST-API) an, so gelten die Einschränkungen wie oben. Dieses API wird jedoch wahrscheinlich über User-Credentials gesichert sein, also ist eine Sicherung dieser Information notwendig. Eine Öffnung der Schnittstelle für eine begrenzte Zeit hätte zur Folge, dass die User-Credentials als kompromittiert angesehen werden müssten; eine mögliche Abrechnung in der Zeit, in der die Schnittstellenabsicherung gebrochen war, ist ebenfalls fraglich oder nicht möglich.

Für den Fall eines internen API, innerhalb eines Unternehmensnetzes, können natürlich andere Möglichkeiten der Zertifikatsvalidierung genutzt werden. Wird die Schnittstelle nur von einer kleinen Anzahl namentlich bekannter Nutzer verwendet, so ist auch eine organisatorische Lösung für einen Key-Verlust denkbar. Das gute alte Telefon hätte hier einen entscheidenden Geschwindigkeitsvorteil.

In der beliebten Container-Lösung Kubernetes ist es denkbar, innerhalb des Clusters ausschließlich HTTP zu verwenden und dafür ein Overlay-Netzwerk zu nutzen, das die Datenintegrität gewährleistet (etwa weave mit Verschlüsselung). Damit erspart man sich, die relativ komplizierten Service-Namen in die Zertifikate zu bekommen sowie die Zertifikatsvalidierung in jedem Service. Nach außen können wiederum herkömmliche Zertifikate in die Ingresses eingespielt werden, die auch „normale" Hostnamen enthalten.

## Quellen


[1] S. Santesson et al (2013), RFC 6960: X.509 Internet Public Key Infrastructure – Online Certificate Status Protocol – OCSP, IETF

[2] SSL.com Support Team (2021), *https://www.ssl.com/blogs/how-do-browsers-handle-revoked-ssl-tls-certificates/*

[3] E. Rescorla (2018), RFC 8446: The Transport Layer Security (TLS) Protocol Version 1.3, IETF

[4] Y. Pettersen (2013), RFC 6961: The Transport Layer Security (TLS) -- Multiple Certificate Status Request Extension, IETF

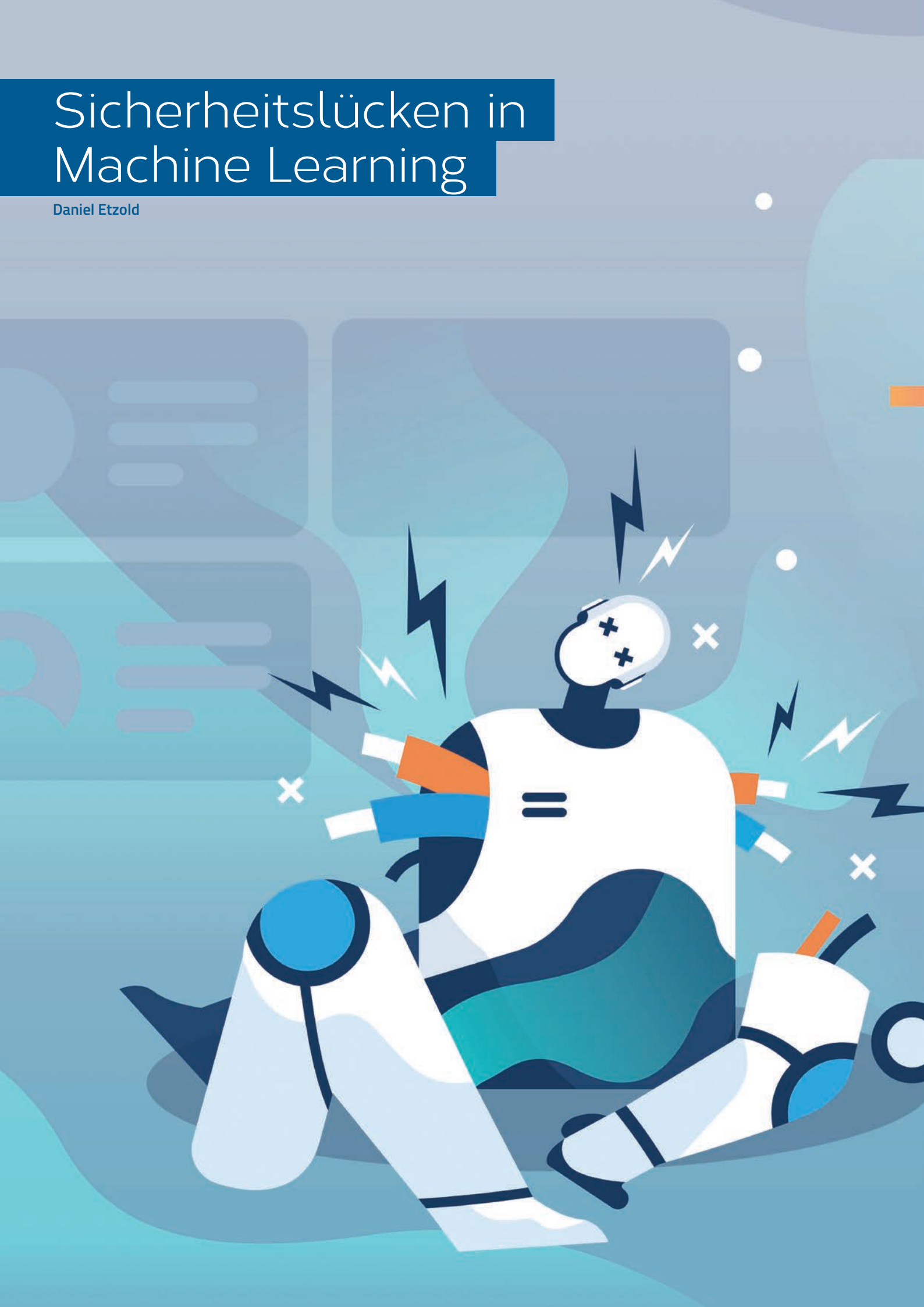


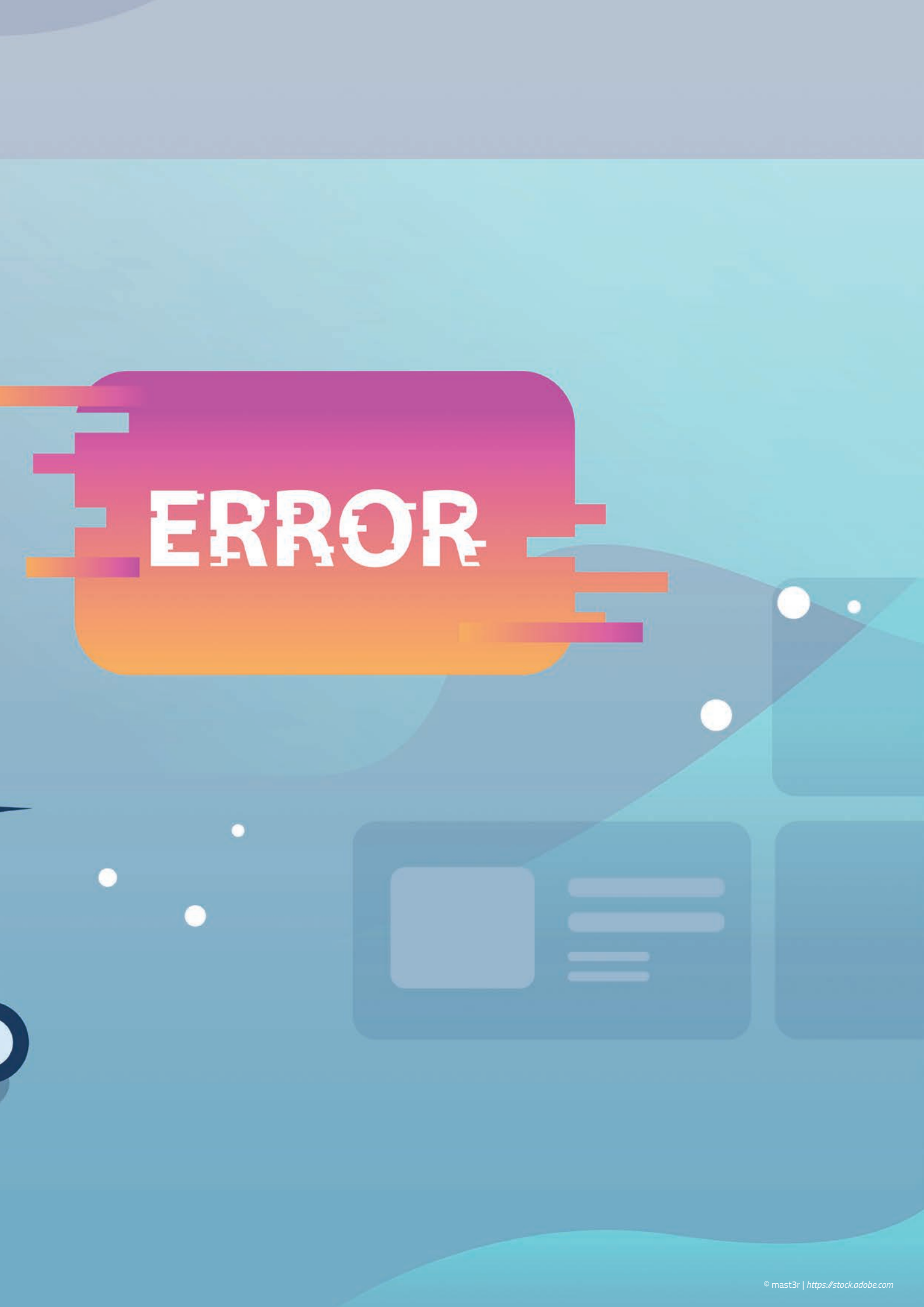
**Nils Bokermann**
Freiberufler
*nils.bokermann@bermuda.de*


Nils Bokermann ist als freiberuflicher IT-Consultant unterwegs und berät seine Kunden in Architektur- und Methodik-Fragen. Er stellt gerne Hype-getriebene Entscheidungen infrage und ist daher auch als Advocatus Diaboli bekannt.

# Sicherheitslücken in Machine Learning

Daniel Etzold





*Dieser Artikel gibt einen Überblick darüber, welche Sicherheitsschwachstellen in Machine-Learning-Modellen existieren können. So können etwa Bilderkennungssysteme mit Adversarial Examples getäuscht werden, Backdoors lassen sich in Modelle integrieren und die Parameter eines Modells lassen sich durch geschickte Abfragen aus den Antworten eines Modells extrahieren.*

In den letzten Jahren kam es zu einem noch nie dagewesenen Hype beim maschinellen Lernen. Zwar gehen die Anfänge des maschinellen Lernens bereits bis in die 1950er zurück, als Frank Rosenblatt das Perceptron vorgestellt hat, aber erst im ersten Jahrzehnt der 2000er konnten Fortschritte verzeichnet werden, die einen wahren Boom ausgelöst haben und schließlich zu einem noch nie dagewesenen Interesse führten. Im Jahr 2012 konnte AlexNet *[1]*, ein Convolutional Neural Network, bei der ImageNet-Challenge alle anderen teilnehmenden Machine-Learning-Modelle deklassieren und neue Maßstäbe setzen. Dieser Zeitpunkt wird auch als „AlexNet-Moment" bezeichnet und war der Beginn des Deep-Learning-Zeitalters. Seither wurden die Modelle kontinuierlich weiterentwickelt und Machine Learning konnte in immer mehr Produkten Einzug halten. Autonome Fahrzeuge oder Sprachassistenten wären ohne diese Entwicklung nicht denkbar gewesen.

Die Sicherheit von Machine-Learning-Algorithmen ist bis vor wenigen Jahren allerdings auf nur wenig Interesse gestoßen. Zwar wurden auch vor dem großen Hype bereits die Sicherheitseigenschaften der eingesetzten Algorithmen untersucht, doch erst vor einigen Jahren wurde entdeckt, dass selbst State-of-the-Art-Deep-Learning-Modelle leicht ausgetrickst werden können. Interessant daran ist, dass im Gegensatz zu Bugs und Sicherheitslücken in Software, die sich mit Patches in der Regel schnell korrigieren lassen, es sich bei den entdeckten Schwachstellen von Machine Learning häufig um inhärente Probleme im zugrunde liegenden Algorithmus handelt, die sich nicht ohne Weiteres korrigieren lassen. Es wurde gezeigt, dass viele populäre und weit verbreitete Architekturen für diese Schwachstellen anfällig sind. Wäre ein Angreifer in der Lage, eine Schwachstelle in einem Modell auszunutzen, kann das schwerwiegende Konsequenzen haben. Vor allem da Machine Learning auch immer häufiger in sicherheitskritischen Anwendungen zum Einsatz kommt, in denen die Gesundheit oder sogar das Leben eines Menschen von der Entscheidung eines Machine-Learning-Modells abhängen kann.

## Adversarial Examples

Bereits in den 90er Jahren wurde versucht, maschinelles Lernen auszutricksen. Zu dieser Zeit wurde zunehmend Machine Learning eingesetzt, um die Erkennungsrate von Spam-Filtern zu verbessern. Spammer haben deshalb unterschiedliche Techniken entwickelt, um diese intelligenten Filter auszutricksen. Im akademischen Bereich gab es allerdings nur wenige Publikationen. Erst mit der Entdeckung der Adversarial Examples *[2]* im Jahr 2013 hat sich das geändert und das Interesse an den Sicherheitseigenschaften von Machine Learning ist auf einmal sprunghaft angestiegen.

Bei Adversarial Examples handelt es sich um Bilder, die eine Bilderkennung auf sehr beeindruckende Art und Weise täuschen können. Drei Beispiele sind in *Abbildung 1* zu sehen. In der ersten Spalte sind Bilder zu sehen, die von AlexNet korrekt erkannt werden. In der mittleren Spalte sind Bilder mit Rauschen zu sehen, die auf die Bilder der ersten Spalte hinzuaddiert wurden, sodass die Bilder auf der rechten Seite entstanden sind. Das Rauschen ist stark verstärkt abgebildet, um es sichtbar zu machen. In Wirklichkeit ist das Rauschen sehr gering und ein Mensch würde einfach nur ein schwarzes Bild sehen. Deshalb kann ein Mensch auch keinen Unterschied zwischen den linken und den rechten Bildern wahrnehmen. Das Rauschen

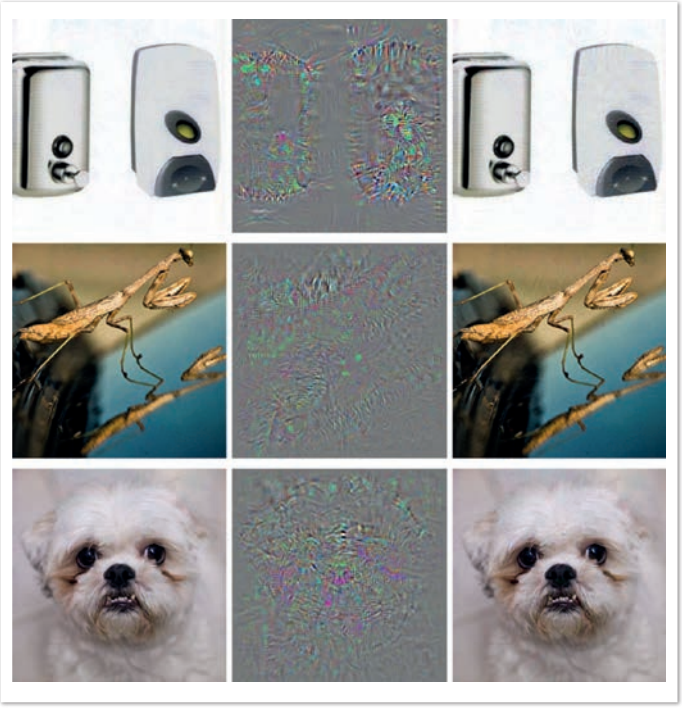


*Abbildung 1: Adversarial Examples (rechte Spalte) werden durch das Hinzufügen von speziellem Rauschen (mittlere Spalte) – hier stark verstärkt – auf korrekt klassifizierte Bilder (linke Spalte) erzeugt. (© Szegedy, Christian, et al. [2])*

*Abbildung 2: Adversarial Patch (© Szegedy, Christian, et al. [2])*

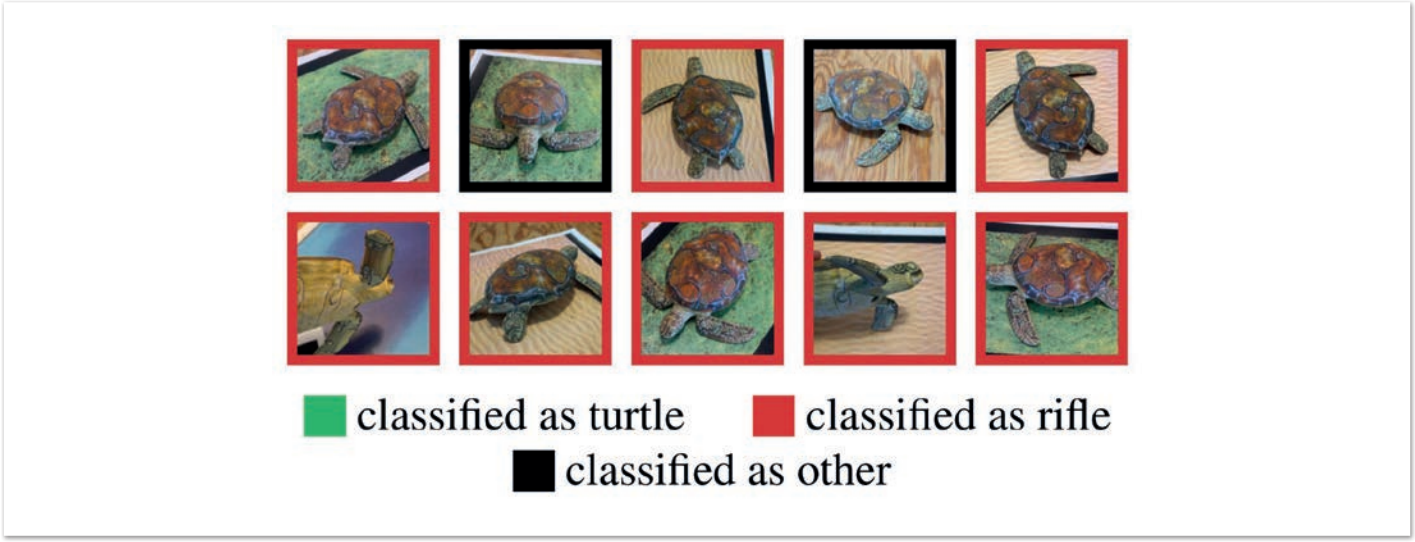



*Abbildung 3: Eine mit einem 3D-Drucker erstellte Schildkröte mit speziell präparierter Textur wird als Waffe erkannt. (© Athalye, Anish, et al. [4])*

führt jedoch dazu, dass AlexNet auf den rechten Bildern jeweils einen Strauß erkennt.

Im Laufe der Zeit wurden die Verfahren zur Erstellung von Adversarial Examples immer weiter verbessert und immer robuster. Konnten die ersten Varianten noch durch einfache Preprocessing-Verfahren unschädlich gemacht werden, überleben heutige Adversarial Examples selbst aufwendige Bildtransformationen. Variationen in der Belichtung, unterschiedliche Ausschnitte eines Bildes und auch JPEG-Kompression können sie mühelos überstehen. Sogar Fotos von ausgedruckten Adversarial Examples können Deep-Learning-Modelle noch täuschen.

Im Laufe der Zeit wurden die Verfahren zur Erstellung von Adversarial Examples immer weiter verbessert und unzählige Varianten entwickelt. So kann zum Beispiel ein einmal berechnetes Adversarial Example unterschiedliche Architekturen täuschen. Außerdem wurden Universal Adversarial Examples entwickelt. Im Unterschied zu den normalen Adversarial Examples, die für jedes Bild neu berechnet werden müssen, um eine bestimmte Fehlklassifizierung zu erzwingen, muss das Rauschen für ein Universal Adversarial Example nur einmal berechnet werden. Dieses eine Rauschen kann dann auf unterschiedliche Bilder angewendet werden, um immer die gleiche Fehlklassifizierung zu erzielen.

Weiterhin wurden Adversarial Patches *[3]* entwickelt *(siehe Abbildung 2)*. Bei einem solchen Patch handelt es sich um einen Sticker, der irgendwo in einer Scene platzieren werden kann. Sobald ein Netzwerk diesen Patch sieht, erkannt das Netzwerk immer einen bestimmten Gegenstand, wie zum Beispiel einen Toaster. Zwar ist so ein Adversarial Patch im Vergleich zum unsichtbaren Rauschen der Adversarial Examples deutlich sichtbar, dadurch aber auch gleichzeitig sehr robust. Durch diese Eigenschaft kann ein solcher Patch in sehr vielen Situationen eingesetzt werden, zum Beispiel auch bei unterschiedlichen Lichtverhältnissen. Zusätzlich hat dieser den Vorteil, dass kein Bild verändert werden muss und dass die Szene, in der der Patch zum Einsatz kommen soll, bei der Erzeugung des Patches nicht bekannt sein muss. Der Angreifer kann den Patch also in Ruhe offline bei sich zuhause in einer sicheren Umgebung berechnen.

Adversarial Examples lassen sich jedoch nicht nur auf Bilder anwenden, sondern auch auf Objekte in unserer physikalischen Welt. Im Jahr 2018 haben Forscher gezeigt *[4]*, dass es möglich ist, die Textur eines dreidimensionalen Objekts minimal zu verändern, sodass dieses Objekt als etwas anderes erkannt wird. Sie haben eine Schildkröte *(siehe Abbildung 3)* mit einem 3D-Drucker erstellt, die ohne Änderungen der Textur von einem neuronalen Netz aus unterschiedlichen Blickrichtungen betrachtet korrekt erkannt wurde. Mit einer gezielten Manipulation der Textur allerdings konnten die Forscher das Netzwerk derart täuschen, dass das Netzwerk bei der Betrachtung der Schildkröte immer eine Waffe erkannt hat. Die Erkennung als Waffe war sogar deutlich zuverlässiger. Die Änderungen an der Textur waren wieder nur sehr schwer erkennbar. Nur bei genauem Hinsehen zeigten sich an einigen Stellen ein paar farbige Artefakte, die auf eine Manipulation hindeuten könnten.

## Abwehrmaßnahmen

Wie man Netzwerke vor Adversarial Examples schützen kann, ist noch immer eine nicht gelöste Frage, obwohl schon seit mehreren Jahren sehr aktiv an dieser Frage geforscht wird. Viele Verfahren wurden vorgeschlagen und für viele Verfahren hat es nicht lange gedauert, bis gezeigt wurde, wie sie wieder umgangen werden können. Bei zwei der am häufigsten untersuchten Verfahren handelt es sich um Denoising- und Adversarial-Training. Beim Denoising werden unter anderem Transformationen auf das Eingabebild angewendet, die zum Beispiel das Rauschen, das dem Originalbild hinzugefügt wurde, wieder entfernen oder zumindest soweit reduzieren, dass das Rauschen keine Auswirkung mehr auf das Modell hat. So etwas könnte zum Beispiel durch eine einfache Rauschunterdrückung erfolgen oder durch Anwendung einer JPEG-Kompression. Beim Adversarial-Training hingegen werden während des Trainings zu den normalen, nicht manipulierten Trainingsdaten zusätzlich Adversarial Examples erstellt und auch diese für das Training verwendet. Das hat zur Folge, dass das Netzwerk Adversarial Examples bereits kennt und weiß, wie diese zu klassifizieren sind. Das führt dann dazu, dass das Netzwerk robuster wird.

## Backdoors

Machine-Learning-Modelle lassen sich jedoch nicht nur bei der Erkennung von Objekten täuschen. Ein anderes Problem, das auch

im Bereich der Softwareentwicklung sehr gut bekannt ist, sind bewusst eingebaute Backdoors. Auch Machine-Learning-Modelle können Backdoors enthalten. Ein Modell mit einer Backdoor verhält sich unter normalen Umständen wie ein Modell ohne Backdoor und erzielt auch Genauigkeiten, die das nicht manipulierte Modell erreichen würde. Enthält eine Eingabe aber einen bestimmten Trigger, der vom Angreifer frei gewählt werden kann, verhält sich das Modell wie vom Angreifer gewünscht. Handelt es sich zum Beispiel um ein Modell, das bei der Erkennung von Bildern verwendet wird, könnte als Trigger etwa ein einfaches Post-it dienen. Wird das Post-it auf ein Objekt angeheftet, wird die Backdoor aktiviert und das Modell erkennt einen vom Angreifer gewünschten Gegenstand.

Gezeigt wurde so etwas unter anderem für Netzwerke, die bei der Erkennung von Straßenschildern zum Einsatz kommen *[5]*. Die Backdoor wurde so erstellt, dass ein Post-it auf einem Straßenschild immer dazu führt, dass ein Tempolimit-Schild erkannt wird. Das kann zur Folge haben, dass ein autonomes Fahrzeug bei einem Stopp-Schild mit einem Post-it nicht mehr anhält, sondern vielleicht sogar beschleunigt und den Fahrgast somit in Lebensgefahr bringt.

Eine Backdoor wird vom Angreifer während des Trainings hinzugefügt. Dies kann zum Beispiel durch manipulierte Trainingsdaten erfolgen, bei denen entweder die Trainingsdaten mit falschen Labels versehen oder die Daten selbst manipuliert werden. Dieses Vorgehen ist auch unter dem Begriff „Data Poisoning" bekannt. Die Daten können wieder so verändert werden, dass die Manipulationen für einen Menschen nicht sichtbar sind. Somit könnte ein Data Scientist, der den Datensatz vor der Verwendung gegebenenfalls prüft, um zum Beispiel falsch gelabelte Daten zu entfernen, keine Manipulation feststellen.

Im Gegensatz zum Quellcode bei Software, den man nach Backdoors relativ leicht durchsuchen kann, ist das bei Machine-Learning-Modellen nicht so einfach möglich. Bei den Modellen handelt es sich in der Regel um eine Blackbox, in die man nicht hineinschauen kann. In den Millionen oder sogar Milliarden Parametern ist die Suche nach einer Backdoor vergleichbar mit der Suche nach einer Nadel im Heuhaufen.

Es gibt viele mögliche Szenarien, in denen ein Angreifer eine Backdoor in ein Modell integrieren kann. Sehr häufig werden zum Beispiel einfach Daten aus dem Internet für die Erstellung von Modellen verwendet. Ein Angreifer könnte deshalb versuchen, manipulierte Daten im Internet so zu platzieren, dass ein Crawler sie irgendwann findet und sie zu einem Trainingsdatensatz hinzufügt.

Ein anderes Risiko stellen Cloud-Dienste dar. Aufgrund des immensen Bedarfs an Ressourcen, der heutzutage für das Erstellen eines State-of-the-Art-Modells notwendig ist, wird das Training zunehmend in die Cloud ausgelagert. Eine Backdoor könnte somit auch über den Cloud-Anbieter in ein Modell integriert werden. Häufig wird auch einfach auf das eigene Training komplett verzichtet. Dann werden zum Beispiel vortrainierte Modelle verwendet, von denen oftmals nicht bekannt ist, wie sie erstellt wurden. Auch solche Modelle können natürlich eine Backdoor enthalten, die selbst dann erhalten bleibt, wenn die mithilfe von Transfer-Learning verändert und an die eigenen Bedürfnisse angepasst werden.

## Parameter extrahieren

Private Unternehmen investieren teilweise viel Geld, um Modelle zu erstellen, die ihnen einen Wettbewerbsvorteil geben könnten. Es wird geschätzt, dass zum Beispiel das Training von GPT-3, einem Sprachmodell mit 175 Milliarden Parametern, zirka fünf Millionen Dollar gekostet hat. Die Gesamtentwicklungskosten inklusive Forschung werden sogar auf zehn bis 20 Millionen Dollar geschätzt.

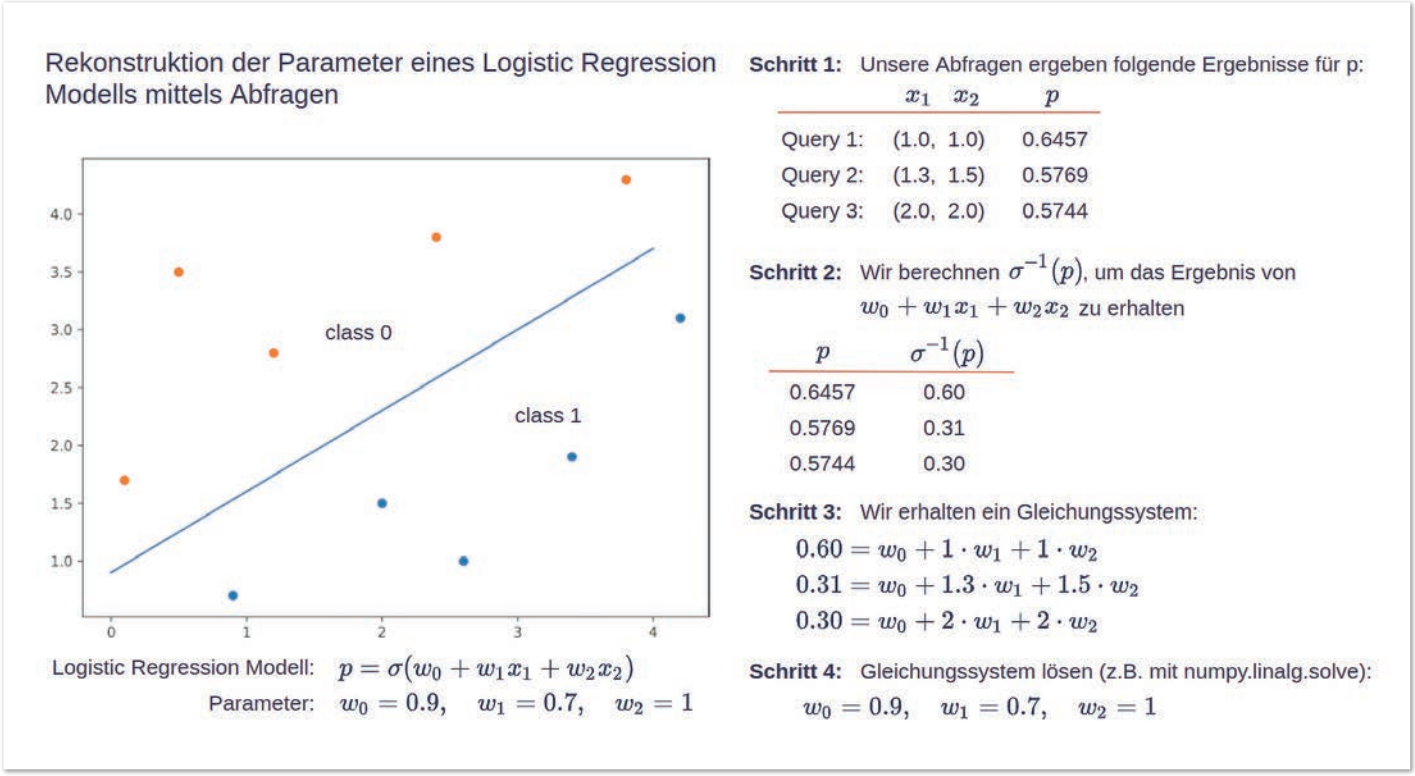



*Abbildung 4: Extrahieren der Parameter eines Logistic-Regression-Modells (© Daniel Etzold)*

Bei derart hohen Investitionen sind Unternehmen natürlich daran interessiert, diese Kosten wieder reinzuholen. Eine Möglichkeit besteht darin, das Modell in der Cloud zu betreiben und einen bezahlten Zugang über ein API anzubieten. Dies ist auch bekannt als Machine-Learning-as-a-Service (MlaaS).

Die Parameter eines solchen Modells stellen somit einen extrem hohen Wert für das Unternehmen dar. Der Zugang muss geschützt werden. Weiterhin darf das Modell keine Informationen in der Antwort auf Queries zurückliefern, die Rückschlüsse auf die Parameter zulassen könnten. Denn: Sollten zu viele Informationen zurückgeliefert werden, können durch geschickte Queries unter Umständen alle Parameter eines Modells extrahiert werden.

Nehmen wir als Beispiel Logistic Regression. Dabei handelt es sich um ein einfaches, binäres Klassifikationsmodell. Ein Beispiel für ein solches Modell ist in *Abbildung 4* zu sehen. Das Modell berechnet für einen Punkt (x1, x2) die Wahrscheinlichkeit, dass der Punkt zu der Klasse 1 gehört. Alle Punkte unterhalb der Geraden (auch als Decision Boundary bezeichnet) haben eine höhere Wahrscheinlichkeit als 50 % und werden deshalb dieser Klasse zugeordnet. Alle Punkte darüber haben eine geringere Wahrscheinlichkeit und werden deshalb der Klasse 0 zugewiesen. Je weiter ein Punkt von der Geraden entfernt ist, desto höher beziehungsweise niedriger ist die Wahrscheinlichkeit. Liegt ein Punkt genau auf der Geraden, liefert uns das Modell als Wahrscheinlichkeit genau 50 %.

Wie in der Abbildung zu sehen ist, besteht das Modell aus den drei Parametern w0, w1 und w2. Diese Parameter sind geheim. Wollen wir sie bestimmen, können wir das nur, indem wir unterschiedliche Punkte von dem Modell klassifizieren lassen und die zurückgelieferten Wahrscheinlichkeiten auswerten. Dabei zeigt sich, dass wir bereits nach nur drei Queries alle notwendigen Informationen haben, die wir zur Berechnung der Parameter benötigen. Denn schon nach drei Queries haben wir ein Gleichungssystem mit drei Gleichungen und drei Unbekannten (die Parameter). Die Lösung eines solchen Gleichungssystems kann nun durch einfaches Umformen eindeutig bestimmt werden. Dazu berechnen wir zunächst die Umkehrfunktion der Sigmoid-Funktion. Anschließend können wir die Parameter durch Lösen des Gleichungssystems berechnen.

Dieses Beispiel ist bewusst einfach gewählt. In der Praxis kommen häufig sehr viel komplexere Modelle zum Einsatz, für die das nicht mehr so einfach wie im Beispiel gezeigt möglich ist. Aber auch für solche Modelle können ähnliche Angriffe durchgeführt werden. Oftmals sind solche Angriffe erfolgreich, weil mehr Informationen in einer Antwort zurückgeliefert werden, als notwendig ist. Viele Modelle liefern zum Beispiel neben der Klassifizierung auch noch die Konfidenz in der Antwort zurück. Würde darauf verzichtet werden, könnten Angriffe bereits deutlich erschwert oder sogar komplett verhindert werden. Häufig reicht es schon aus, nicht den exakten Wert zurückzuliefern, sondern beispielsweise die Werte auf die erste Nachkommastelle zu runden.

## Fazit

Machine-Learning-Modelle können auf unterschiedliche Art und Weise angegriffen werden. Sie können während der Inference getäuscht werden, Backdoors können während des Trainings integriert werden und die Parameter eines Modells können extrahiert werden. Dabei handelt es sich noch nicht um alle möglichen Bedrohungen für ein Machine-Learning-System.

Waren früher die beschriebenen Angriffe eher von akademischer Natur, werden sie inzwischen für real eingesetzte Systeme zunehmend relevanter. Denn wie auch in der Softwareentwicklung werden Angriffe auf Machine Learning weiterentwickelt und mit der Zeit immer besser und raffinierter. Allerdings scheint es so, dass sich Schwachstellen in Machine-Learning-Modellen nicht so einfach wie bei Software durch das Einspielen eines Patches beseitigen lassen. Bei diesen Schwachstellen handelt es sich vielmehr um inhärente Probleme in den zugrunde liegenden Algorithmen. Abwehrmaßnahmen zu entwickeln und Machine-Learning-Modelle robuster zu machen benötigt Zeit. In der Regel vergeht aber deutlich weniger Zeit, bis eine Abwehrmaßnahme wieder umgangen werden kann. Die Angreifer scheinen also bei diesem Katz-und-Maus-Spiel immer einen Schritt voraus zu sein und wir werden wohl noch einige Zeit mit verwundbaren Machine-Learning-Modellen umgehen müssen.

## Quellen


[1] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. (2012): „Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25.

[2] Szegedy, Christian, et al. (2013): „Intriguing properties of neural networks." arXiv preprint arXiv:1312.6199

[3] Brown, Tom B., et al. (2017): „Adversarial patch." arXiv preprint arXiv:1712.09665

[4] Athalye, Anish, et al. (2018): „Synthesizing robust adversarial examples." International conference on machine learning. PMLR

[5] Gu, Tianyu, et al. (2019): „Badnets: Evaluating backdooring attacks on deep neural networks." IEEE Access 7: 47230-47244.



**Daniel Etzold**

*daniel@etzold.io*


Daniel Etzold hat mehr als 10 Jahre Erfahrung im Bereich Security, ist Autor von „Security Flashcards" und beschäftigt sich mit Bedrohungen und Schwachstellen von Machine-Learning-Systemen. Zu diesem Thema schreibt er regelmäßig Artikel und ist auch immer wieder auf Konferenzen anzutreffen.

# Keycloak IAM & SSO erfolgreich (im Unternehmen) einsetzen

Niko Köbler


*Die Anwendungs- und Service-Landschaft in Unternehmen hat sich in den letzten Jahren zunehmend verändert. Mehrere kleine, spezialisierte Anwendungen, verteilte Anwendungen, server- und clientseitige Anwendungen, mobile Anwendungen und so weiter. Alle Anwendungen haben jedoch eines gemeinsam: Sie erfordern eine Zugriffssteuerung hinsichtlich Authentifizierung und Autorisierung. Schließlich soll nicht einfach ein beliebiger Anwender eine beliebige Funktion ausführen können, sondern man will wissen, WER die Anwendung benutzt (Authentifizierung), und dann steuern, WAS dieser Benutzer darf (Autorisierung).*

Früher war dies relativ einfach, man hat im einzigen Unternehmensmonolithen einen Login-Screen eingebaut, dahinter eine Datenbank gelegt, in der die Benutzerdaten, vorrangig Benutzername und Passwort (dieses bitte gehasht), eingetragen wurden.

Ein Benutzer konnte sich somit an der Anwendung anmelden und irgendwo standen zu diesem Benutzer dann auch noch dessen Rechte. So weit, so gut. Oder auch nicht. Schließlich sind AuthN und AuthZ, wie Authentifizierung und Autorisierung gerne abgekürzt werden, sicherheitskritische Themen. Und Sicherheit geht uns Entwickler alle an: Wer sich nicht damit beschäftigt, handelt schon fast fahrlässig. Das ist bis zu einem gewissen Grad sicherlich richtig, dennoch müssen wir Entwickler nicht alle Experten auf diesem Gebiet sein. Die meisten sind sicherlich Experten auf ihrem fachlichen Gebiet und können Geschäftslogik bestens abbilden und in Programmcode wandeln – und das ist gut so. Aber gerade deshalb ist es fragwürdig, ob wir etwas implementieren sollten, über das wir kein Expertenwissen haben. Und Authentifizierung ist ein Thema, das für eine sichere und korrekte Implementierung Expertenwissen benötigt. Schließlich haben wir es hier mit persönlichen Daten zu tun, die nicht zuletzt durch die DSGVO eines besonderen Schutzes bedürfen. Persönliche Daten sollten grundsätzlich als hochsensibel behandelt werden.

Warum also sollten Unternehmen Dinge, über die sie kein bis nur wenig Wissen besitzen, immer wieder selbst implementieren (außer die Führungsebenen leiden am „Not-invented-here-Syndrom" *[1]*, was aber ganz andere Probleme mit sich bringt)!?

Und das dann auch noch für moderne, gewachsene und verteilte Anwendungslandschaften, wie eingangs erwähnt? Also repetitiv die falschen Dinge tun!? Zudem sind dann die Anwender gezwungen, sich immer wieder an den einzelnen Applikationen anzumelden. Wenn es „gut" läuft, dann liegt hinter

jedem Login-Screen das gleiche LDAP oder ActiveDirectory und das Passwort ist zumindest identisch – im „schlechten" Falle muss sich der Anwender allerdings für jede Applikation ein eigenes Passwort merken. Da Passwortmanager leider immer noch rar gesät sind, werden die Post-its rund um die Monitore mit notierten Passwörtern nicht weniger – ein Graus für Security-Experten.

## Single Sign-on

Dabei ist die Lösung so simpel wie nah: Das Stichwort heißt Single Sign-on (SSO); dies ist kein neues Konzept, sondern hat schon einige Jahrzehnte auf dem Buckel.

Nur nutzen es immer noch zu wenige Unternehmen sinnvoll und richtig für ihre eigenen Anwendungslandschaften. Dabei ist SSO nicht kompliziert anzuwenden und zu integrieren. SSO wir von den meisten Personen damit „übersetzt", dass es nur ein Passwort für einen Benutzer gibt oder dass ein Benutzer sich nur einmal anmelden muss, um mit mehreren Anwendungen zu arbeiten. Das ist zwar richtig, umfasst aber nicht die gesamte Bedeutung. Als weiteres, sehr zentrales und wichtiges Kriterium zählt zu einem SSO, dass das Benutzerpasswort nur einem (single!) sicheren System, dem Auth-Server, bekannt ist, nur dieser Auth-Server das Passwort validiert und dass der Benutzer sein Passwort (oder die sonstigen Credentials, Zwei-Faktor etc.) nur diesem einen (single!) Auth-Server bekannt gibt. Diese Anforderung an ein SSO impliziert, dass es nur eine zentrale Anmeldeseite geben kann. Würde ein Benutzer nämlich seine Credentials in der Fachanwendung selbst eingeben und diese leitet sie dann an ein (zentrales) Verzeichnis zur Überprüfung weiter, sprechen wir von einem klassischen Man-in-the-Middle-Szenario *[2]*. Die Fachanwendung ist der Man-in-the-Middle und bekommt Zugriff auf das, wahrscheinlich im Klartext vorliegende, Benutzerpasswort. Nach obiger Definition, dass nur der Auth-Server die Credentials kennt und nur an diesem die Credentials eingegeben werden sollen (dürfen!), widerspricht eine lokale Login-Seite dem SSO-Konzept.

Die Lösung für die Herausforderung mit einem SSO heißt Tokens. Der Benutzer meldet sich am Auth-Server an, dieser validiert die Credentials und stellt dann für die beteiligten Anwendungen Tokens aus. In einem Token können dann alle für die Anwendung interessanten Daten stehen. Mit SAML *[3]* steht schon seit über 20 Jahren eine Token-basierte Lösung bereit. SAML mag passabel bis gut in unternehmensinternen Anwendungen eingesetzt werden können, bietet jedoch hinsichtlich moderner Kommunikationsprotokolle und Kompatibilitäten über System- und Programmiersprachen hinweg einige Nachteile. Auch aus diesem Grund sind mit OAuth2 *[4]* und OpenID-Connect (OIDC, basiert technisch auf den Flows von OAuth2, bringt aber noch standardisierte Identitätsinformationen mit *[5]*) vor einigen Jahren zwei neue Protokolle entstanden, die die aktuellen Heraus- und Anforderungen lösen wollen (und dies größtenteils auch gut können). Es stehen also viele Möglichkeiten zur Verfügung.

Da das OAuth2-Protokoll primär für die Autorisierung zwischen verteilten und durchaus auch fremden Anwendungen, also Anwendungen zweier oder mehrerer verschiedener Parteien/Unternehmen, über das öffentliche Internet konzipiert wurde, haben sich schnell Serviceanbieter gefunden, die OAuth2- und OIDC-basierte Authentifizierung und Autorisierungen gepaart mit einer Identitätsverwaltung für Benutzer als Dienstleistung anbieten. Leider haben die meisten dieser Services jedoch den Nachteil, dass man (das Unternehmen) die Benutzerdaten – und damit schützenswerte persönliche Daten! – an den Serviceanbieter auslagern und abgeben muss. Egal, ob dieser Serviceanbieter in Europa sitzt oder in den USA: Ein Dritter bekommt Zugriff auf die Daten des eigenen Unternehmens und deren Benutzer oder Kunden. Eine Entscheidung, die sehr gut überlegt sein will. Auch, wenn Passwörter ausreichend gut verschlüsselt dort abgelegt sind, sind dennoch nicht alle Daten verschlüsselt und unbrauchbar für den Serviceanbieter. Eine selbstgehostete Lösung muss also her, die ein Unternehmen im eigenen Rechenzentrum oder in der selbst verwalteten Cloud betreiben kann und so die Hoheit der gespeicherten Daten behält. Wir erinnern uns an weiter oben in diesem Artikel: Wir sind allerdings keine Auth(N/Z)-Experten und wollen das System natürlich nicht selbst entwickeln und warten.

## Keycloak als (Unternehmens-)Lösung

Mit Keycloak *[6]* steht ein solches SSO- und IAM(Identity und Access Management)-System zur Verfügung, das die gebräuchlichsten Protokolle wie SAMLv2 und OAuth2/OIDC abdeckt, eine Benutzerverwaltung mitbringt und selbst gehostet werden kann (muss). Keycloak kommt aus dem Hause Red Hat, ist bereits zirka acht Jahre alt und in Java implementiert. Basierten die ersten Versionen auf dem WildFly Application Server, hat Keycloak gerade mit Version 17 eine

Migration auf die Quarkus-Plattform hinter sich, um auch für die nächsten Jahre und zukünftige Anforderungen hinsichtlich Erweiterbarkeit, neuer Features, Performance und Stabilität gewappnet zu sein. Für Betrieb, Administration und Entwickler gibt es umfangreiche Dokumentationen *[7]*.

Keycloak macht viele Dinge gut und bringt viele Features von Haus aus mit. Dennoch, es ist – wie man so schön zu sagen pflegt – „opinionated". Es muss nicht jedem gefallen, wie Keycloak Dinge macht oder abbildet. Gerade im Authentifizierungsumfeld hat jedes Unternehmen eine andere Sicht auf die Welt (und ist meist davon überzeugt, dass die eigene die einzig richtige Sicht ist) und Keycloak hat eben auch eine eigene.

Viel wichtiger aber ist die Tatsache, dass Keycloak auf der Java-SPI-Architektur *[8]* basiert und somit eine perfekte Plattform ist, um die eigenen Anforderungen mit möglichst wenig Aufwand umzusetzen, zu implementieren und zur eigenen Keycloak-Umgebung hinzuzufügen, ohne den eigentlichen Kern von Keycloak zu verändern und somit jederzeit update- und upgradefähig zu bleiben. Fast jedes Feature in Keycloak ist als Service-Provider-Interface ausgelegt, mit einem oder mehreren Default-Provider-Implementierungen, um die Basis-Funktionalität abzubilden. Reicht das nicht aus, kann über eine Custom-SPI-Implementierung das gewünschte Verhalten einfach hinzugefügt und damit Keycloak erweitert werden.

## User Federation

Unter User Federation versteht man, dass Keycloak bei der Verwaltung von Benutzerdaten mit einer externen Datenquelle zusammenarbeitet, also föderiert.

In dieser externen Datenquelle können die kompletten Benutzerdaten gespeichert beziehungsweise verwaltet werden, ohne dass Keycloak selbst Daten dazu hält (außer notwendige Metadaten, die aber keine persönlichen Daten enthalten). Bei Bedarf kann Keycloak jedoch auch weitere Daten zum Benutzer im eigenen Storage ablegen.

Ein Beispiel für eine User Federation ist ein im Unternehmen gegebenenfalls existierendes LDAP-Verzeichnis oder AD (mit LDAP-Protokoll). Dies kann als Quelle für die Benutzer einfach konfiguriert werden und die Benutzer müssen nicht redundant in Keycloak verwaltet werden. Natürlich können die Benutzer dann auch ihr existierendes LDAP-Passwort weiterverwenden. Als Default-Implementierung des User-Storage-SPI steht es direkt zur Verfügung, kann einfach konfiguriert werden und bedarf keines Programmieraufwands.

Wenn allerdings eine (beliebige) andere Quelle, ganz gleich ob Datenbank, ein Service über ein API etc., als User Federation genutzt werden soll, können über das User-Storage-SPI eine eigene Implementierung bereitgestellt und die Daten aus der externen Quelle gelesen und bei Änderungen auch wieder dorthin zurückgeschrieben werden. Für Keycloak selbst macht es keinen (großen) Unterschied, ob die Benutzerdaten aus der eigenen Datenhaltung oder aus einer externen Quelle stammen. Keycloak operiert ja nur über die Interfaces und auch der interne Storage ist eine (spezielle) Implementierung des User-Storage-SPI. Ein Beispiel für eine Read-only-Implementierung eines User Storage über einen externen REST-Service kann im Beispiel-Repository `dasniko/keycloak-extensions-demo` *[9]* gefunden werden.

## Externer Identity Provider

Zu beachten ist, dass eine User Federation nicht gleich einem externen Identity Provider (IdP) ist. Ein externer IdP überprüft die Identität eines Benutzers selbstständig und mit eigenen Prozessen. Das Wissen, ob ein Benutzer auch wirklich der ist, der er vorgibt zu sein, hat nur der IdP. Keycloak vertraut an dieser Stelle dem IdP und übernimmt die von diesem zu dem Benutzer zur Verfügung gestellten Daten.

In Keycloak können als externe IdPs per Konfiguration eine ganze Menge Social Provider (beispielsweise Google, Twitter, Facebook, Microsoft, GitHub etc.) eingerichtet werden. Auch die Standard-Protokolle OIDC und SAML stehen als generische Adapter zur Verfügung. Möchte man einen Provider anbinden, der keinem der Standardprotokolle folgt und nicht bereits als Implementierung vorliegt, kann man dies über das Identity-Provider-SPI gemäß den Vorgaben und Protokollen des IdP selbst zur Verfügung stellen und in Keycloak deployen.

## Authenticator

Die Authentifikatoren sind eines der mächtigsten Interfaces und Werkzeuge in Keycloak. Sie steuern letztendlich, wie sich ein Benutzer authentifiziert und welche Daten er für die erfolgreiche Überprüfung zur Verfügung stellen muss. Im Standardfall sind das Benutzername und Passwort. Dann können noch beliebige Multi-Faktoren hinzukommen oder aufgrund von anderen Gegebenheiten die Authentifizierung durchgeführt werden. Für das typische SSO-Verhalten ist beispielsweise ein Cookie-Authentifikator vorhanden, der ein zuvor (von Keycloak selbst!) gesetztes Cookie auswertet und auf dieser Basis einen Benutzer wiedererkennt.

Für die Zwei- oder Multi-Faktor-Authentifizierung (2FA/MFA) ist bereits ein (T)OTP*[10]*-Authentifikator inklusive benötigter Eingabeformulare zur Konfiguration und Verifizierung vorhanden. Möchte man eine andere Basis für eine MFA verwenden, etwa SMS oder E-Mail (ohne jetzt auf die Vor- und Nachteile und Sicherheitsbedenken dieser Ansätze einzugehen) oder ein Hardware-Token, implementiert man eben das Authentication-SPI mit dem gewünschten Verhalten.

Auch bedingte Authentifikatoren sind möglich, die also nur aufgrund einer bestimmten, erfüllten oder auch nicht erfüllten Bedingung ausgeführt werden (müssen). So kann beispielsweise ein TOTP-Verfahren nur für Benutzer mit einer Admin-Rolle konfiguriert werden. Auch die bedingten Authentifikatoren sind, man erwartet es bereits, beliebig implementierbar, wenn die mitgelieferten Varianten nicht ausreichen.

Hat man alle Authentifikatoren als Bausteine vorliegen, egal ob mitgeliefert oder selbst implementiert, werden diese in den Authentication-Flows konfiguriert.

Hier wird festgelegt, in welcher Reihenfolge die Authentifikatoren ausgeführt oder aufgerufen werden, ob diese Bedingungen haben (conditional), wenn ja, welche, ob ein Authentifikator erforderlich (required) ist oder optional (optional) ausgeführt werden kann.

Ein Beispiel für eine eigene 2FA-SMS-Lösung ist unter `dasniko/dasniko/keycloak-2fasms-authenticator` *[11]* (inkl. Video) zu finden.

## EmailTemplate und Forms

Keycloak liefert natürlich alle benötigten Screens und Eingabeformulare mit aus. Hierfür gibt es auch ein Default-Styling, basierend auf der PatternFly-UI-Bibliothek. In den allermeisten Fällen entspricht dies nicht den Wünschen oder CI-Vorgaben der Unternehmen, die Keycloak einsetzen.

Das Theming der Oberflächen ist flexibel gestaltet und kann beliebig angepasst werden. Als Templating-Engine wird FreeMarker *[12]* eingesetzt, eine ausgereifte und mächtige Template-Engine, die zwar schon einige Jahre auf dem Buckel hat, aber dafür auch zuverlässig arbeitet. Möchte man die Templates auf einer anderen technologischen Basis (etwa Thymeleaf, Mustache etc.) rendern, kann das über verschiedene Forms-SPIs und das EmailTemplate-SPI geschehen.

Die einfachste Option, das Look-and-Feel der Oberflächen anzupassen, ist über ein selbsterstelltes Theme, das nur angepasste Ressourcen (CSS, Fonts, Bilder etc.) bereitstellt. Durch die Template-Vererbung können immer die kompletten Ressourcen des Basis-Theme genutzt werden und es muss nur das überschrieben werden, was man anpassen möchte. So können etwa die HTML-Templates aus der Basis verwendet und es müssen nur die eigenen CSS-Layouts deployt werden. CSS-Klassennamen sind über Variablen umfangreich anpassbar.

Auch die Gestaltung der E-Mails, die Keycloak versendet, kann komplett kontrolliert und angepasst werden. Hier ist darauf zu achten, dass Keycloak E-Mails immer im HTML- und Text-Format sendet, es müssen also zwei Varianten der E-Mail-Templates verwaltet werden.

## EventListener

Für fast alle Vorgänge in Keycloak werden Events emittiert. Events sind beispielsweise Logins, Logouts, Profil-Updates, Passwort-Änderungen und so weiter. Aber auch Admin-Events, für alle Konfigurationsänderungen innerhalb von Keycloak, werden ausgesendet. Für jedes Event gibt es ein Erfolgs- und ein Fehlerevent. Events können in der eigenen Datenbank gespeichert werden, dies muss jedoch explizit aktiviert werden. Im Standard laufen die Events einfach in die Log-Ausgabe des Servers. Erfolgsevents mit DEBUG-Level, Fehler mit WARN.

Die Verwaltung von Events innerhalb von Keycloak ist recht einfach implementiert und stellt keine großen Möglichkeiten bereit. Deshalb implementieren viele Keycloak-Anwender als erstes SPI das EventsListener-SPI, das auch das am einfachsten zu verwendende SPI ist. Beispiele zu SPI-Implementierungen nutzen fast alle das EventListener-SPI als das „Hello World"-Beispiel im Keycloak-Ökosystem.

Die einfachste und meist auch sinnvoll(st)e Implementierung schickt einfach alle Events auf einen externen Message Broker oder in ein Storage-System, in dem die Events umfangreich und sinnvoll ausgewertet werden können. Eine einfache EventListener-SPI-Implementierung, die alle Events auf einem AWS-SNS-Topic veröffentlicht, findet sich im Beispiel-Repository *[9]*.

## REST-Ressourcen

Im OAuth2-/OIDC-Protokoll gibt es kein REST-API, über das man sich authentifiziert. Die Hoffnung und der Irrglaube hält sich leider stetig in den Köpfen vieler Entwickler und Architekten. Es gibt zwar den Resource Owner Password Credential Grant (ROPC), dieser ist allerdings im OAuth-2.0-Security-Best-Current-Practice-Dokument *[13]* mittlerweile mit „MUST NOT be used (Section 2.4)" angegeben.

Das in Keycloak enthaltene REST-API ist ein ausschließlich für Admin-Zwecke vorgesehenes API. So nutzt die Admin-UI dieses API, aber auch die Java `keycloak-admin`-Bibliothek und die Kommandozeilenversion `kcadm.sh` greifen auf dieses Admin-REST-API zurück. Dieses API für Authentifizierungszwecke zu missbrauchen, ist aus mehreren Gründen schon nicht mehr mit Fahrlässigkeit zu bewerten.

Reicht dieses API nicht aus oder sollen bewusst bestimmte und zusätzliche/neue (REST-)Endpunkte zur Verfügung gestellt werden, kann dies über die Implementierung des Realm-Resource-SPI geschehen. Hierüber können mittels JAX-RS beliebige Endpunkte implementiert werden, die dann in jedem Realm in Keycloak als Pfad zur Verfügung stehen.

Vorsicht ist geboten, was die Absicherung (Authentifizierung/Autorisierung) der Endpunkte betrifft. Realm-Ressourcen sind standardmäßig offen für jeden Zugriff! Leider stehen keine einfach nutzbaren Annotationen zur Verfügung, mit denen man einen Endpunkt beispielsweise nur für bestimmte Rollen zugreifbar machen kann. Auth(N/Z) kann nur über eine eigene Implementierung der gewünschten Logik beziehungsweise des gewünschten Verhaltens erreicht werden. Vielleicht ist das auch gut so, denn für Endpunkte im Auth-Server sind oftmals komplexere und je nach Anwender/Unternehmen völlig verschiedene Ansätze der Zugriffssteuerung gewünscht oder notwendig. Keycloak stellt hierfür ein paar Hilfsklassen und -methoden zur Verfügung, damit man nicht alles selbst machen muss und einem bestimmte repetitive Dinge, wie das Auslesen der Token-Infos aus dem Request oder aus einem Cookie etc., abgenommen werden.

Ein einfaches Beispiel für eine Realm-Ressource mit öffentlichem und abgesichertem Endpunkt ist im Beispiel-Repository *[9]* zu finden.

## Was noch?

Die oben beschriebenen SPIs stellen nur eine sehr kleine Auswahl der SPIs in Keycloak dar. So gibt es SPIs für (auch diese Aufzählung ist nicht abschließend) die Account-Verwaltung, Actiontokens, RequiredActions, verschiedene Verschlüsselungs- und Hashing-Algorithmen, Policies aller Art, E-Mail-Sender, Credentials, Passwörter, Daten-Mapper, Validatoren und viele mehr. Aus meiner Keycloak-Erfahrung der letzten sieben Jahren kann ich aus voller Überzeugung sagen: Jedes(!) Unternehmen hat andere Vorstellungen davon, wie eine sinnvolle und sichere Authentifizierung und das damit zusammenhängende Identity-Management ablaufen soll. Mit Keycloak jedoch bietet sich eine gute Grundlage und Plattform, fast alle Vorstellungen, und seien sie noch so merkwürdig, auf diesem Gebiet zu verwirklichen und eine Implementierung zu bieten. Die Defaults in Keycloak sind bestimmt nicht so, wie alle Unternehmen es sich wünschen, und ich höre oft die Aussage „*Warum macht Keycloak das so und so? Das ist noch nicht sinnvoll und anders ist es doch viel besser!*" Das kann man so sehen, muss man aber nicht.

Keycloak verfolgt einen eigenen Ansatz, so wie viele Unternehmen auch. Jedoch bietet Keycloak hier den Vorteil, sich anpassen zu können. Und das über eine sinnvollen SPI-Mechanismus und, man muss es betonen, mit nur sehr wenigen nicht-kompatiblen Änderungen in den zur Verfügung gestellten Interfaces. Der Anpassungs- und Wartungsaufwand bei Versionsänderungen hält sich also in Grenzen. Mit den SPIs erhält man volle Flexibilität bei einer guten Entkopplung vom Anwendungskern, bleibt so auch mit vielen Erweiterungen jederzeit upgradefähig und gibt zudem die eigenen Benutzer- und Kundendaten nicht aus der Hand. Wer heute noch selbst versucht, Authentifizierung und Identitätsmanagement zu implementieren oder seine Daten außer Haus gibt, sollte sich auf jeden Fall Keycloak anschauen und testen. *Authentication & IAM done right!*

**Niko Köbler**
*info@n-k.de*

Niko Köbler ist Freelancer und seit über sieben Jahren als Keycloak-Experte für die unterschiedlichsten Kunden europaweit tätig. Auf YouTube betreibt er seit Anfang 2021 einen erfolgreichen Keycloak-Channel und unterstützt die Community in verschiedenen Foren. Außerdem ist er Co-Lead der JUG Darmstadt, Sprecher auf Konferenzen und schreibt als Autor Artikel für verschiedene Fachzeitschriften.

Website: http://keycloak-experte.de
YouTube: https://www.youtube.com/c/NikoKöbler
Twitter: @dasniko

# Ethik im Mittelpunkt von Cybersicherheitspraktiken

**Corinna Wiedenmann**

*Der Schutz vor Missbrauch durch unkontrollierte Zugriffe, das Aufheben von Informationsungleichgewicht im digitalen Raum, wo persönliche Daten eine neue Währung geworden sind, Prävention von Diskriminierung sowie die Wahrung von Autonomie und Selbstbestimmung der eigenen Person – all das sind wichtige ethische Grundsätze für Cybersicherheit. Warum Ethik im Mittelpunkt von Cybersicherheitspraktiken stehen sollte, welche Anknüpfungspunkte ethische Theorien liefern und wie konkrete Orientierungshilfen aussehen können, möchte ich in diesem Artikel etwas näher betrachten. Anschließend werde ich einen Ausblick auf politische Entwicklungen in Europa geben, um die Dynamik des Themas aufzuzeigen und zu begründen, warum Ethik bei der Weiterentwicklung rechtlicher Rahmenwerke stets Berücksichtigung finden sollte.*

International nimmt der Konsens über die Bedeutung von Ethik für technische Anwendungen kontinuierlich zu. Die Gefahren, insbesondere im digitalen Raum, sowohl für Individuen und Unternehmen wie auch für internationale Institutionen und Staaten, wachsen. Die gestiegene Komplexität und Multivariabilität erfordern deshalb von allen Beteiligten und Betroffenen nicht nur die Einhaltung bestimmter rechtlicher Regelwerke, sondern eine ethische Grundhaltung.

Gemäß dem jährlich publizierten Statusbericht der Europäischen Agentur für Cybersicherheit (ENISA) werden die größten Risiken aktuell ausgelöst durch: Ransomware, Malware, Cryptojacking, E-Mail-Bedrohungen, Datenschutzverletzungen und -missbrauch, Desinformation, Lieferkettenangriffe und nicht-bösartige Bedrohungen. Die 2004 gegründete Agentur unterstützt die Umsetzung der europäischen Cyberpolitik und arbeitet mit den Mitgliedsstaaten und EU-Institutionen zusammen an der Verbesserung der Cybersicherheit durch Zertifizierungssysteme und Awareness-Kampagnen *[1]*. Sie ist eine von mehreren internationalen Institutionen, um sich den dynamischen Anforderungen dieses Bereiches zu widmen. Denn die zunehmende Vernetzung von mobilen Geräten, smarter Infrastruktur – sowohl im eigenen Zuhause wie auch im Straßenverkehr – multiplizieren die Einfallstore zum Missbrauch von Daten und Systemen durch böswillige Akteure um ein Vielfaches. Dies wiederum erhöht die ethische Verantwortung von Cybersicherheitsexperten zum Schutz anderer und bedarf einer ethischen Sensibilisierung jedes einzelnen Akteurs *[2]*.

## Ethik als Notwendigkeit

Im Folgenden möchte ich zunächst erläutern, warum es nicht ausreicht, den genannten Gefahren mit Compliance-Prozessen und Sicherheitsanforderungen für Applikationen und Akteure zu begegnen. Ethik zu berücksichtigen ist nötig, um den Schutz vor Missbrauch zu reduzieren. Unkontrollierter Zugriff, beispielsweise auf ein Benutzerkonto, und die damit einhergehenden Vollmachten können weitreichende Folgen sowohl für den direkt Betroffenen als auch für sein Netzwerk haben. Der Schaden kann von Irreführung über Raub bis hin zu physischen Komplikationen reichen.

Eine weitere Begründung liefert die kontinuierliche Stärkung der Relevanz persönlicher Daten als „Währung". Daten sind das neue Gold, hieß es schon von manch einem CEO. Oft haben Nutzer jedoch keine Vorstellung über den Wert ihrer Daten, das heißt darüber, welchen Mehrwert analysierende Parteien daraus ziehen können. Damit geht einher, dass sie oft nicht überprüfen können, was tatsächlich mit den zur Verfügung gestellten Daten gemacht wird. Der Ruf nach mehr Verbraucherschutzrechten, insbesondere in Bezug auf personalisierte, sensible Daten, wurde daher zuletzt immer lauter. Nur so kann mehr Fairness in diesem Verhältnis des Informationsungleichgewichts erzeugt werden.

Schutz vor Diskriminierung ist ebenfalls in die Liste der Gründe für ethische Cybersicherheitspraxen aufzunehmen. Das Risiko, dass erhobene Daten in Kontexten verwendet werden, in denen sie zu Diskriminierung und Benachteiligung führen, besteht ab dem Zeitpunkt der zugestimmten Erfassung. Selbst, wenn der Nutzer zustimmt, dass seine Informationen in einem bestimmten Bereich (beispielsweise Gesundheitsdaten bei Krankenkassen) genutzt werden, kann nicht davon ausgegangen werden, dass er der Übertragung an weitere Stellen zustimmt. Insbesondere wenn diese in einem anderen Kontext Probleme für den Nutzer erzeugen können, wie beispielsweise den Ausschluss von bestimmten beruflichen Möglichkeiten aufgrund gesundheitlicher Konditionen. Philosophen wie Helen Nissenbaum fordern daher kontextuelle Integrität bei der Nutzung und Auswertung von Daten. Die Relevanz lässt sich vor allem aus der zunehmenden Sammlung, Speicherung, Bearbeitung und Weitergabe von personenbezogenen Daten ableiten, die den Wert der Privatsphäre häufig unberücksichtigt lassen. Dies führt auch häufig dazu, dass Autonomie über die Kontrolle der eigenen Daten unterminiert wird. Nissenbaum sowie weitere Wissenschaftler wiederholen daher in verschiedenen Beiträgen konstruktiver Kritik die Notwendigkeit eines formalen Rahmens für die Formulierung von Datenschutzbestimmungen und -praktiken, die sich an kontextueller Integrität orientieren *[3]*.

Als ein letzter Grund in dieser ersten Ausführung für die Notwendigkeit von Ethik im Mittelpunkt muss das Risiko des Verlusts der moralischen Autonomie und der Menschenwürde genannt werden. Sobald Privatsphäre nicht mehr geschützt wird, kann es zu Manipulation von Akteuren kommen. Sowohl indirekte als auch direkte Beeinflussung kann so subtil durch das Reduzieren von Privatsphäre und Räumen der Sicherheit erzeugt werden. Dies führt zu Entscheidungen, die sonst nicht getroffen würden. Beispielsweise zeigt sich, dass Massenüberwachung und die Kenntnis darüber zu Verhaltensänderungen führt und Autonomie raubt, was wiederum eine Verletzung der Menschenwürde darstellt.

## Unterschiedliche Anknüpfungspunkte

Um die ethischen Anknüpfungspunkte und Cybersicherheitspflichten besser einordnen zu können, hilft zunächst eine Einführung in drei ethische Perspektiven: die tugendgeleitete Perspektive, die nutzenorientierte Perspektive und die prinzipienbasierte Perspektive. Aus diesen lässt sich ein Set an Empfehlungen ableiten, die in der Praxis bereits erfolgreich umgesetzt werden, um Cybersicherheit zu stärken.

## Tugendethik

Die grundlegende Idee und Hauptfragestellung ist die Exploration des sittlich richtigen Handelns. Als Richtungsweisung wird hier das tugendhafte Leben in den Mittelpunkt gestellt. Bis in die frühe Neuzeit war dies der dominierende Ethiktyp der abendländischen Philosophie *[4]*. Leitfragen, die sich aus Perspektive der Tugendorientierung stellen, sind:

- Was ist meine grundlegende moralische Verpflichtung als Individuum, Gruppe oder Unternehmen? Und damit einhergehend auch: Halte ich mich konsequent und bewusst daran und bemühe mich um die Weiterentwicklung eines moralischen Charakters?
- Wer sind meine Vorbilder und versuche ich, mich nach moralischen Verhaltensnormen auszurichten?
- Inwieweit verstehe ich es als lebenslange praktische Verantwortung, ein moralisches Urteilsvermögen zu kultivieren?

## Ethik des Nutzens oder auch konsequentialistische Perspektive

Dieser Ethiktyp baut auf moralische Entscheidungen, die aufgrund ihrer Konsequenzen beurteilt und getroffen werden. Anders als die Tugendethik wird der Blick nicht auf den tugendhaften Charakter gerichtet, sondern auf die Abwägung der potenziellen Folgen. Dies mag zunächst einfach erscheinen, jedoch ergibt sich im Fall des klassischen Utilitarismus, der die bekannteste Richtung der kon-

sequentialistischen Theorien darstellt, eine komplexe Kombination über moralische Richtigkeit und Pflicht [5]. Fragen, die sich aus Perspektive der Nutzenethik stellen, sind:

- Was führt zu dem größten Wohl und der höchsten Sicherheit für alle Beteiligten oder Betroffenen?
- Welche potenziellen Folgen und Risikoszenarien muss ich vor einer Entscheidung analysieren und entsprechend in den Entscheidungsfindungsprozess mit einbeziehen?
- Wie stelle ich sicher, dass ich den Nutzen nicht aus einer verzerrten oder egoistischen Perspektive bestimme?

## Prinzipiengeleitete Ethik oder deontologische Perspektive

Das Fundament dieses Ansatzes basiert auf einem regel- beziehungsweise prinzipiengeleiteten Ethiksystem. Sie sind vorgegeben und müssen erkannt werden, ändern sich jedoch nicht aufgrund geänderter Umstände oder Zeiten. Beispiele hierfür lassen sich in vielen Religionen finden, beispielsweise liefern die Zehn Gebote in der christlichen Ethik die Definition eines Wertesystems. Auch die Menschenrechte, die als gesetzt angesehen werden, lassen sich zu dieser Art von Wertesystem zählen [6]. Leitfragen, die sich daraus ergeben, sind:

- Was sind die priorisierten Prinzipien und Werte, denen ich folge? Gelten diese unabhängig von den Umständen?
- Wie überprüfe ich den Einklang meiner Entscheidungen und Handlungen mit den Prinzipien, nach denen ich mich orientieren möchte? Gibt es eine Art des Monitorings und hält mich jemand in der Verantwortlichkeit?

## Sechs ausgewählte praktische Orientierungshilfen

Diese zunächst nur theoretisch anmutenden Ansätze liefern praktische Orientierungshilfen im Alltag. Die Europäische Agentur für Cybersicherheit hat – neben einigen anderen Institutionen – am 14. Februar 2022 eine Sammlung bewährter Verfahren für Cybersicherheit öffentlicher und privater Organisationen in der EU herausgegeben [8]. Weitere Tipps und Tricks wurden auch von dem World Economic Forum in einer Publikation von 2020 in einem Rahmenwerk zu Ethics-by-Design herausgegeben [9]. Aufgrund der Vielzahl der Empfehlungen werde ich einige Ausgewählte, die sich in der Praxis bewährten, gebündelt und in Verbindung mit den vorgestellten Theorien im Folgenden in sechs Abschnitten vorstellen:

## 1) Sensibilisierung für das Thema und Einführung einer „menschlichen" Perspektive auf Daten

Der erste wichtige Praxistipp ist, breite Aufmerksamkeit für das Thema zu erzeugen. Nicht nur Spezialisten in den IT-Abteilungen, sondern auch die Anwender aus anderen Abteilungen, sollten von den Grundzügen der Cybersicherheitsethik Kenntnis haben. Da das Thema immersiv in alle Lebensbereiche stößt und umfangreiche Schäden für Individuen, Gruppen und Unternehmen mit sich bringen kann, ist mehr als Compliance-Konformität, also Pflichterfüllung, gefordert. Wenn Anwender wissen, welche Folgen ihr unbedachter Umgang mit beispielsweise starkem Passwortschutz haben können und dass davon nicht nur sie selbst, sondern ein ganzes Netzwerk an Nutzern bedroht werden kann, wird die Verantwortung sichtbar.

Ein weiterer, damit in Zusammenhang stehender Aspekt ist die Notwendigkeit, Daten in Kontext zu stellen und zu verdeutlichen, dass es sich nicht nur um abstrakte „Units" handelt. Wichtig ist, die Bedeutung und Vernetzung mit dem eigenen Leben transparent zu machen – die Daten sozusagen zu „vermenschlichen". Denn sie berühren oft sensibelste Aspekte einer Person: ihren Ruf, ihr Eigentum sowie Freiheiten und soziale Beziehungen. Sobald diese Zusammenhänge besser verstanden werden, steigt das Verantwortungsbewusstsein jedes Einzelnen. Führen Sie daher idealerweise regelmäßig Schulungen durch, um die Kenntnis über Sicherheitsrichtlinien aktuell zu halten.

## 2) Kenntnis über Risikoszenarien

Risiken von Cyberangriffen sind oft nicht transparent. Was passieren kann, wenn sensible Informationen in „die falschen Hände" geraten, ist für viele unbekanntes Territorium. Man kann sich vielleicht noch vorstellen, dass der Verlust eines Passworts dazu führt, dass dadurch Manipulationen oder der Klau von Eigentum erfolgen können. Welche Folgeschäden jedoch auch bei den mit diesem Profil in Verbindung stehenden Netzwerken oder Nutzern entstehen kann, entzieht sich oft dem Risikobewusstsein. Bei vielen ist das Bewusstsein über Worst-Case-Szenarien niedrig, was auch darauf zurückzuführen ist, dass häufig Intransparenz darüber herrscht, wie viele Prozessschritte einer Datenerhebung eigentlich folgen (können).

Wenn Nutzer wüssten, wie viele Weitergaben und Auswertungen ihre Daten nach der Eingabe durchlaufen und wie hoch das jeweilige Risiko bei der Weitergabe ist, käme es gewiss zu einer reflektierteren Abwägung, welche Informationen wo und mit wem geteilt werden können und welche Auflagen man von den Empfängern der Informationen verlangen möchte. Hier bewährte es sich in der Praxis, regelmäßig Veranstaltungen zur Sensibilisierung für das Thema den unterschiedlichen Zielgruppen anzubieten.

## 3) Realistische Einschätzung von nicht-technischen Nutzern und ihren Interessen

Ein weiterer wichtiger Aspekt ist das Überwinden der eigenen „Scheuklappen". Sicherheitsspezialisten neigen aufgrund ihrer eigenen Expertise leicht dazu, das Wissen und die Sensibilisierung von nicht-technischen Nutzern zu überschätzen. Beispielsweise wissen Spezialisten, dass Antivirus-Software nur ein begrenztes Mittel und kein Allheilmittel darstellt, das weitere Sicherheitsmaßnahmen überflüssig macht. Unerfahrene Nutzer hingegen legen hier meist andere Annahmen zugrunde, was leicht zu fahrlässigem Umgang in der Nutzung verleitet. Daher ist es enorm wichtig, bei der Abwägung und dem Etablieren von Cybersicherheitsstandards unterschiedliche Nutzer zu berücksichtigen und Prozesse auch mit unerwarteten Fehlerquellen zu konfrontieren.

Darüber hinaus helfen hier die Etablierung von Multi-Faktor-Authentifizierungen sowie das Einrichten bestimmter Auflagen, beispielswese für Passwörter, um nicht-technischen Nutzern praktische Unterstützung beim Schutz ihrer eigenen Daten zu geben. Einige Unternehmen führten sogar einen unternehmenseigenen Passwortmanager ein, um eine starke Zugangssicherung für die Nutzer zu ermöglichen.

## 4) Verkettung von ethischer Verantwortung und Rechenschaftspflicht

Auf der einen Seite ist die klare Zuweisung von Verantwortung ein wichtiger Schritt, um Unklarheiten zu beseitigen. Zugleich sollte

man darauf achten, dass die Zuteilung nicht dazu führt, dass mit der eigenen Zuständigkeit das Gefühl der Verantwortlichkeit endet. Wichtig ist, dass jeder einzelne, sowohl Nutzer als auch Entwickler oder Tester, einen End-to-End-Blick vermittelt bekommt und für mögliche Risiken entlang des gesamten Zyklus sensibilisiert wird. Das Bewusstsein darüber, welche potenziellen Folgen und Risikoszenarien vor einer Entscheidung analysiert werden müssen, kann so geschärft werden. Denn nur das ermöglicht praktische Schritte und das Aufsetzen klarer Rechenschaftspflichtprozesse, die Sicherheitsrisikomanagement und -prävention in effizienter Art und Weise.

## 5) Katastrophenplanung

Das kontinuierliche Auseinandersetzen mit Schwachstellen und möglichen Folgen bei der Ausnutzung solcher verbessert die Arbeit von Experten. Hierbei kann von den Prinzipien der Fehlerkultur und des Krisenmanagements viel abgeschaut und übernommen werden. Die positiven Effekte einer transparenten Analyse und Diskussion über erfolgte Fehler sensibilisiert für zukünftige Projekte und verhindert im Idealfall Wiederholungen. Darüber hinaus ermöglicht es allen Beteiligten, eine lernende Grundhaltung einzunehmen und in einen kontinuierlichen Verbesserungsprozess einzutreten. Aus dem Krisenmanagement weiß man, dass Katastrophenplanung selbst einen wichtigen Vorsprung verschafft, wenn ein Krisenfall eintritt. Sie sollte daher entwickelt werden, ohne Wertung und ohne Schuldzuweisung. Dies hilft, um in Krisen einen adäquaten Umgang mit einem eingetretenen Szenario ab der Stunde null verfolgen zu können. Dies alles fördert Transparenz, Autonomie und Vertrauenswürdigkeit als Grundwerte ethischer Richtlinien in der Cybersicherheit.

## 6) Inhärente Privatsphäre (oder Privacy-by-Design)

Ethik-by-Design oder Privatsphäre-by-Design als Orientierungshilfe darf zuletzt in dieser Liste nicht fehlen. Im Bereich Cybersicherheit ist dieser Ansatz vor allem deswegen auch so wichtig, weil er auf die Notwendigkeit der Ganzheitlichkeit hinweist. Nicht nur beim technischen Aufsetzen von Datenbanken, Sicherheitsprozessen, Netzwerken, Geräten, Plattformen etc., sondern auch in der organisationalen Struktur von Gruppen, Richtlinien und Prozessen des Monitorings ist es wichtig, Privatsphäre und Sicherheit zu berücksichtigen. Diese Prinzipien sollten von Anfang an als Werte in der DNA jedes Projektelements einfließen, sodass in der Projektplanung, -gestaltung, -ausführung und -überwachung eine klare Orientierungshilfe reflektiert ist. Da sich die Art und Weise der Bedrohung stetig weiterentwickelt, ist es auch wichtig, dass die Cybersicherheitsethik als ein aktiver Lebenszyklus betrachtet wird, der nicht isoliert in einzelne Prozesselemente einfließt.

## Ausblick

In der diesjährigen Rede zur Münchner Sicherheitskonferenz beteuerte auch die Vize-Präsidentin der Europäischen Kommission Margaritis Schinas, dass die Gefahr von Cyberattacken kontinuierlich steigt. „Because not only their patterns but also the motives of these attacks have changed. They no longer just have an economic rationale. They target ever more sectors and entities that are critical to the functioning of our economies and of our societies." *[10]*. Dies macht die Relevanz einer ganzheitlichen Perspektive auf das Thema einmal mehr deutlich. Die aufgezeigten Ansätze sind erste Anknüpfungspunkte, zu denen jedoch der Konsens besteht, dass sie in Zukunft weiter ausgebaut werden müssen.

## Quellen


[1] European Union Agency for Cybersecurity (2021): ENISA Threat Landscape 2021. April 2020 to mid-July 2021. *https://www.enisa.europa.eu/publications/enisa-threat-landscape-2021*

[2] Shannon Vallor et al. (2018): An introduction to cybersecurity ethics. Santa Clara University. *https://www.scu.edu/media/ethics-center/technology-ethics/IntroToCybersecurityEthics.pdf*

[3] Adam Barth, Anupam Datta, John Mitchel und Helen Nissenbaum (2006): Privacy and contextual integrity: framework and applications. IEEE. *https://ieeexplore.ieee.org/abstract/document/1624011*

[4] Stanford Encyclopedia of Philosophy (2016): Virtue Ethics. *https://plato.stanford.edu/entries/ethics-virtue/*

[5] Stanford Encyclopedia of Philosophy (2019): Consequentialism. *https://plato.stanford.edu/entries/consequentialism/*

[6] Stanford Encyclopedia of Philosophy (2020): Deonotological Ethics. *https://plato.stanford.edu/entries/ethics-deontological/*

[7] European Union Agency for Cybersecurity (ENISA) and CERT (2022): Joint Publication – Boosting your Organisation's Cyber Resilience. *https://www.enisa.europa.eu/news/enisa-news/joint-publication-boosting-your-organisations-cyber-resilience*

[8] European Union Agency for Cybersecurity (ENISA) and CERT (2022): Joint Publication – Boosting your Organisation's Cyber Resilience. *https://www.enisa.europa.eu/news/enisa-news/joint-publication-boosting-your-organisations-cyber-resilience*

[9] World Economic Forum (2020): Ethics-by-Design: An organizational approach to responsible use of technology. *https://www.weforum.org/whitepapers/ethics-by-design-an-organizational-approach-to-responsible-use-of-technology*

[10] Keynote speech by Vice-President Schinas at the Munich Cyber Security Conference (2022): "Check against delivery". *https://ec.europa.eu/commission/presscorner/detail/en/SPEECH_22_1163*

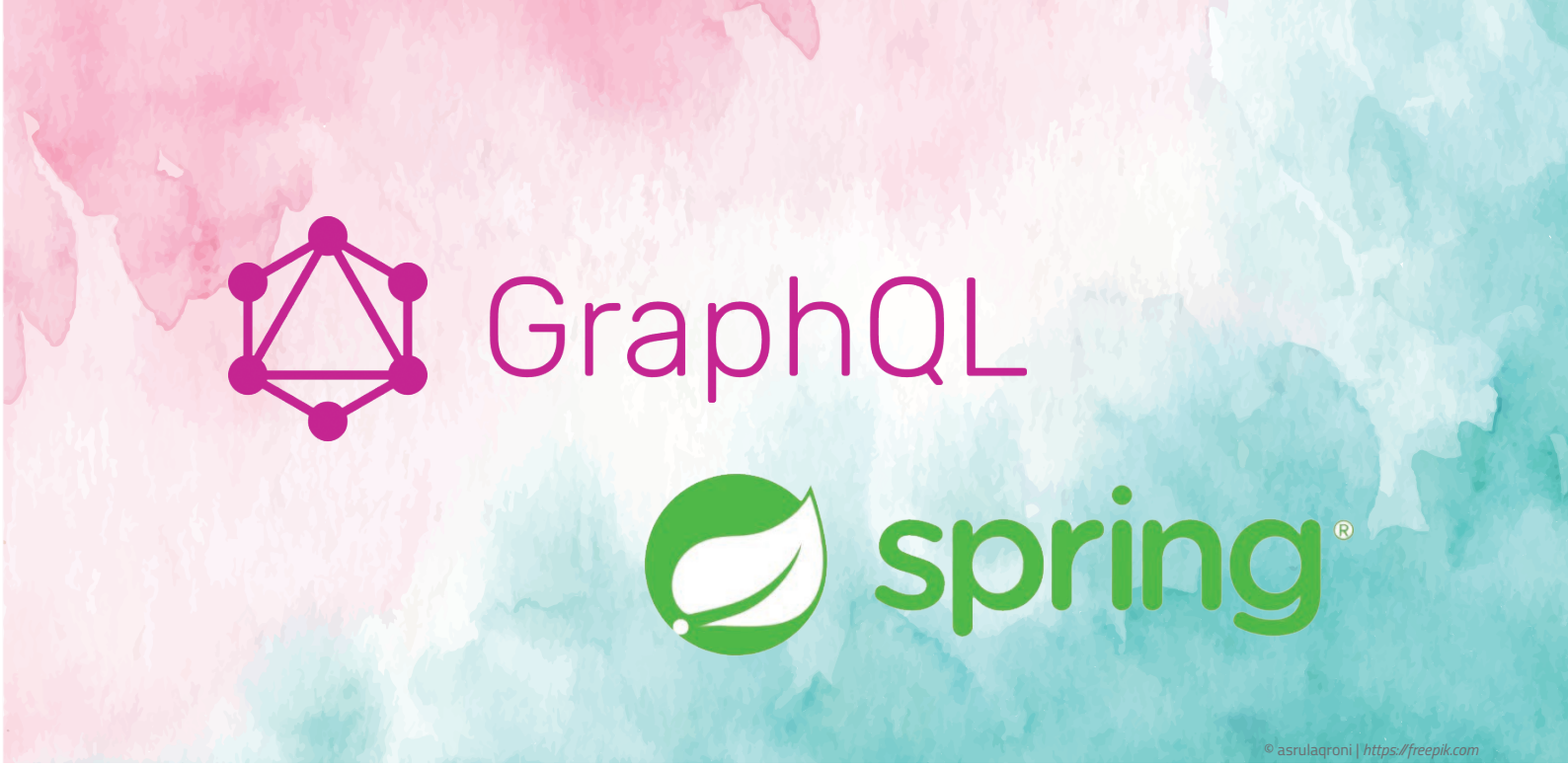


**Corinna Wiedenmann**

*wiedenmann.corinna@gmail.com*


Corinna Wiedenmann studierte Philosophie und Wirtschaft, arbeitete mehrere Jahre im Innovationsmanagement und beschäftigt sich daher stark mit Themen rund um Ethik und Digitalisierung.

Heute arbeitet sie als Leitung der Geschäftsfeldentwicklung für eine international tätige Wirtschaftskanzlei.




# GraphQL-Anwendungen mit Spring

**Nils Hartmann**

*GraphQL-APIs sind eine mächtige Möglichkeit, dem Client flexibel genau die Daten zur Verfügung zu stellen, die er benötigt. Mit der GraphQL-Query-Sprache kann der Client pro Use Case selbst auswählen, welche Daten er vom Server lesen beziehungsweise schreiben will. Die Implementierung eines GraphQL-API bringt jedoch einige Herausforderungen mit sich. Dieser Artikel stellt das neue Projekt spring-graphql vor, das die Entwicklung von GraphQL-APIs mit Spring ermöglicht.*

GraphQL-APIs und die GraphQL-Abfragesprache gibt es bereits seit 2015. Damals als Open-Source-Lösung von Facebook vorgestellt, wird die zugehörige Spezifikation mittlerweile in einem firmenübergreifenden Konsortium, der GraphQL Foundation, weiterentwickelt. Die Spezifikation umfasst die GraphQL-Query-Sprache und beschreibt, wie Server eine GraphQL-Query interpretieren, ausführen und das Ergebnis zurückliefern müssen. Da es sich „nur" um eine Spezifikation handelt, obliegt die Implementierung den Anwendungen selbst, und so gibt es mittlerweile eine Reihe von Programmiersprachen, Bibliotheken und Frameworks, die Entwicklung und Betrieb eigener GraphQL-APIs ermöglichen. Im Java-Umfeld ist dabei besonders das Projekt graphql-java zu nennen, das bereits 2015, also kurz nach Vorstellung von GraphQL, in einer ersten Version veröffentlicht wurde und unter anderem erfolgreich von Twitter eingesetzt wird *[1]*. Dieses Projekt stellt alle Features zur Verfügung, die eine Anwendung benötigt, um GraphQL-Queries ausführen und beantworten zu können. Das API ist dabei vergleichsweise low-level und auch die Anbindung zum Beispiel an einen HTTP-Endpunkt oder die Integration in Laufzeitumgebungen wie Spring oder JEE fehlen.

Auf Grundlage von graphql-java sind in den vergangenen Jahren jedoch mehrere Projekte entstanden, die diese Punkte adressieren und sowohl ein anderes Abstraktionsniveau als auch die Einbindung in Serverumgebungen umsetzen. So wurde Anfang 2020 das „MicroProfile GraphQL" veröffentlicht, dessen prominenteste Implementierung (SmallRye GraphQL*[2]*) auf graphql-java basiert und unter anderem in Quarkus verwendet wird.

Für die Entwicklung von GraphQL-APIs im Spring-Umfeld gab es zunächst eine Reihe von Ansätzen, hier ist insbesondere das „GraphQL Java Kickstart"-Projekt zu nennen *[3]*, das neben der Unterstützung von Spring beziehungsweise Spring Boot auch ein POJO-basiertes Programmiermodell für graphql-java anbietet. Im vergangenen Jahr schließlich stellte zunächst Netflix mit DGS *[4]* ein Projekt zur Entwicklung von GraphQL-Anwendungen vor, das sich explizit an Spring Boot richtet. Mitte des Jahres folgte dann die Ankündigung

des Spring-Teams, ein eigenes GraphQL-Projekt der Spring-Projektfamilie hinzuzufügen: spring-graphql *[5]*. Damit gibt es nun auch ein „offizielles" Projekt für Spring beziehungsweise Spring Boot, das im Folgenden vorgestellt wird.

## pring-graphql

Wie in vielen Spring-Projekten üblich, besteht spring-graphql aus zwei Teilen. Das eigentliche API und die Implementierung sind Spring-basiert und daneben gibt es für Spring Boot ein Starter-Projekt, dass die automatische Konfiguration für spring-graphql innerhalb von Spring Boot vornimmt. Die Aktivierung in einer Spring-Boot-Anwendung erfolgt dann durch das Hinzufügen des Starter-Projekts, das auch für die Integration in die weitere Spring-Boot-Landschaft sorgt. So wird ein HTTP-Endpunkt, über den ein Client GraphQL-Queries ausführen kann, durch die bekannten Projekte SpringMVC oder Spring WebFlux automatisch konfiguriert und zur Verfügung gestellt. Auch die Integration zur Kommunikation über WebSockets (zum Beispiel für GraphQL Mutations) und Unterstützung für Spring Security erfolgen automatisch, sobald die entsprechenden Starter-Projekte der Anwendung hinzugefügt werden. Per Konfiguration lässt sich außerdem der GraphQL-Web-Editor GraphiQL *[6]* aktivieren und das Introspection-API ein- oder ausschalten.

Die Herausforderung beim Anbieten eines GraphQL-API besteht also nicht länger darin, einen stabilen Toolstack für Spring zu finden. Wir können uns nun ganz auf die Implementierung des API konzentrieren und dabei das gewohnte Spring-Programmiermodell und die gewohnten Spring-Projekte verwenden.

## GraphQL-API beschreiben

Die Entwicklung eines GraphQL-API beginnt mit der Definition eines GraphQL-Schemas. Das Schema beschreibt, welche Typen es gibt, welche Felder die Typen haben und welche Beziehungen es zwischen den einzelnen Typen gibt. Auf diese Weise entsteht ein Objekt-Graph, aus dem ein Client dann mit einer Query die gewünschten Felder auswählen kann. Der Client kann dabei nur dem beschriebenen Objekt-Graphen folgen, kann also – anders als zum Beispiel in SQL – keine beliebigen Joins machen. Außerdem gibt es genau einen Einstiegspunkt, ein Root-Objekt, in den Objektgraphen. Dieses Root-Objekt wird „Query" genannt und besteht ebenso wie fachliche Typen aus einer Reihe von Feldern. Ein Client muss seine Query mit einem dieser im Root-Objekt beschriebenen Felder beginnen.

*Listing 1* zeigt ein exemplarisches Schema eines API für eine Blog-Anwendung, das mit der Schema-Definition-Language (SDL) beschrieben wird, die Teil der GraphQL-Spezifikation ist und die auch von spring-graphql für die Beschreibung der Schemas unterstützt wird.

Es besteht aus den fachlichen Typen `Post`, `Comment` und `User`. Im Root-Typ sind die beiden Felder `posts` und `post` definiert, sodass ein Client seine Query mit einem dieser beiden Felder beginnen muss. Aus einem Post könnte der Client dann zum Beispiel den Titel auswählen (`title`), die Information, wer den Post geschrieben hat (`writtenBy`), oder die Kommentare zu dem Post (`comments`). Ein Client könnte aber nicht direkt die Kommentare ohne einen Post abfragen, was in SQL möglich wäre, wenn man annimmt, dass die Objekte jeweils eigene Datenbanktabellen wären. Auch aus diesem Grunde ist GraphQL

```
type Comment {
  id: ID!
  createdAt: String!
  comment: String!
}

type User {
  id: ID!
  username: String!
  fullname: String!
}

type Post {
  id: ID!
  title: String!
  excerpt: String!
  body: String!
  createdAt: String!
  writtenBy: User!

  comments: [Comment!]!
}
type Query {
  posts(size: Int, page: Int): [Post!]!
  post: Post
}
```

*Listing 1: Schema eines GraphQL-API*

```
query {
  post {
     id
     title
     excerpt
  }
}
```

*Listing 2: Eine GraphQL-Query*

übrigens nicht mit SQL vergleichbar, auch wenn die Endung „Query Language" in beiden Technologien dies suggerieren könnte.

## Verarbeitung einer GraphQL-Query

Bestandteil der GraphQL-Spezifikation ist die Beschreibung, wie eine Query aufgebaut sein muss (also die Syntax und Grammatik der Sprache), in welchem Format diese an den Server geschickt wird und in welchem Format der Server das Ergebnis an den Client zurückschicken muss. Wie die eigentliche Bearbeitung der Query, also in erster Linie die Ermittlung der abgefragten Daten, erfolgt, ist nicht Bestandteil der Spezifikation, sondern Teil des verwendeten GraphQL-Frameworks.

Die meisten GraphQL-Frameworks parsen dazu zunächst die vom Client gesendete Query. Ist diese gültig, rufen die Frameworks für jedes darin enthaltene Feld eine „Resolver-Funktion" auf. Die Resolver-Funktion muss von der Anwendung implementiert werden und ist dafür zuständig, den Wert für das abgefragte Feld zurückzuliefern. Auf diese Weise funktionieren auch graphql-java beziehungsweise spring-graphql. In graphql-java werden diese Funktionen DataFetcher genannt. Spring-graphql stellt darüber eine Annotation-basierte Abstraktion („Annotated Controllers") zur Verfügung. Mit diesen Annotationen werden Handler-Funktionen markiert, die für die Ermittlung der Daten eines konkreten Felds zuständig sind. Dieses Muster ist ähnlich wie in REST-Controllern, bei denen mit der Request-Mapping-Annotation Methoden ge-

```
@Controller
public class BlogGraphQLController {
  @Autowired
  private PostRepository postRepository;

  @QueryMapping
  public Optional<Post> post() {
     return postRepository.findFirstByOrderByCreatedAtDesc();
  }
}
```

*Listing 3: Die Handler-Methode für das post-Feld*

kennzeichnet werden, die Daten für einen konkreten REST-Endpunkt ermitteln sollen.

## Handler-Methoden für Queries

In *Listing 2* ist eine GraphQL-Query zu sehen, die einen Post aus dem Query-Objekt auswählt und aus dem Post die Felder `id`, `title` und `excerpt` abfragt.

Die Controller-Klasse, die die Daten für das post-Feld am Query-Objekt ermitteln soll, könnte wie in *Listing 3* gezeigt, aussehen. Da mit spring-graphql die gewohnte Spring-Infrastruktur verwendet werden kann, verwendet die Handler-Funktion ein Repository, um Daten aus der Datenbank zu lesen. Dabei handelt es sich um ein Spring Data JPA Repository und per Namenskonvention der Methode ist ausgedrückt, welche Datenbank-Query ausgeführt werden soll. In diesem Fall soll das neuste Post-Objekt aus der Datenbank gelesen und an das GraphQL-Framework zurückgeliefert werden *(siehe Listing 3)*.

Mit der `@QueryMapping`-Annotation an einer Methode wird ausgedrückt, dass die Methode die Daten für ein Feld am Root-Objekt („Query") ermittelt. Das Feld im Schema muss genauso heißen wie die Methode selbst (`post` in diesem Beispiel). Sofern es eine Abweichung zwischen dem Feld-Namen am Schema und dem Methoden-Namen gibt, kann mit einem Argument an der Annotation der GraphQL-Feldname explizit angegeben werden.

In diesem Beispiel soll die Handler-Funktion ein JPA-Entity-Objekt Post zurückliefern, sofern mindestens eins in der Datenbank gefunden wurde. In unserer Beispiel-Query wird auf dem GraphQL-Post-Typ nun das `title`-Feld abgefragt. Sofern das aus einer Handler-Funktion zurückgegebene Java-Objekt über ein gleichnamiges Feld beziehungsweise eine gleichnamige `getter`-Methode verfügt, braucht für das Feld keine weitere Handler-Funktion geschrieben zu werden.

Anders sieht es für Felder aus, die nicht an der Java-Klasse vorhanden sind. Das `excerpt`-Feld zum Beispiel soll einen kurzen Ausriss aus dem Blog-Post zurückliefern (etwa, damit der Client eine Vorschau des Posts darstellen kann). Diese Funktionalität ist in der Post-Entity allerdings nicht vorgesehen, somit gibt es dort weder eine `getter`-Funktion noch ein entsprechendes Feld.

In diesem Fall kann der Server eine Schema-Mapping-Methode implementieren. Diese Methode funktioniert wie eine Query-Mapping-Methode, nur dass sie sich auf ein Feld bezieht, das an einem anderen Typ als dem Root-Typ definiert ist (in diesem Fall dem Post). Auch hier entspricht der Methoden-Name wieder dem Namen des Feldes des GraphQL-API, dessen Daten ermittelt werden sollen. Als Parameter bekommt die Funktion von spring-graphql das Parent-Objekt übergeben, das zuvor mit der Handler-Funktion ermittelt wurde (hier die Post Entity, *siehe Listing 4*).

Mit einer Schema-Mapping-Methode können also Felder implementiert werden, die im Java-Klassenmodell nicht vorhanden sind oder dort anders funktionieren als am API gewünscht. Andersherum ist es übrigens nicht möglich, dass ein Client ein Feld der zurückgelieferten Java-Objekt-Instanz abfragt, das nicht im Schema definiert ist. Ein Client kann immer nur die im Schema definierten Felder abfragen.

Felder in GraphQL können Argumente haben und ähneln damit Methoden in Java. Soll der Client beispielsweise in der Lage sein, die Länge des Excerpt selbst zu bestimmen, könnte die Definition des Feldes im Schema entsprechend wie in *Listing 5* angepasst werden. In *Listing 6* ist die aktualisierte Query, in der ein Client nun auch die Länge des Excerpt angibt, zu sehen.

```
@Controller
public class BlogGraphQLController {
  . . .
  @QueryMapping
  public Optional<Post> post() { . . . }

  @SchemaMapping
  rublic String excerpt(Post post) {
     return post.getBody().substring(0, 10);
  }
}
```

*Listing 4: Eine Schema-Mapping-Methode*

```
type Post {
  ...
  excerpt(maxLength: Int): String!
  ...
}
```

*Listing 5: Argumente an einem GraphQL-Feld*

```
query {
  post {
     id
     title
     excerpt(maxLength: 15)
  }
}
```

*Listing 6: Argumente in einer GraphQL-Query*

Die vom Client angegebenen Argumente werden den Handler-Funktionen als Parameter übergeben, die dazu mit @Argument annotiert sein müssen. Da es sich beim maxLength-Argument um ein optionales Argument handelt, kann das Argument ein Java-Optional sein. Die Implementierung dafür ist in *Listing 7* zu sehen.

## Integration mit weiteren Spring-Projekten

Neben den Queries gibt es noch zwei weitere Operationen in GraphQL, die ein Client ausführen kann. Mit einer Mutation kann ein Client Daten auf dem Server ändern. Mit Subscriptions kann sich ein Client über neue Daten auf dem Server informieren lassen.

Diese beiden `Operation`-Typen werden ebenfalls über Handler-Funktionen, die mit `@MutationMapping` beziehungsweise `@SubscriptionMapping` annotiert werden, implementiert. Dabei gibt es keinen Unterschied zu den Query- beziehungsweise Schema-Mapping-Funktionen. Bei einem Subscription-Mapping muss allerdings ein reaktives Flux-Objekt zurückgeliefert werden. Daten, die von einer Subscription ermittelt werden, werden an den Client üblicherweise über eine WebSocket-Verbindung zurückgegeben. Das ist im Betrieb technisch nicht trivial, deswegen sollte es gute Gründe geben, wenn ein GraphQL-API Subscriptions zur Verfügung stellt.

Im *Listing 8* ist die Schema-Definition für den `Mutation`-Typen zu sehen, der ein Feld (`newPost`) enthält, mit dem ein Client einen neuen Blog-Post hinzufügen kann. Außerdem wird ein `input`-Typ beschrieben, der die Daten für einen neuen Blog-Post enthält. Ein Objekt dieses Typs muss vom Client zum Server als Argument für das `newPost`-Feld geschickt werden. Der Rückgabe-Typ ist hier ein Union-Typ, der aus mehreren Typen zusammengesetzt ist. Der Server kann, je nach Ergebnis der Mutation, einen der Typen zurückliefern und damit in diesem Fall ausdrücken, ob das Anlegen des neuen Posts funktioniert hat oder nicht.

Eine mögliche Implementierung für diese Mutation ist in *Listing 9* zu sehen. Die Prüfung, ob der Client die korrekten Felder mit den korrekten Typen sendet, erfolgt durch spring-graphql immer automatisch. Durch die Integration von spring-graphql mit den weiteren Komponenten des Spring-Frameworks ist es aber zum Beispiel möglich, mittels Bean-Validation-Annotation weitere Bedingungen zu definieren, die vor der Verarbeitung der Query überprüft werden, die sich jedoch im GraphQL-Schema nicht ausdrücken lassen.

```
input NewPostInput {
     title: String!
     body: String!
}

type NewPostSuccess {
  newPost: Post!
}

type NewPostFailed {
  error: String!
}

union NewPostPayload = NewPostSuccess | NewPostFailed

type Mutation {
  newPost(input: NewPostInput!): NewPostPayload!
}
```

*Listing 8: Das Schema einer GraphQL-Mutation*

```
@Controller
public class BlogGraphQLController {
  . . .
  @QueryMapping
  public Optional<Post> post() { . . . }

  @SchemaMapping
  public String excerpt(Post post,
            @Argument Optional<Integer> maxLength) {
     return post.getBody().substring(0, maxLength.
orElse(10));
  }
}
```

*Listing 7: GraphQL-Argumente an einer Handler-Funktion*

Die Integration mit Spring Security erlaubt es, die bekannten Annotationen zum Überprüfen der erforderlichen Rechte an einer Handler-Methode zu verwenden (etwa `PreAuthorize`). Außerdem kann sich eine Handler-Methode mit der `@Authentication-Princial`-Annotation den aktuellen Principal übergeben lassen. Die Anmeldung und Authentifizierung am GraphQL-Endpunkt erfolgt dann über die gewohnten Wege aus Spring-Security.

## Verarbeitung einer Query optimieren

Die Abarbeitung von Queries erfolgt in spring-graphql zunächst sequenziell, es werden also Feld für Feld nacheinander die Handler-Funktionen aufgerufen. Dieses Verhalten kann sich bei komplexen Queries mit vielen Feldern allerdings negativ auf die Performance auswirken. Es könnte zum Beispiel sein, dass der Client eine Menge von Stories über das `stories`-Feld abfragt und für jede der Story das zugehörige User-Objekt. Je nach Implementierung kann es dabei zum 1+N-Problem kommen (ein Datenbank-Request liefert n Stories zurück und die Schema-Mapping-Funktion für das User-Feld an der Story wird dann jeweils einmal für alle n Stories aufgerufen). Möglicherweise liegen die User-Objekte gar nicht in der lokalen Datenbank, sondern müssen sogar von einem anderen Microservice abgefragt werden. Für solche Szenarien gibt es eine Reihe von Möglichkeiten, wie die Verarbeitung einer Query optimiert werden kann:

- Mit einem „DataLoader" können Aufrufe, zum Beispiel an einen Remote-Service, zusammengefasst werden. In diesen Fall würde dann die Handler-Funktion nicht den User für jede einzelne Story zurückliefern, sondern zum Beispiel nur eine ID des Users, die an den Story-Objekten direkt hinterlegt ist und keine weiteren Datenbank- oder Servicezugriffe erfordert. Wenn alle IDs ermittelt sind, wird dann ein „Batch Loader" aufgerufen, der von spring-graphql alle ermittelten IDs übergeben bekommt und dann dazu die passenden User-Objekte zurückliefern kann (beispielsweise durch eine Datenbank-Query oder einen Aufruf des Remote-Service).
- Handler-Methoden können außerdem asynchron und reaktiv implementiert werden. Dazu liefern sie ein Java CompletableFuture oder ein Spring Reactor Flux- beziehungsweise Mono-Objekt zurück. Spring-graphql kann die Handler-Funktionen dann parallel ausführen. Insbesondere Zugriffe auf Microservices können dank des Spring WebClient sehr einfach reaktiv erfolgen.
- Eine Handler-Funktion kann abfragen, welche konkreten Felder in einer Query abgefragt sind. Mit diesen Informationen kann eine Handler-Funktion dann zum Beispiel eine Datenbank-Query generieren, die genau die passenden Daten ermittelt und da-

```
@Controller
public class BlogGraphQLController {

  @Autowired
  private BlogService blogService;
  . . .

  record NewPostInput(
     @Size(max=50) String title,
     String body) {}

  record NewPostSuccess(Post newPost) {}
  record NewPostFailed(String error) {}

  @MutationMapping
  @PreAuthorized("hasRole('ROLE_USER')")
  public Object newPost(@Valid @Argument NewPostInput input,
                                  @AuthenticationPrincipal User user) {
     try {
        Post newPost = blogService.addPost(
           // Verwendet zum Anlegen des neuen Posts
           // den aktuell angemeldeten User
           user.getId(),
           input.title(),
           input.body()
        );

        return new NewPostSuccess(newPost);
     } catch (Exception ex) {
        return new NewPostFailed(ex.getMessage());
     }
  }
}
```

*Listing 9: Implementierung einer Mutation mit Bean-Validation und Spring-Security*

bei nicht „zu viele" oder „zu wenige" Datenbank-Joins macht, was bei einem statischen Mapping passieren könnte.

## Fazit

Mit spring-graphql gibt es nun endlich eine „offizielle" Standard-Lösung für die Entwicklung von GraphQL-APIs auf Basis von Spring/Spring Boot.

Auch wenn bei der Entwicklung der Teufel aufgrund der Flexibilität des Clients im Detail steckt, ist es durch die Integration mit dem Spring-Ökosystem zumindest einfach und schnell möglich, einen Prototyp, ein Proof-of-Concept oder MVP zu bauen, auszuprobieren und Erfahrungen mit GraphQL zu sammeln. Dabei kommt einem zugute, dass gewohnte Programmierkonzepte und Komponenten aus der Spring-Welt weiterverwendet werden können.

Für die Herausforderungen, die sich beim Bauen „echter" GraphQL-APIs ergeben, gibt es eine Reihe von flexiblen Lösungsmöglichkeiten, die sich ganz gezielt abhängig von der eigenen Architektur und eigenen Anforderungen anwenden lassen.

## Referenzen


[1] *https://github.com/graphql-java/graphql-java/discussions/2591*
[2] *https://github.com/smallrye/smallrye-graphql*
[3] *https://www.graphql-java-kickstart.com/spring-boot/*
[4] *https://netflix.github.io/dgs/*
[5] *https://spring.io/projects/spring-graphql*
[6] *https://github.com/graphql/graphiql*



**Nils Hartmann**
*nils@nilshartmann.net*

Nils Hartmann (Twitter: @nilshartmann) ist freiberuflicher Softwareentwickler und -architekt, Trainer und Coach aus Hamburg. Er unterstützt Teams bei der Entwicklung von Backend- und Frontend-Anwendungen mit Java, Spring, GraphQL, React und TypeScript und gibt Workshops und Schulungen zu diesen Themen. Nils ist Co-Autor des Buches „React - Die praktische Einführung" (dpunkt-Verlag)

Twitter: @nilshartmann

# Flexible Anwendungsarchitektur mit der Clean Architecture Teil 2: Flexibilität auf dem nächsten Level

Matthias Eschhold, Novatec Consulting GmbH

*In „Flexible Anwendungsarchitektur Teil 1" in der vorigen Java aktuell 2/22 wurde die Clean Architecture als Architekturmuster für Flexibilität beschrieben. In Teil 2 werden wir Flexibilität hinsichtlich Zeit, Kosten, Komplexität und Geschäftswert betrachten. Eine entscheidende Herausforderung besteht darin, das richtige Verhältnis dieser Faktoren zu finden. Hierfür bedarf es „Flexibility Enablers", die die Architekturarbeit beschleunigen und den gewünschten Qualitätsstandard sicherstellen!*

Der vorige Teil beschreibt Flexibilität vereinfacht als höherwertige Stufe von Erweiterbarkeit und Wartbarkeit. Flexibilität in der Technologie, Flexibilität durch Stabilität sowie Flexibilität durch eine fachlich ausdrucksstarke Abbildung der Architektur werden durch die Anwendung der Clean Architecture und der in Teil 1 beschriebenen Ideen, wie Ports und Adapters, Dependency Inversion, Interface Segregation, Dependency Injection oder die Two-Way-Mapping-Strategie, erreicht.

## Das Qualitätsmerkmal Flexibilität für die Anwendungsarchitektur

Flexibilität ist die Anpassungsfähigkeit von einem aktuellen Systemzustand in einen neuen, gewünschten Systemzustand. Durch die Anwendung des Ports-und-Adapters-Musters wird zum Beispiel die Anpassungsfähigkeit bei Schnittstellenänderungen integrierter IT-Systeme unterstützt, da Änderungen nur lokal im Adapter und nicht verteilt in der Anwendung sind. Analog trifft diese Aussage auch auf die Flexibilität durch eine fachliche, ausdrucksstarke Architektur zu. Auf Basis fachlicher Module mit granularen Schnittstellen entsteht eine klare und verständliche Architektur, die dadurch eine schnelle Reaktionszeit bei Änderungen und Erweiterungen ermöglicht. *[1, 2, 3]* beschreiben diese Eigenschaften als „Design for Change" und betrachten dies als Grundlage für eine flexible Architektur, deren Flexibilität sich auf Basis des Anpassungsaufwands beziehungsweise der Stabilität von Domänenoperationen, bei Eintreten eines Änderungsbedarf, bewerten lässt. Flexibilität wird folglich dann benötigt, wenn sich

- funktionale Anforderungen,
- Qualitätsanforderungen oder
- die Umgebung des Systems ändert.

Um auf diese Änderung zu reagieren, muss ein neuer Zustand des Systems hergestellt werden.

Der Flexibilitätsgrad einer Anwendungsarchitektur definiert sich vereinfacht anhand des Zeit- und Kostenbedarfs für die Anpassungen an einen sich wechselnden Umstand *(siehe Abbildung 1)*. Der Flexibilitätsgrad ist am höchsten, wenn Zeit- und Kostenbedarf nahe null ist. Dies bedeutet, dass das System auf einen veränderten Umstand sehr schnell reagieren kann und hierfür nur wenige Anpassungen im Quellcode notwendig sind. Die Adaption von einem alten auf einen neuen Systemzustand benötigt folglich wenig Entwicklungszeit. Ist das Gegenteil der Fall, ist der Flexibilitätsgrad niedrig.

Flexibilität muss initial konzipiert und in Softwarebausteinen implementiert werden. Es entsteht folglich initialer Entwicklungsaufwand. In Summe muss der durch Flexibilität erzielte Geschäftswert größer sein als die Gesamtkosten für die Flexibilität. Wird kein Geschäftswert erzielt oder wird eine Lösung zu kostenintensiv, sodass keine Wertgenerierung stattfindet, sind konkretere Lösungen zu bevorzugen. Wird jedoch ein Geschäftswert erzielt, rechtfertigt dies höhere Gesamtkosten. Dies unterliegt der Annahme, dass je höher der Geschäftswert ist, desto höher auch die Komplexität und Kosten für die Anforderung sind *[1, 3]*. Diesen Zusammenhang visualisiert *Abbildung 2*.

Hierbei muss jedoch beachtet werden, dass ab einem gewissen Grad von Flexibilisierung anhand von Generalisierung, Abstraktion und Wiederverwendung die Komplexität der Softwarebausteine exponentiell zunimmt *(siehe Abbildung 3)*. Dies hat wiederum eine ganze Reihe von Nachteilen, wie

- der Aufbau von technischen Schulden aufgrund fehlender Verständlichkeit,
- erhöhter Abstimmungsbedarf zwischen Teams sowie
- ein erhöhtes Risiko für Fehler.

Dadurch steigen die Entwicklungskosten unkontrolliert und die Geschäftsziele sind in Gefahr *[4]*.

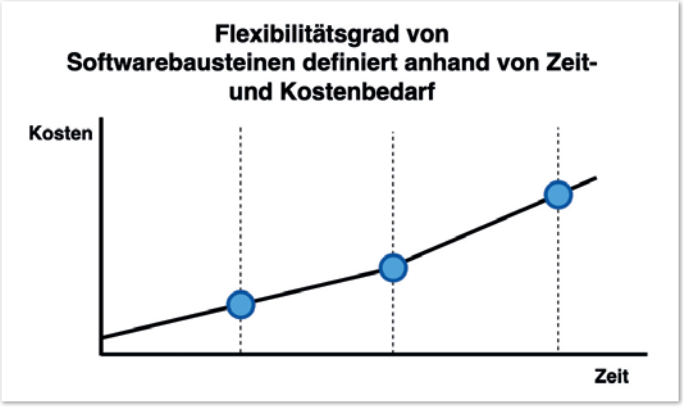



*Abbildung 1: Flexibilitätsgrad von Softwarebausteinen, definiert anhand von Zeit- und Kostenbedarf (© Matthias Eschhold, in Anlehnung an [1, 2])*

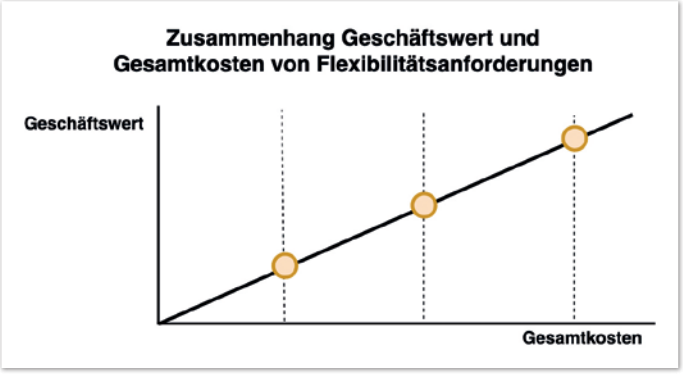



*Abbildung 2: Zusammenhang von Geschäftswert und Gesamtkosten von Flexibili-tätsanforderungen (© Matthias Eschhold)*

Die *Abbildungen 1 bis 3* verdeutlichen die Herausforderung des Architekturentwurfs. Die Frage ist nun, wie dies alles in der Praxis berücksichtigt werden kann. Es erscheint unrealistisch, ausreichend Information über die benötigten Faktoren wie Zeit, Kosten, Veränderung und künftig zu erwartender Komplexität für jeden konkreten Fall zu ermitteln und in der Entscheidung berücksichtigen zu können. Es ist weder möglich noch zielführend, alle Elemente eines Softwaresystems zu flexibilisieren. Dies ist, wie beschrieben, zu kostenintensiv und die verfügbaren Ressourcen sind in der Regel zu knapp. Davon abgesehen steigt die Komplexität unnötig. Eine Stellschraube ist also, exakt zu wissen, an welchen Stellen und auf welche Art schnelle Anpassungsfähigkeit im System benötigt wird. Hierfür bietet sich an, Qualitätsszenarien über Qualitätsbäume zu ermitteln *[3, 5]*. Der Qualitätsbaum in *Abbildung 4* zeigt das Qualitätsmerkmal Flexibilität und definierte Teilmerkmale aus realen Projektkontexten.

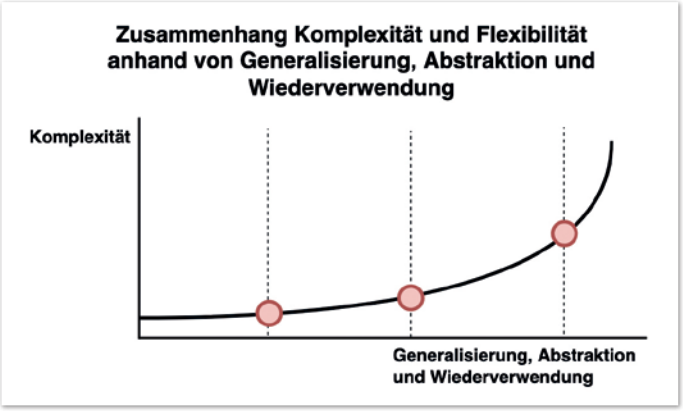



*Abbildung 3: Zusammenhang zwischen Komplexität und Flexibilität anhand von Ge-neralisierung, Abstraktion und Wiederverwendung (© Matthias Eschhold, in Anleh-nung an [4])*

## Effektive Architektur- und Entwicklungsarbeit durch Flexibilitätsmuster

Die analytischen Betrachtungen müssen zu Lösungsstrategien führen, die die Flexibilitätsanforderungen frei vom funktionalen Kontext adressieren und die Entwickler/innen auf Basis einer gemeinsamen Idee unterstützen. Benötigt wird eine zweite Stellschraube, die die Anforderungen in der Architektur und der Implementierung effizient erfüllt. Ein Mittel der Unterstützung in der Softwarearchitektur, das die aufgeführten Aspekte erfüllt, findet sich im Muster. Die Clean Architecture als Architekturmuster definiert Verantwortungsbereiche als Ringe und Beziehungsregeln zwischen den Elementen der Ringe. Die Elemente der Ringe werden als Klassenstereotype ausgedrückt. Indem die Domäne klar abgegrenzt wird und keine Beziehungen auf die Infrastruktur aufweist, soll eine gute Basis für Flexibilität erreicht werden. Dies fördert Verständlichkeit, schnelle Anpassung auf Veränderung, erleichterte Testbarkeit und Refactoring des Quellcodes. Dies wurde ausführlich in Teil 1 beschrieben. *Abbildung 5* zeigt die beschriebene stereotypische Mustersprache der Anwendungsarchitektur.

Durch diesen Rahmen, insbesondere durch die Stereotype, lassen sich **Flexibilitätsmuster** definieren, die zielgerichtet Flexibilität in der Domäne oder in der Infrastruktur ansprechen. Die Grundideen der Flexibilitätsmuster finden sich in den Entwurfsmustern der Gang of Four wieder. Im Kontext der Clean Architecture können die Flexibilitätsmuster als Spezialisierung der Entwurfsmuster der Gang of Four verstanden werden. Durch die Stereotype können die Flexibilitätsmuster präzise und unmissverständlich kommuniziert werden. Dadurch sinken die Entwicklungskosten für gleichartigen Flexibilitätsbedarf; dies führt zu einem ausgewogeneren Verhältnis zwischen Geschäftswert, Kosten und Komplexität.

Für eine ausführliche Beschreibung der Entwurfsmuster *Factory*, *Strategy* und *Decorator* wird auf *[6, 7]* verwiesen. Die neue Interpretation als Flexibilitätsmuster in der Clean Architecture wird auf einem Steckbrief zusammengefasst. Der Steckbrief umfasst

- die Grundidee des verwendeten Entwurfsmusters,
- den Einsatzzweck in der Clean Architecture
- sowie ein verallgemeinertes Klassenmodell.

Für die folgenden Musterbeschreibungen finden sich Codebeispiele in Java auf GitHub *[8]*.

## Output Adapter Factory

Die Grundidee des *Factory*-Musters ist es, die Erzeugung komplexer Objekte in eine extra hierfür vorgesehene Klasse – die *Factory* – auszulagern und dies nicht in der Klasse, die das Objekt nutzt, zu tun. Dadurch verringert sich die Kopplung, insbesondere wenn sich hinter dem erzeugten Objekt ein Verbund zusammengehöriger Objekte befindet und eine hohe Komplexität im Erzeugungsprozess besteht. Im Zusammenhang der *Output Adapter Factory* wird bei der


*Abbildung 4: Qualitätsbaum des Qualitätsmerkmals Flexibilität (© Matthias Esch-hold)*

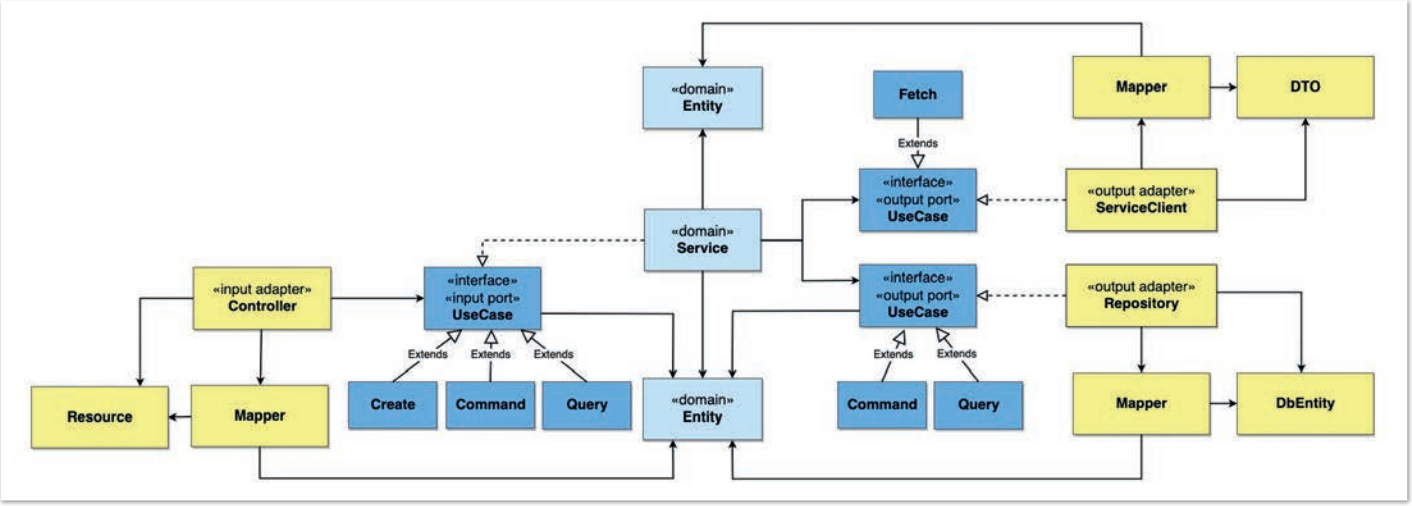



*Abbildung 5: Klassenstereotype (Mustersprache) der Clean Architecture (© Matthias Eschhold)*

Objekterzeugung eine konkrete Adapter-Implementierung zur Nutzung in einem Service erzeugt. Hinter ausgehenden Adaptern zu anderen IT-Systemen verbergen sich Abhängigkeiten und Risiken, wie beispielsweise

- nicht kommunizierte Schnittstellenänderungen,
- fehlende fachliche Datenkonsistenz bei zugelieferten Daten,
- inakzeptable Antwortzeiten oder
- fehlende termingerechte Zulieferung vereinbarter Service-Schnittstellen.

Diese Risiken können durch das *Output Adapter Factory Pattern* *(siehe Abbildung 6)* abgemildert werden.

Wenn die *Adapter*-Erzeugung auf eine Factory ausgelagert wird, ist der Austausch der Implementierung für die Domäne transparent. Die Entscheidung, welche Adapter-Implementierung erzeugt werden soll, trifft die *Factory*. Um diese Entscheidung zu treffen, muss die *Factory* notwendige Informationen aus einer Quelle *(Source)* beziehen. Eine Quelle kann eine Datei, eine Umgebungsvariable, eine Datenbank oder auch ein Parameter der Erzeugen-Methode einer *Factory* sein.

Sowohl die Reaktion auf temporäre Störungen bei Servicegebern als auch die risikoarme Einführung von Features gehören zu gängigen Anforderungen von Product Ownern und IT-Projektleiter/innen in agilen Entwicklungsumfeldern. *[9]* beschreibt ein Risikomanagement auf Basis von Feature Toggles. Die *Output Port Adapter Factory* ist allerdings kein Feature Toggle. Es unterstützt jedoch als Muster die Implementierung eines Feature Toggle in der Clean Architecture. Im einfachsten Fall wird die Adapter-Implementierung durch eine Mock-Implementierung ausgetauscht.

Um dieses Muster an einem praktischen Beispiel zu verdeutlichen, wird die Funktion der Fahrzeugbewertung betrachtet. Der durchschnittliche Marktwert eines Fahrzeugs wird über einen externen Service ermittelt. Hierfür werden der Fahrzeugtyp, das Baujahr, die Ausstattung und der Kilometerstand benötigt. Die Stammdaten für Fahrzeuge können aus dem zentralen Bestandssystem anhand der Fahrgestellnummer ermittelt werden. Der Kilometerstand und die

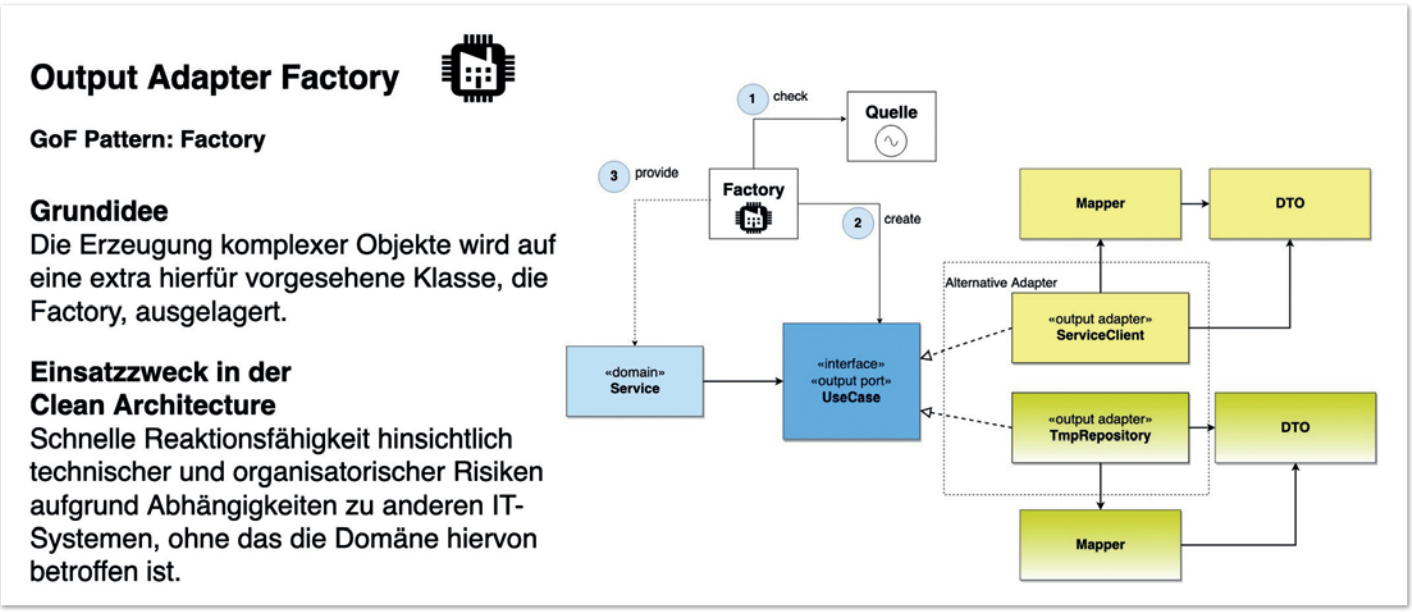



*Abbildung 6: Steckbrief Output Adapter Factory Pattern (© Matthias Eschhold)*

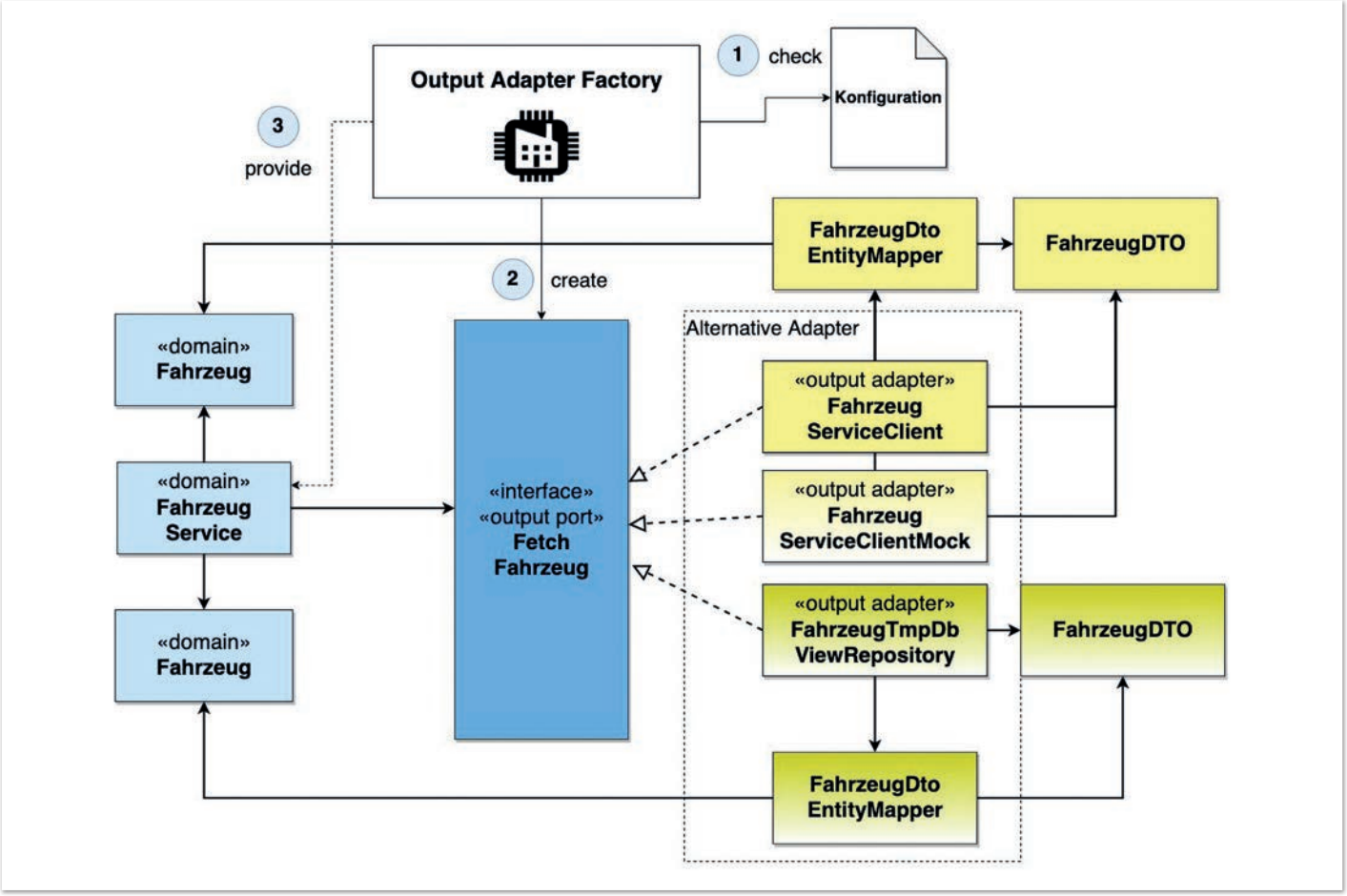



*Abbildung 7: Output Adapter Factory, angewendet in der Fahrzeugbewertung (© Matthias Eschhold)*

Fahrgestellnummer werden als Eingabeparameter erfasst. Dies stellt das Minimum Viable Product (MVP) des neuen Fahrzeugbewertungssystems dar.

Stellen wir uns vor, dass die Entwicklung der Schnittstelle des zentralen Bestandssystem zeitgleich mit der Entwicklung des MVP beginnt und in der Folge keine Beschreibung der Schnittstelle zur Verfügung steht. Oder noch schlimmer, dass die Schnittstelle nicht rechtzeitig zugeliefert wird und deswegen das MVP in Gefahr ist. Die gute Nachricht: Für die Entwicklung und zum Testen der Domäne benötigen wir die Schnittstelle zunächst nicht! Diese Flexibilität erreichen wir durch Anwendung der Clean Architecture. Trotz allem rettet dies nicht das MVP.

Die *Output Adapter Factory* kombiniert mit einer fachlichen Kompensation allerdings schon! Eine Zwischenlösung aus dieser misslichen Lage könnte etwa eine lesende Datenbanksicht oder ein Datenaustausch über eine Datei sein. *Abbildung 7* zeigt die Anwendung des Musters am Beispiel der Fahrzeugbewertung. Das Team des zentralen Stammdatensystems für Fahrzeuge stellt als Kompensationslösung eine Datenbanksicht zur Verfügung, um das MVP zu erreichen. Die Fahrzeugbewertung implementiert einen *Adapter* und einen dazugehörigen Mapper. Parallel wird die HTTP-Schnittstelle fertiggestellt. Ein Umstieg von einer zur anderen Systemumgebung kann dann kontrolliert durch Austausch der *Adapter*-Implementierung erfolgen. Entstehen Probleme bei der Integration des zentralen Stammdatensystems, kann wieder auf die lesende Datenbanksicht zurückgesprungen werden.

Die Stärke der *Output Adapter Factory* ist die Nachhaltigkeit in der Architektur aus Sicht der Domäne. Die fachliche Kompensation lässt sich ohne Quellcodemodifikation in der Domäne durchführen. Sind Änderungen in der Infrastruktur für die Domäne transparent, können Aufwände für fachliche Abnahmetests im besten Fall vermieden oder zumindest stark reduziert werden. Auch dies beschleunigt die Auslieferungen der adaptierten Software und senkt Kosten.

## Service Strategy

Im Vergleich zum *Output Adapter Factory Pattern* zielt die *Service Strategy* auf unterschiede (Varianz) in der Domäne ab. Die Varianz wird hier intern, das heißt innerhalb der Domäne, gesteuert. Die Grundidee der *Strategy* ist es, eine Austauschbarkeit von Algorithmen zu ermöglichen *(siehe Abbildung 8)*.

Im Kontext der Clean Architecture stellen unterschiedliche Algorithmen Varianz in der Domänenlogik dar. Diese Varianz entsteht zum Beispiel aufgrund unterschiedlicher Gruppen (Nutzergruppe, Kundengruppe, Rabattgruppe etc.), Typen (Kundentyp, Fahrzeugtyp, Vorgangstyp etc.), Algorithmus-Varianten und -Versionen. Die Varianz in der Domänenlogik *(Strategy)* ist immer Bestandteil einer gemeinsamen Basis *(Service)*, die intern exklusiv eine konkrete Strategie verwendet. Entscheidend ist, dass die Verwendung der Strategie für den Konsumenten, wie zum Beispiel den *Controller*, transparent ist.

Die Frage ist nun, wie eine konkrete Strategie für die Nutzung im *Service* erzeugt wird. Dies kann prinzipiell im *Service* erfolgen. Ist das Finden der Entscheidung für eine Strategie wiederum mit Komple-

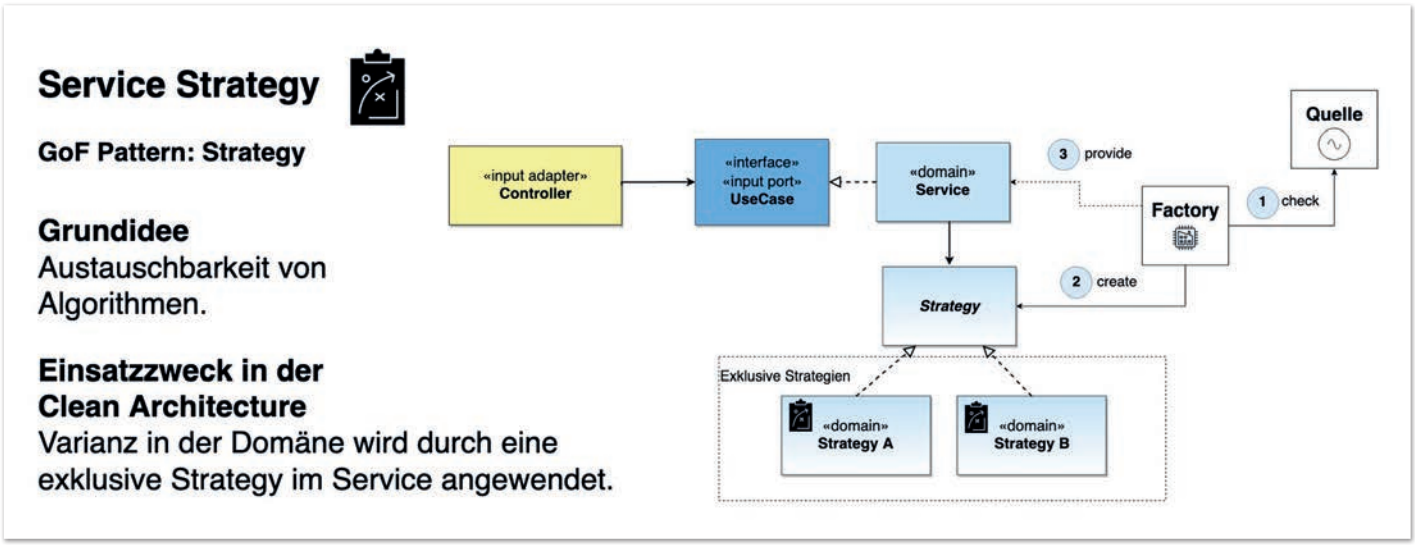



*Abbildung 8: Steckbrief Service Strategy (© Matthias Eschhold)*

xität verbunden und bedingt zusätzlich die Anbindung einer Quelle, empfiehlt sich hierfür wiederum die Verwendung einer Factory. Das erneute Auftauchen der *Factory* innerhalb der *Service Strategy* verdeutlicht zweierlei:

1. Die hier beschriebenen Flexibilitätsmuster haben einen konkreten Kontext. Entweder liegt der Fokus auf Varianz in der Domäne oder auf Stabilität der Domäne, bei Veränderung in der Infrastruktur. Letzteres, also der Kontext, unterscheidet die *Output Adapter Factory* von der *Factory* innerhalb der *Service Strategy*
2. In einem System können unabhängig von den hier vorgestellten Flexibilitätsmustern weitere Entwurfsmuster eine wichtige Rolle spielen

*Abbildung 9* zeigt die *Service Strategy*, angewendet auf das fachliche Beispiel der Fahrzeugbewertung. Ein/e Interessent/in hat den Wunsch, über die Website einfach und schnell einen ungefähren Fahrzeugwert zu ermitteln. Die Schnelligkeit der Berechnung ist dabei wichtiger als die Genauigkeit. Der/die Automobilverkäufer/in muss weitere Faktoren berücksichtigen, um den Fahrzeugwert im Rahmen der Ein- und Verkaufsprozesse von Fahrzeugen exakt zu ermitteln. Der *FahrzeugbewertungsService* implementiert einen eingehenden Use Case der Domäne und stellt die Basisermittlung des Fahrzeugwerts dar. Intern wird dann die *SchnellBewertung* oder die *ExakteBewertung* als Strategie herangezogen.

Die Erfahrung zeigt, je dynamischer die Entscheidung für eine Strategie ist, desto komplexer wird auch die Lösung. Nutzen Interessent/innen und Automobilverkäufer/innen beispielsweise aufgrund von Lastverteilung unterschiedliche Deployments, kann die Quelle für die entscheidende Information eine Konfiguration sein.

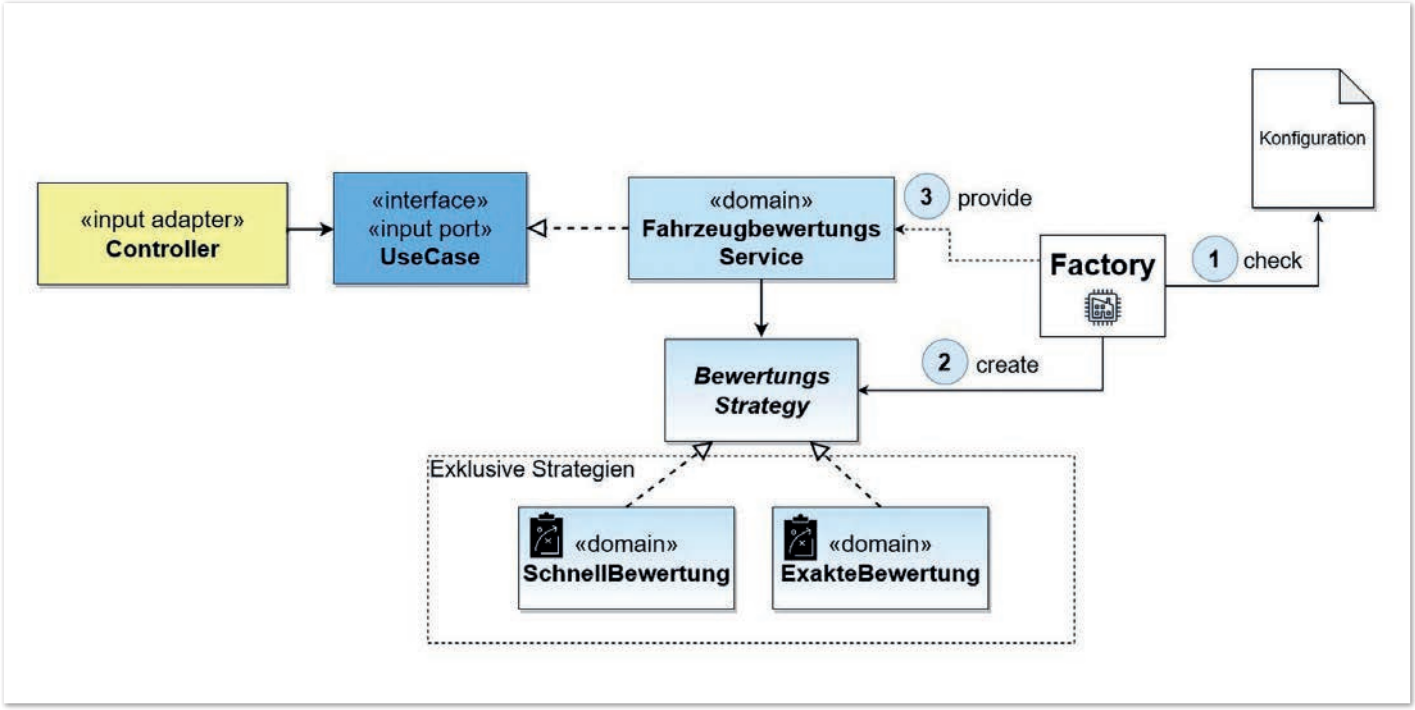



*Abbildung 9: Service Strategy, angewendet in der Fahrzeugbewertung (© Matthias Eschhold)*

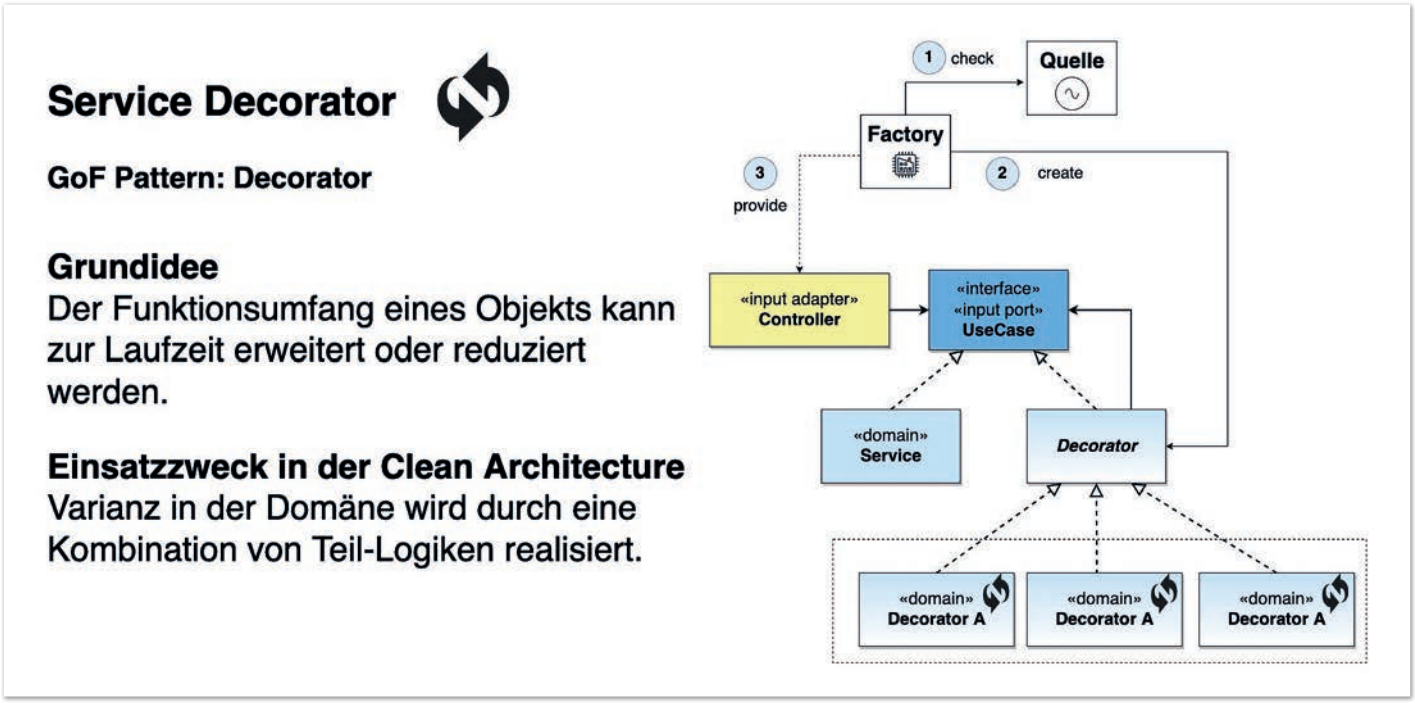



*Abbildung 10: Steckbrief Service Decorator (© Matthias Eschhold)*

Neben der Flexibilität fördert die *Service Strategy* die Wartbarkeit und die Erweiterbarkeit des Systems aus folgenden Gründen:

1. Komplexitätsreduktion und erleichterte Testbarkeit aufgrund der klaren Trennung unterschiedlicher fachlicher Aspekte in Strategien
2. Erweiterbarkeit durch Hinzufügen einer neuen Strategie, ohne hierfür bestehenden Quellcode modifizieren zu müssen

## Service Decorator

Der *Service Decorator* *(siehe Abbildung 10)* behandelt, wie die *Service Strategy*, Varianz in der Domäne. Die Unterschiede zwischen diesen Mustern spiegeln sich in ihren Grundideen wider. Anstatt Austauschbarkeit von Algorithmen ermöglicht der *Decorator*, dass der Funktionsumfang einer Schnittstelle erweitert beziehungsweise reduziert werden kann. Statt eine Varianz exklusiv auszuführen, ermöglicht der *Decorator* Varianz durch eine Kombination von Teil-Logiken. Ein weiterer Unterschied zur *Strategy* ist, dass der Decorator von Konsumenten, das heißt von außen, gesteuert wird.

Neben den Interessent/innen und den Automobilverkäufer/innen wird das System von KFZ-Gutachter/innen verwendet. Eine Fahrzeugbewertung ist im Ergebnis ein Wertgutachten, das aus

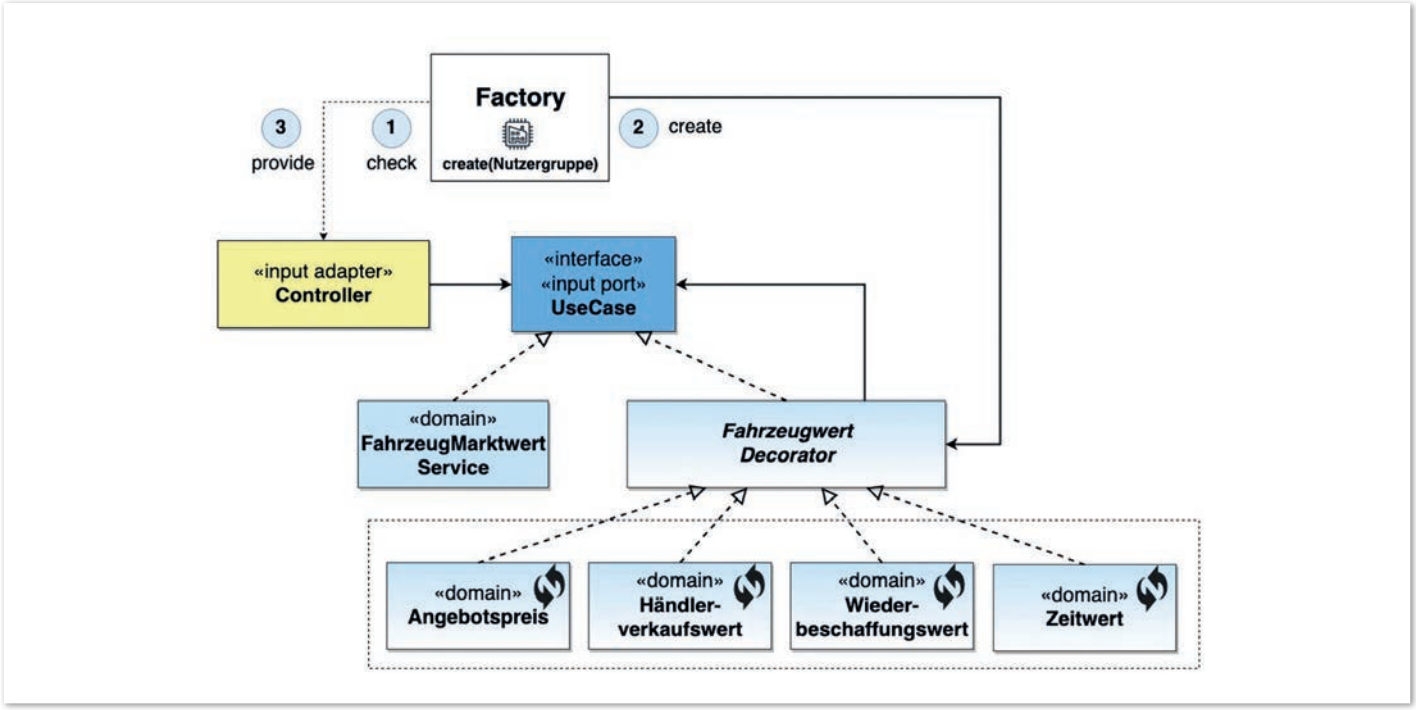



*Abbildung 11: Service Decorator, angewendet in der Fahrzeugbewertung (© Matthias Eschhold)*

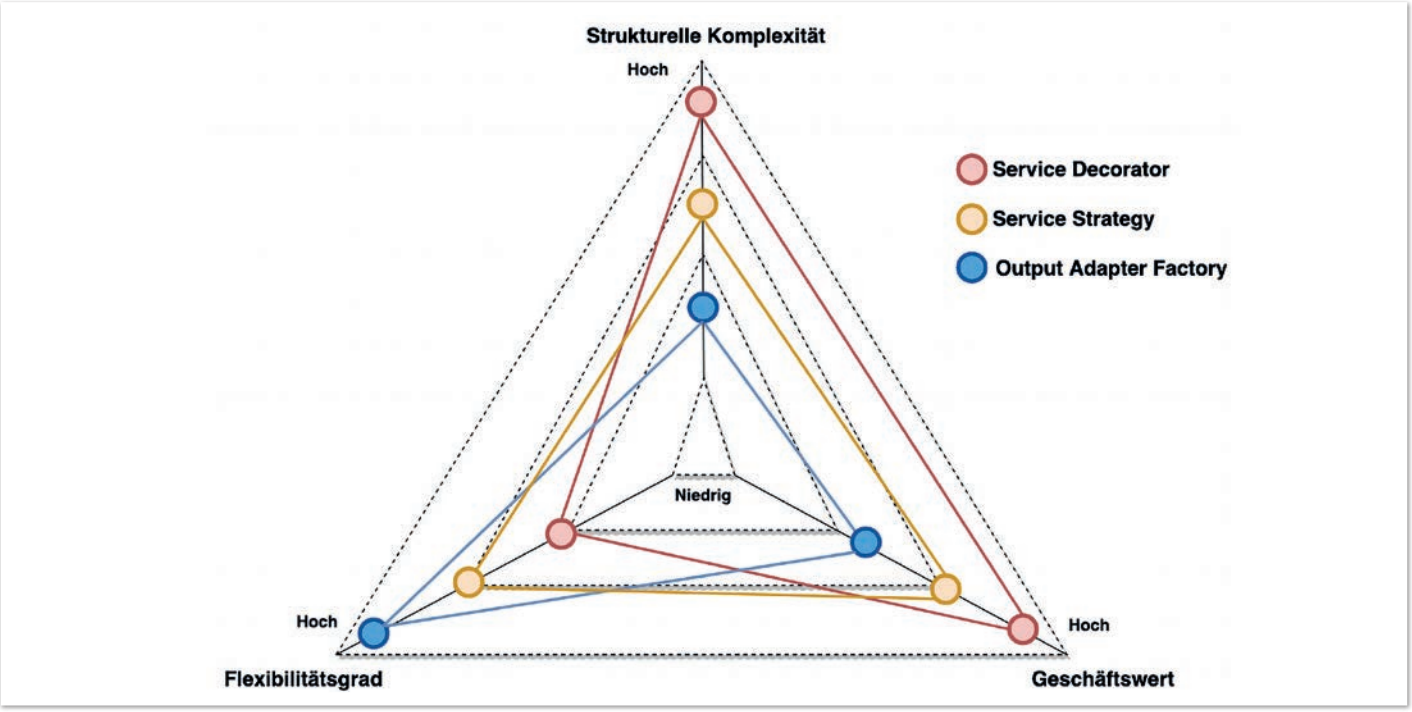



*Abbildung 12: Bewertung der Flexibilitätsmuster hinsichtlich der Dimensionen „Strukturelle Komplexität", „Flexibilitätsgrad" und „Geschäftswert" (© Matthias Eschhold)*

unterschiedlichen Fahrzeugwerten besteht. Fahrzeugwerte sind beispielsweise der Marktwert, der Händlerverkaufswert, der Wiederbeschaffungswert oder der Zeitwert. Für die jeweilige Nutzergruppe sind andere dieser Fahrzeugwerte relevant. Für Flexibilitätsbedarf dieser Art ist der *Service Decorator* das geeignetere Muster. Die Basisfunktion ist die Berechnung des durchschnittlichen Marktwerts. Für Interessent/innen, die ihr Fahrzeug in Zahlung geben möchten, wird ein voraussichtlicher Angebotspreis ermittelt. Für Automobilverkäufer/innen wird zusätzlich der Händlerverkaufswert ermittelt. Für KFZ-Gutachter/innen wird neben dem durchschnittlichen Marktwert ebenfalls der Händlerverkaufswert, der Wiederbeschaffungswert und der Zeitwert benötigt. Dies bedeutet, dass für jede Nutzergruppe die Fahrzeugbewertung anders zu dekorieren ist. Die Basis ist der Marktwert, der mit mehr oder weniger Fahrzeugwerten dekoriert wird *(siehe Abbildung 11)*.

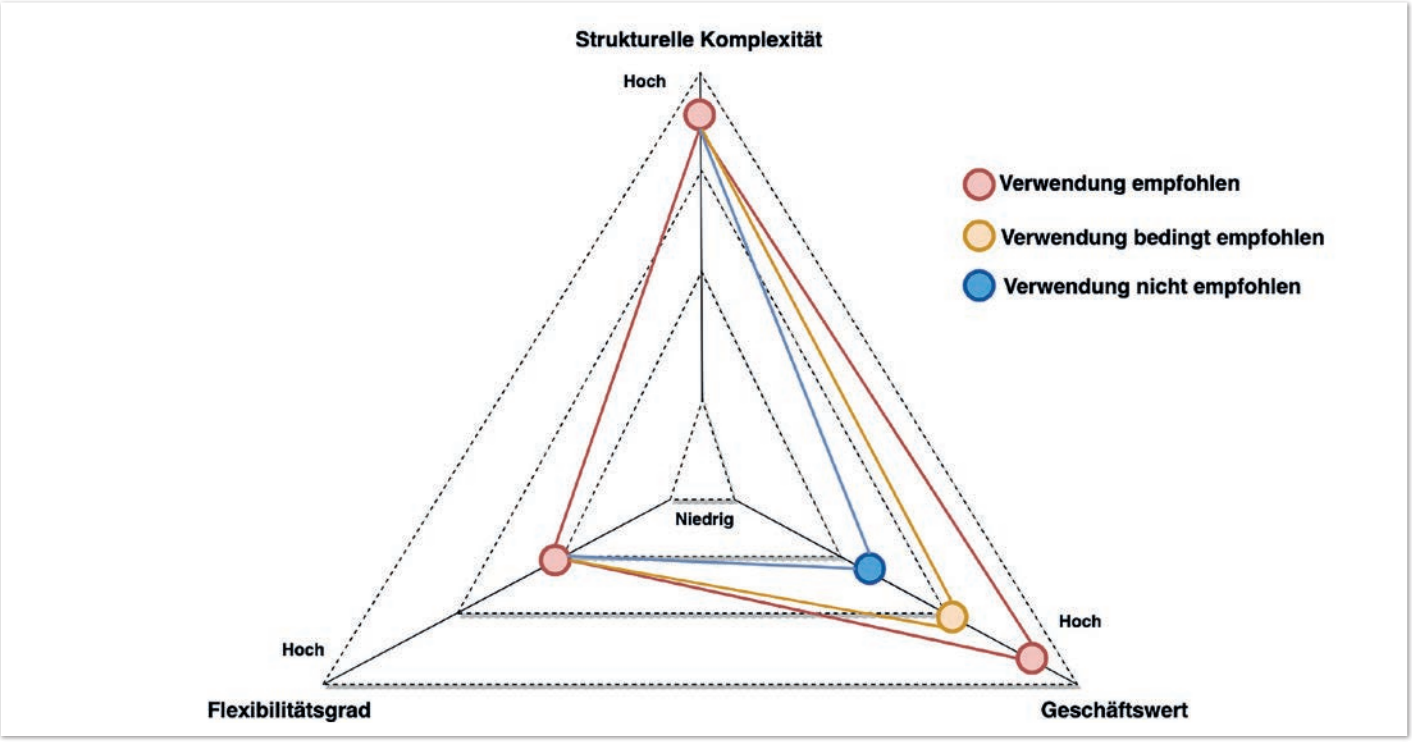



*Abbildung 13: Bewertung des Service Decorator anhand der Dimensionen „Strukturelle Komplexität", „Flexibilitätsgrad" und „Geschäftswert" (© Matthias Eschhold)*

Es wird empfohlen, den Konstruktionsprozess des *Service* und seine Dekoration in eine *Factory* auszulagern. Dadurch reduziert zum einen der *Controller* seine Kopplung zum *Service* und zum anderen wird der Aspekt der Nutzergruppe nicht mit der Fachlichkeit der Fahrzeugbewertung vermischt. Dies reduziert wiederum Komplexität, erleichtert die Testbarkeit und fördert die Wartbarkeit.

## Bewertung der Flexibilitätsmuster

Ob Entwurfsmuster oder Flexibilitätsmuster: Muster haben immer den Charakter, dass die grundlegende Problemstellung einen gemeinsamen Nenner benötigt. Ist dies der Fall, sind Muster sehr effektiv. Dass Entwurfsmuster, insbesondere das *Strategy*-, das *Decorator*- und das *Factory*-Muster dabei helfen, die Anwendungsarchitektur zu flexibilisieren, bestätigen weitere Untersuchungen *[2]*. *Abbildung 12* zeigt die Bewertung der beschriebenen Flexibilitätsmuster hinsichtlich der Dimensionen „Strukturelle Komplexität", „Flexibilitätsgrad" und „Geschäftswert". Je fachlicher und komplexer die Anforderungen sind, desto schwieriger ist es, einen hohen Flexibilitätsgrad in Kombination mit einer geringen strukturellen Komplexität zu erreichen. Allerdings gilt auch die Annahme, dass je komplexer die Anforderung ist, desto höher auch der Geschäftswert ist.

Der Einsatz der Flexibilitätsmuster empfiehlt sich dann, wenn

- ein Geschäftswert mit angemessener struktureller Komplexität erreicht wird
- oder die Qualitätsmerkmale Wartbarkeit, Verständlichkeit und Langlebigkeit des Softwaresystems einen hohen Stellenwert im Projekt genießen.

In allen anderen Fällen sind potenziell einfachere und konkretere Lösungen zu bevorzugen. Dies verdeutlicht exemplarisch die Bewertung des Service Decorator in *Abbildung 13*.

## Fazit und Ausblick

Bis hierhin wurde sehr intensiv Anwendungsarchitektur durchgeführt. In der Implementierung erfordert dies hohe Disziplin und Architekturfokus. Die aufgeführten Anforderungen hinsichtlich Wartbarkeit, Erweiterbarkeit, Verständlichkeit und insbesondere Flexibilität können auf diesem Weg erreicht werden! In Teil 3 werden weitere Muster und Lösungsstrategien vorgestellt, die dazu dienen, die Basisflexibilität zu erhalten und die Architekturaufwände zu minimieren. Teil 3 finalisiert diese Artikelserie und widmet sich dem dritten, noch nicht ausreichend diskutierten Teilmerkmal „Erhalt der Basisflexibilität" des Qualitätsmerkmals Flexibilität.

## Quellen


[1] Muhammad Ehsan Rana et. al. (2020): Impact of Design Principles and Patterns on Software Flexibility: An Experimental Evaluation Using Flexible Point (FXP). Journal of Computer Sience, Online.

[2] A.H. Eden und T. Mens (2006): Measuring software flexibility. IEE Proceedings Software, Online.

[3] Muhammad Ali Babar et al. (2014): Agile Software Architecture. Morgan Kaufmann, Waltham.

[4] Gernot Starke und Peter Hruschka (2016): Kolumne: Knigge für Softwarearchitekten Softwarearchitektur. Online. *https://entwickler.de/software-architektur/kolumne-knigge-fur-softwarearchitekten-002*

[5] Gernot Starke (2015): Effektive Softwarearchitekturen. Hanser Verlag. Heidelberg.

[6] Eric Freeman und Elisabeth Freeman (2006): Entwurfsmuster von Kopf bis Fuß. O'Reilly. Köln.

[7] Matthias Geirhos (2015): Effektive Softwarearchitekturen. Rheinwerk Verlag. Bonn.

[8] *https://github.com/MatthiasEschhold/clean-architecture-and-flexibility-patterns*

[9] Nils Hyoma et al. (2021): Agile Softwareentwicklung in risikoreichen, hochdynamischen und hochkomplexen Umgebungen. JavaSPEKTRUM, 06/21 und 01/22.



**Matthias Eschhold**
Novatec Consulting GmbH
*matthias.eschhold@novatec-gmbh.de*


Warum ist es wichtig, dass Softwarearchitektur aus fachlicher Sicht strukturiert ist? Seit 2014 beantwortet Matthias Eschhold in seinem beruflichen Alltag diese und weitere Fragen zum Architekturentwurf sowie zu Architekturmitteln, Entwurfsprinzipien und -mustern. Sein schönstes Erfolgserlebnis dabei: zu sehen, dass die getroffenen Maßnahmen und Entscheidungen wirken und Ziele wie Wartbarkeit und Verständlichkeit erreicht werden. Darüber hinaus vermittelt Matthias sein Wissen leidenschaftlich als Trainer für Softwarearchitektur und Domain-driven Design.

# Innovationen im Lernen von Java-Grundlagen

Johannes Schildgen, Ostbayerische Technische Hochschule Regensburg

*Ob in der Schule, im Studium oder Selbststudium: Die Möglichkeiten und Techniken, Java zu lernen, haben sich in den vergangenen Jahren entscheidend weiterentwickelt. Dieser Artikel gibt einen Einblick in innovative Möglichkeiten des Java-Lernens sowie einen Ausblick, der zeigt, dass sich nicht nur Programmiersprachen weiterentwickeln, sondern auch der Weg, wie wir sie lernen und lehren.*

## Wie lernt man 2022 eine Programmiersprache?

Gibt man in einer Suchmaschine „Java lernen" ein, muss man erst einige Ergebnisse nach unten scrollen, bis man eine Liste von Literaturempfehlungen findet *[1]*. Dabei war das Erlernen einer Programmiersprache mit einem Buch bis vor wenigen Jahren noch der Standard. Zwar ist dies auch heute immer noch ein gängiger Weg und es erscheinen ständig neue gute Lehrbücher, jedoch zeigt sich anhand der Suchergebnisliste auch, dass vor allem das Angebot an Online-Kursen, Live-Workshops und Videomaterialien stark dominiert. Diese sind im Wesentlichen so ähnlich gestaltet wie eine Informatik-Vorlesung an einer Hochschule oder Universität: Theoretische Inhalte werden vermittelt und durch Programmcodebeispiele veranschaulicht. Danach probieren die Lernenden alles in praktischen Übungen noch einmal selbst aus *[2]*. In diesem Artikel möchte ich auf genau diese Elemente eingehen und neuartige Möglichkeiten zeigen, die sie mit sich bringen.

## Präsentationsfolien mit Code

Ich habe mal gelernt: „Schreibe niemals Programmcode auf Folien!" An diese Regel halten sich aber leider viel zu wenige Lehrende und andere Vortragende. Ich will damit nicht sagen, dass Programmcode auf Folien nichts zu suchen hat – ganz im Gegenteil: Programmcode ist wichtig! Aber man sollte den Code nicht direkt in PowerPoint und Co. eintippen, sondern ihn zuerst in einer IDE wie IntelliJ formulieren und ausprobieren und erst danach auf die Folie kopieren und einfügen. Das hat den Vorteil, dass nur gültiger Programmcode ohne Syntax-Fehler auf den Folien landet, der genau das tut, was er soll. Des Weiteren unterstützt PowerPoint kein automatisches Syntax-Highlighting. Beim Kopieren und Einfügen jedoch kann PowerPoint in der Regel die von der IDE vorgenommene Formatierung direkt übernehmen *(siehe Abbildung 1)*. Alternativ lässt sich der Programmcode auch mithilfe eines Online-Tools *[3]* formatieren und anschließend in PowerPoint kopieren.

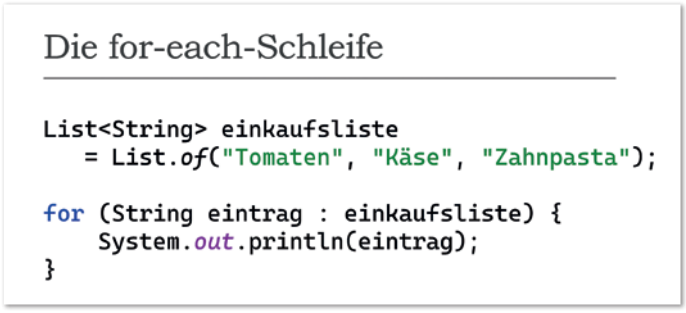



*Abbildung 1: Programmcode, der auf eine PowerPoint-Folie aus IntelliJ heraus eingefügt wurde (© Johannes Schildgen)*

Noch mehr Möglichkeiten als PowerPoint bieten alternative Präsentationstools wie LaTeX Beamer oder reveal.js. Beamer *[4]* ist eine Klasse, mit der sich eine Präsentation textuell in LaTeX-Syntax erstellen lässt. Diese wird schließlich zu einer PDF-Datei kompiliert, die sich in einem beliebigen PDF-Viewer öffnen lässt. Wegen der hervorragenden Unterstützung für mathematische Formeln ist LaTeX Beamer vor allem in den MINT-Fächern beliebt. Auch zur Präsentation von Programmcode existieren Pakete und Kommandos, die eine optisch ansprechende Darstellung mit automatischer Syntax-Hervorhebung ermöglichen. In LaTeX ist es auch möglich, Programmcode direkt aus einer externen Datei zu laden. *Listing 1* zeigt, wie sich die Zeilen 2 bis 4 einer Java-Datei, die sich in einem Unterordner befindet, auf einer Folie einblenden lassen.

```
\lstinputlisting[language=java, linerange={2-4}]{code/
Main.java}
```

*Listing 1: Einfügen von Code aus einer Datei in eine LaTeX-Beamer-Präsentation*

Dieses Vorgehen kommt dem Grundsatz „Niemals Programmcode auf Folien" noch näher: Jetzt wird nur noch auf den Programmcode verwiesen. Nachträgliche Änderungen an diesem wirken sich beim nächsten Kompilieren direkt auf die Präsentationsfolien aus.

Eine Alternative zu PowerPoint und LaTeX Beamer sind JavaScript-basierte Präsentationsframeworks. Mit solchen Frameworks lassen sich Präsentationen in HTML erstellen und diese dann im Webbrowser betrachten. Eines dieser Frameworks nennt sich reveal.js *[5]*. Es unterstützt Animationen und das schrittweise Einblenden von Folienelementen, wie man es aus PowerPoint und Co. kennt. Analog zu LaTeX Beamer wird auch in reveal.js Programmcode mittels Syntax-Highlighting optisch hervorgehoben und er lässt sich direkt aus externen Dateien einlesen *[6, 7] (siehe Listing 2)*.

```
<pre><code class="java" contenteditable data-
sample="code/Main.java#ungerade_test"></code></pre>
```

*Listing 2: Einfügen von Code aus einer Datei in eine reveal.js-Präsentation*

Wie in *Listing 2* zu sehen ist, wurden hier nicht mittels #2-4 bestimmte Zeilen mit entsprechenden Zeilennummern extrahiert, sondern diejenigen Zeilen, die zwischen zwei Sample-Kommentaren im Programmcode stehen *(siehe Listing 3)*.

```
// sample(ungerade_test)
Predicate<Integer> ungerade = n -> n%2 == 1;
if(ungerade.test(17)) {
  System.out.println("17 ist ungerade!");
}
// end-sample
```

*Listing 3: Sample-Markierungen im Java-Programmcode*

Da die Darstellung im Browser erfolgt, sind anders als bei PDF-Präsentationen deutlich mehr Interaktionen möglich. Das Attribut `contenteditable` bedeutet, dass sich der auf der Folie gezeigte Code live verändern lässt. Dies ermöglicht Demonstrationen und Interaktion mit den Zuhörern ohne ständigen Wechsel zwischen Präsentationstool und IDE. Mithilfe eines Plug-ins lässt sich sogar ein Terminal auf den Folien einfügen, um den Code direkt auszuführen. Und mit JavaScript kann reveal.js beliebig erweitert werden, um beispielsweise den auf den Folien befindlichen Programmcode an einen Serverprozess zu senden, der das


*Abbildung 2: reveal.js-Folien werden im Browser betrachtet und können Code-Beispiele aus externen Dateien laden (© Johannes Schildgen)*

Programm kompiliert, ausführt und das Ergebnis direkt auf der Folie einblendet.

Andere Plug-ins für reveal.js erzeugen UML-Diagramme aus einer textuellen PlantUML-Notation, die sich direkt im Programmcode befindet, oder Multiple-Choice-Umfragen, um das Publikum einzubeziehen.

## Java ausprobieren

Ein „Hallo Welt" kann einem Programmieranfänger auch in der heutigen Zeit noch ein Lächeln ins Gesicht zaubern. Denn ein „Hallo Welt" auf dem Bildschirm zeigt, dass bis hier hin schon mal alles geklappt hat. Die zum Programmieren notwendige Software wurde erfolgreich installiert und konfiguriert, die IDE entsprechend eingerichtet. Vor allem in der Programmiersprache Java ist zur Ersteinrichtung mehr nötig und man kann mehr falsch machen als beispielsweise in Python oder JavaScript. Umso wichtiger ist es also, eine gute und aktuelle Anleitung für das jeweilige Betriebssystem und die Entwicklungsumgebung seiner Wahl zur Verfügung stehen zu haben, sei es textuell oder in Form eines Videos.

Viel schneller kommt man zum Erfolg, wenn man eine Browser-basierte IDE verwendet. Auf den Websites online-java.com und rextester.com lässt sich Java-Code in ein Eingabeformular eingeben und sofort im Browser ausführen. Die Ausführung erfolgt dabei nicht auf dem Client-Rechner, sondern auf einem Server in der Cloud. Nach wenigen Sekunden wird die Konsolenausgabe des Programms angezeigt. Die genannten Tools eignen sich auch hervorragend zum kollaborativen Arbeiten. Mithilfe eines individuellen Links kann der Programmcode anderen Personen geschickt werden, die ihn ebenfalls direkt im Browser modifizieren und ausführen können.

Replit *[8]* verfolgt einen ähnlichen Ansatz, es gleicht jedoch von der Optik und von den Funktionen her einer nahezu voll ausgestatteten IDE: Ein Projekt-Explorer, Terminal-Fenster, Versionsverwaltung, Debugging, Paketmanagement und Unit-Tests sind in Replit fest eingebaut und können sofort genutzt werden. Auch ist es möglich, Projekte vorzubereiten und Lernenden zur Verfügung zu stellen. Dann fangen diese nicht bei null an, sondern beginnen mit vordefinierten Klassen. Sie können diese direkt ausprobieren und erweitern.

Dadurch, dass sich neben Java-Klassen auch beliebige andere Dateien einem Replit-Projekt hinzufügen lassen, kann man Replit beispielsweise um Konsolenanwendungen erweitern, die den Programmcode testen, etwa daraufhin, ob eine Aufgabe richtig gelöst wurde. Auch ist es dadurch möglich, den Programmcode per Kommandozeilenbefehl online abzugeben, damit dieser – automatisch oder händisch – bewertet und kommentiert werden kann.

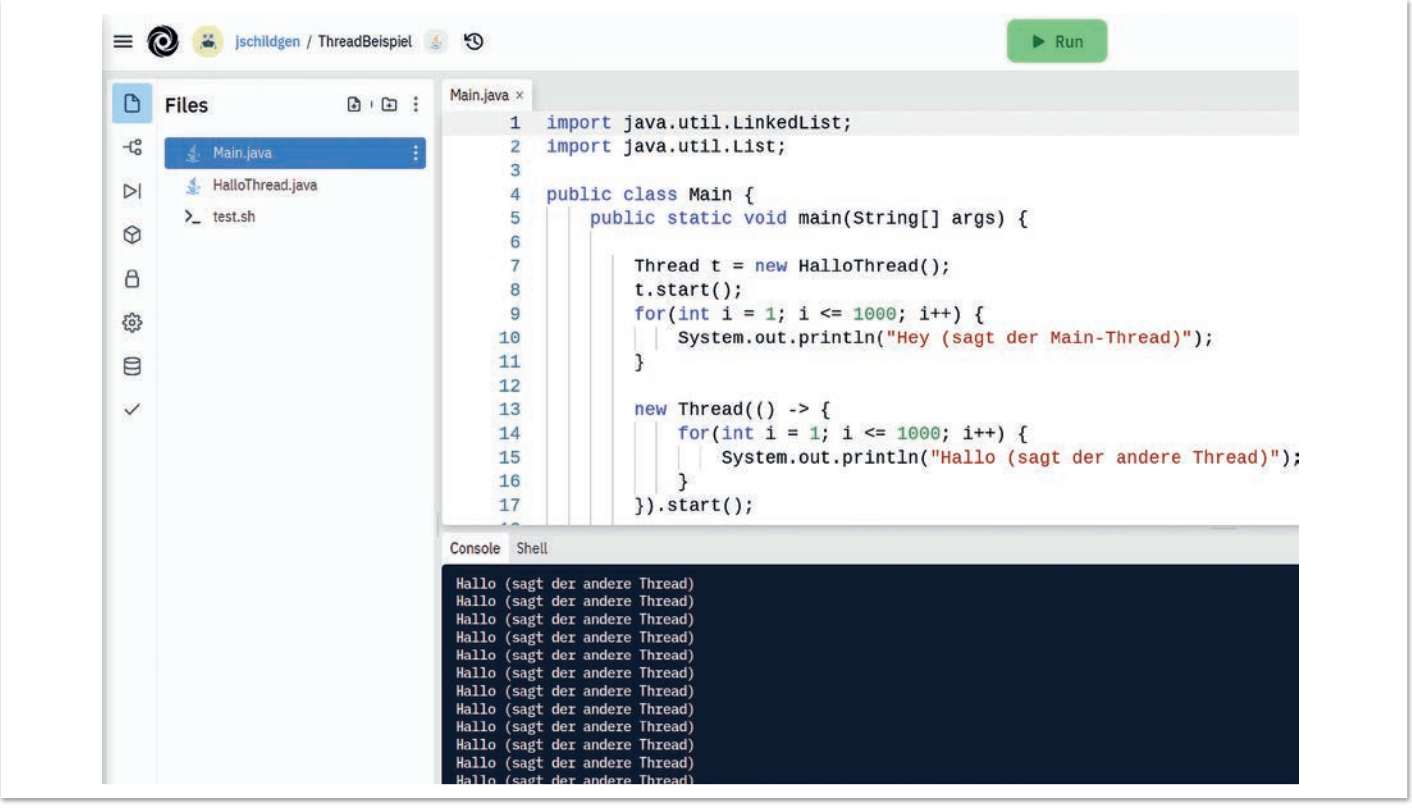



*Abbildung 3: Replit: Eine vollständige IDE im Browser. (© Johannes Schildgen)*

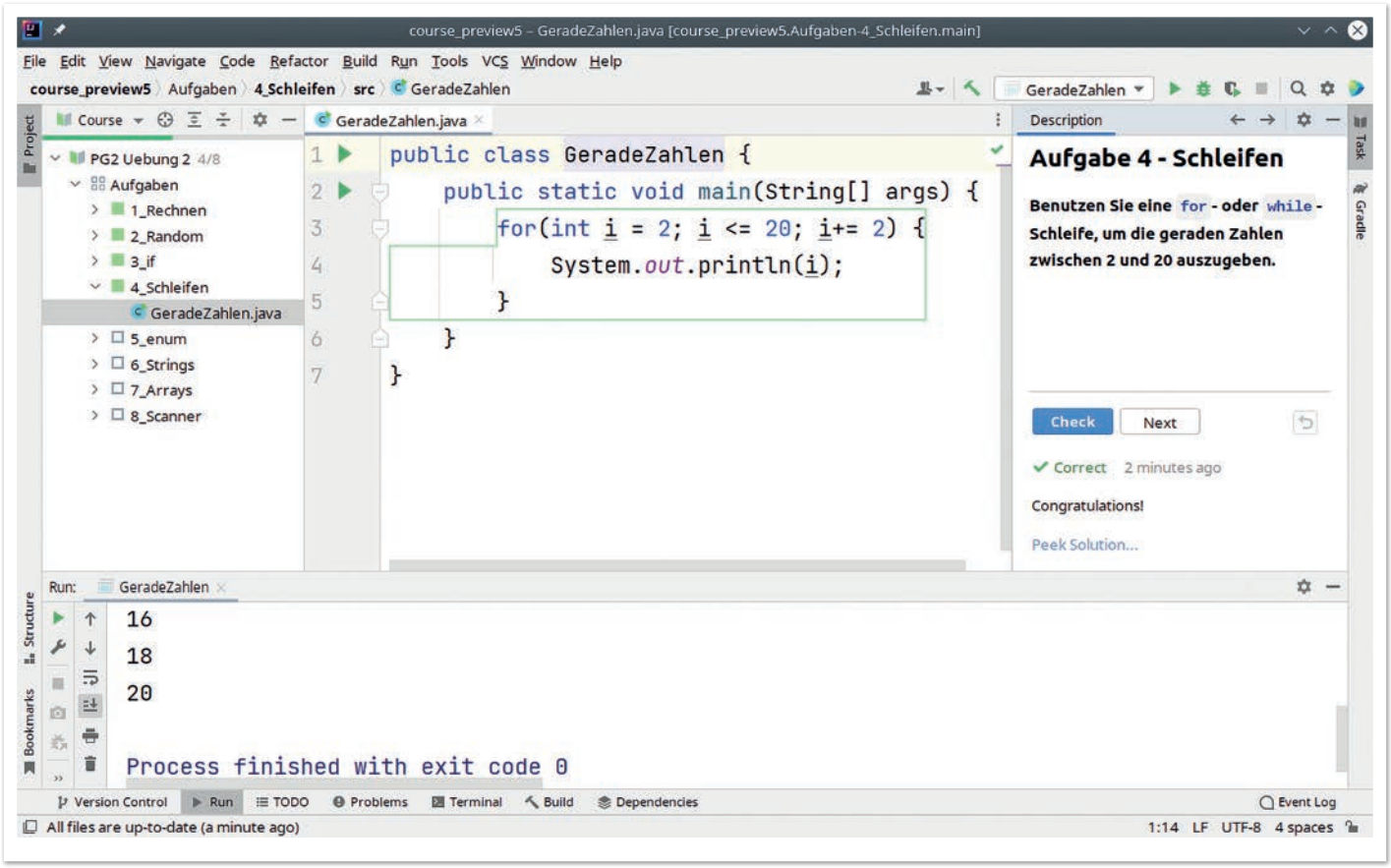



*Abbildung 4: Das EduTools-Plug-in für IntelliJ prüft automatisch, ob eine Programmieraufgabe korrekt gelöst wurde. (© Johannes Schildgen)*

## EduTools

Die im vorherigen Abschnitt erwähnten Möglichkeiten, Programmcode zu validieren, sind in Replit nur über manuell entwickelte Zusatzsoftware möglich. Für die IDE IntelliJ (und auch für alle anderen IDEs von JetBrains) gibt es jedoch ein Plug-in, das genau für diesen Zweck gemacht ist. Es lässt sich entweder nachträglich über den Plug-in-Explorer installieren, indem man nach „EduTools" sucht, oder man installiert direkt die sogenannte IntelliJ-Edu-Edition. Diese basiert auf der kostenlosen Community-Edition und hat das EduTools-Plug-in bereits vorinstalliert.

Mit EduTools *[9]* können Lehrende einen Kurs erstellen und diesen auf einer Online-Lernplattform den Lernenden zur Verfügung stellen. Als Plattformen stehen aktuell der JetBrains Marketplace, Stepik, Coursera, CheckiO und die JetBrains Academy zur Wahl. Einige dieser Plattformen sind offen, sodass jeder dort Kurse publizieren kann. Alternativ kann ein Kurs auch als Zip-Datei exportiert werden, die Lernende dann über eine Import-Funktionalität öffnen können.

Ein Kurs besteht aus sogenannten Lessons, und in jeder Lesson gibt es eine Folge von Tasks. Im Folgenden wird auf zwei Arten von Tasks detailliert eingegangen. Solche, die den Output eines Programms mit einer Musterlösung vergleichen, und solche, die über eine Test-Klasse validiert werden. Als weitere Task-Typen bietet EduTools auch Multiple-Choice, IDE-Übungen und reine Theorie-Aufgaben ohne Validation an.

Sowohl beim Aufgabentyp „Output" als auch beim Typ „Edu" sind die zentralen Elemente eine Java-Paketstruktur mit Klassen und Interfaces, eine Markdown- oder HTML-Datei mit einer Aufgabenbeschreibung sowie ein Ordner „test", der später für die Lernenden unsichtbar gemacht wird. Am einfachsten ist es, die Aufgabe zunächst in Form einer Musterlösung vollständig zu implementieren und im Anschluss ganze Code-Bereiche in sogenannte Placeholder umzuwandeln. Diese Platzhalter erscheinen später für den Lernenden als leere Kästchen, die es mit Programmcode zu füllen gilt.

Beginnen Lernende eine Aufgabe *(siehe Abbildung 4)*, sehen sie am rechten Bildschirmrand die Aufgabenbeschreibung. Dadurch, dass diese wahlweise in Markdown oder HTML formuliert werden kann, lassen sich optisch ansprechende Formatierungen sowie Code-Blöcke und Bilder verwenden. Auf der linken Seite des Editors ist wie gewohnt die Projektstruktur zu sehen, sodass sich alle Klassen und Interfaces im Editor öffnen lassen, Lernende können sogar weitere neue Dateien erstellen.

Um die jeweilige Aufgabe zu lösen, werden die im Code befindlichen Platzhalter mit eigenen Statements gefüllt. Daraufhin kann das Programm per Play-Button regulär ausgeführt werden. Zusätzlich bietet jede Aufgabe einen „Check"-Button, der im Hintergrund eine Reihe von Tests ausführt und schließlich zurückmeldet, ob die Aufgabe korrekt gelöst wurde oder nicht.

Beim Aufgabentyp „Output" wird nach der Ausführung des zu entwickelnden Java-Programms überprüft, ob dieses eine Konsolenausgabe erzeugt, die exakt der Musterlösung entspricht. Dazu wird vom Kursersteller eine Datei test/output.txt bereitgestellt, die den erwarteten Output enthält.

Komplexere Aufgaben mit individuelleren Feedback-Möglichkeiten sind mit dem Aufgabentyp „Edu" möglich. Hier bestimmt eine Klas-

se test/Tests.java, wann eine Aufgabe als gelöst gilt. Konkret wird dies über einen JUnit-Test definiert. Mittels `Assert.assertTrue`, `Assert.assertEquals`, `AssertThrows` etc. können so die Klassen und Methoden, die die Lernenden entwickeln sollen, auf ihre Richtigkeit überprüft werden. Sind alle Assertions erfüllt, wird die Aufgabe als korrekt gelöst gewertet. Scheitert eine Assertion, sehen die Lernenden eine entsprechende Feedback-Meldung, die der jeweiligen `assert`-Methode übergeben wurde.

Wie oben erwähnt, bündelt man mehrere Tasks zu einer Lesson. In einer besonderen Art von Lessons – sogenannten Framework-Lessons – können Platzhalter automatisch mit dem Inhalt von Platzhaltern einer vorherigen Aufgabe belegt werden. Dies ermöglicht es, Aufgaben aufeinander aufzubauen und größere Probleme schrittweise lösen zu lassen.

Das EduTools-Plug-in verfolgt das Prinzip der Selbstkontrolle. Es gibt standardmäßig keinen Rückkanal zum Lehrenden. Lösungen können nicht abgegeben, eingereicht oder bepunktet werden. Ähnlich wie bei Replit lässt sich dies jedoch auch hier ermöglichen, indem beispielsweise ein Konsolenprogramm oder eine Java-Klasse einer Aufgabe beigelegt wird, die die Lösungen der Lernenden oder lediglich die Information, ob die Aufgabe erfolgreich gelöst wurde, an einen Server übermittelt. Es ist sogar möglich, diese Übermittlung direkt in der JUnit-Klasse durchzuführen, die den Code auf seine Richtigkeit überprüft. Um die Abgabe später den Lernenden zuordnen zu können, kann beispielsweise von diesen verlangt werden, ihren Namen oder eine ID in eine Datei oder in eine Klassenkonstante zu schreiben.

## Sprachkurs Java – Das Hörbuch über objektorientierte Programmierung

Die bisher gezeigten Ansätze können als Erweiterungen für Präsentationssoftware sowie IDEs gesehen werden. Einen ganz anderen Ansatz verfolgt das Hörbuch „Sprachkurs Java“ *[10]*. Das in diesem Jahr erschienene Hörbuch mit einer Laufzeit von etwas mehr als zehn Stunden vermittelt Java-Grundlagen vom if-Statement über Exception-Handling bis hin zu Lambda-Ausdrücken und Threads. Zudem gibt es einen Ausblick in die Entwicklung von Android-Apps, GUI- und Web-Anwendungen.

Bisher wurde sich an das Thema Hörbücher zu Programmiersprachen nicht herangetraut, weil sich die meisten klassischen Fachbücher nicht dazu eignen, vorgelesen zu werden. Daher gibt es Sprachkurs Java nur als reines Hörbuch. Alle Code-Beispiele werden schrittweise erarbeitet und erklärt, sie lassen sich aber auch vollständig auf sprachkurs-java.de nachschlagen. Ein Element, das im Hörbuch einzigartig ist, sind interaktive Quiz. Immer mal wieder wird dem Hörer oder der Hörerin eine Frage gestellt und dann eine kurze Pause gemacht, in der diese beantwortet werden soll.

Das Hörbuch ist geeignet, um während der Autofahrt, in der U-Bahn oder während der Hausarbeit Java zu lernen. Natürlich ist es wichtig, das Gelernte auch am PC noch selbst auszuprobieren. Dies gilt aber bei Büchern, Videos und Vorträgen genauso.

## Fazit

In diesem Artikel wurden neuartige Möglichkeiten vorgestellt, die das Lehren und Lernen von Java-Grundlagen modern, interaktiv und abwechslungsreich gestalten. Es wurden Ansätze gezeigt, um Code adäquat zu präsentieren. Dazu bringen moderne LaTeX- oder HTML-basierte Präsentationsframeworks nicht nur Syntax-Highlighting-Features mit, sondern auch die Möglichkeit, Code direkt aus Dateien zu laden oder ihn sogar live auszuführen. Um einen schnellen Start in die Entwicklung eigener Java-Anwendungen zu ermöglichen, gibt es Online-IDEs wie Replit sowie das IntelliJ EduTools-Plug-in. Mit Letzterem können ganze Aufgaben vordefiniert werden, die sich automatisch durch JUnit-Tests überprüfen lassen. Das Hörbuch Sprachkurs Java kann als Alternative zu einem Lehrbuch gesehen werden, jedoch kann man es ganz nebenbei hören, selbst bei der Autofahrt.

Natürlich ersetzen all diese Tools keine herkömmlichen Lehrmethoden, sie ergänzen sie vielmehr. Ein bunter Mix aus Büchern, Hörbüchern, Präsentationen, Videos und ganz viel Selbst-Ausprobieren ist immer noch das Wichtigste, um erfolgreich Java zu lernen.

## Quellen


[1] Java Aktuell (2021): Java-Bücher – Programmieren mit Java – Einführung und fortgeschrittene Bücher *https://www.informatik-aktuell.de/aktuelle-meldungen/2021/maerz/java-buecher-programmieren-mit-java.html*

[2] Antonio Bello (2017): Learning Techniques for Programmers, by Programmers *https://www.raywenderlich.com/467-learning-techniques-for-programmers-by-programmers*

[3] Java Formatter *https://jsonformatter.org/java-formatter*

[4] Beamer – A LaTeX class for producing presentation *https://github.com/josephwright/beamer/*

[5] Hakim El Hattab: reveal.js – The HTML presentation framework *https://revealjs.com/*

[6] Reveal Sampler *https://opensourcelibs.com/lib/reveal-sampler*

[7] Java-Vorlesungsfolien mit reveal.js *https://github.com/jschildgen/java-slides*

[8] *Replit https://replit.com*

[9] JetBrains EduTools *https://plugins.jetbrains.com/plugin/10081-edutools*

[10] Johannes Schildgen (2022): Sprachkurs Java – Das Hörbuch über objektorientierte Programmierung



**Prof. Dr. Johannes Schildgen**
OTH Regensburg
*johannes.schildgen@oth-regensburg.de*


Prof. Dr. Johannes Schildgen ist Professor für Datenbanken und Big-Data-Analytics an der Ostbayerischen Technischen Hochschule Regensburg. Neben seiner hauptberuflichen Tätigkeit ist er als Datenbank-Experte, Fachautor und Keynote-Speaker tätig. Wenn's um Programmieren geht, ist Java seine Leidenschaft. Neben den Java-Programmiervorlesungen, die er an der Hochschule hält, ist er auch Autor des Hörbuchs „Sprachkurs Java“.

# Wie skaliert man Agilität? Eine Zeitreise

Edwin Günthner, IBM Deutschland Research & Development GmbH

*Dieser Artikel zeigt auf, wie sich die Organisation von Softwareentwicklung historisch entwickelt hat. Er vermittelt, welche Methoden heute zur Verfügung stehen, um jene Techniken, die im Kleinen so gut funktionieren, erfolgreich auf größere Teams zu übertragen.*

## Worum geht es hier?

Agilität steht hier für die breite Palette an Methoden, die alle auf jenen Haltungen basieren, wie sie beispielsweise im Manifest für agile Softwareentwicklung *[1]* zum Ausdruck gebracht werden. Diese Methoden zeichnen sich dadurch aus, dass sie in der Praxis gut funktionieren: Teams, die einem gut eingespielten Scrum-Prozess folgen, sind in der Lage, kontinuierlich hohe Qualität zu liefern und flexibel auf Änderungen zu reagieren. Ebenso real ist aber auch die Erkenntnis: Was gut für ein Start-up mit 20 Entwicklern funktioniert, taugt nur bedingt für Firmen, die mit Tausenden Entwicklern an historisch gewachsenen Produkten arbeiten. Um genau dieses Spannungsfeld soll es hier gehen.

## Sprung in die IT-Steinzeit: Worum ging es damals?

Antiquiert formuliert lautet die Antwort:

- „The problems of achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society,
- the difficulties of meeting schedules and specifications on large software projects"

Diese Zeilen stammen aus dem Report *[2]* zur NATO-Konferenz „Software Engineering" aus dem Jahr 1968 und markieren den Beginn der systematischen Auseinandersetzung mit der Softwarekrise; jenem ärgerlichen Umstand, dass die Kosten für die Entwicklung der Software schon damals die Kosten für die Anschaffung der Hardware überholt haben.

Ein bekannter früher Ansatz zum Umgang mit diesem Problem wurde 1970 von Winston Royce veröffentlicht. Sein Artikel „Managing the Development of Large Software Systems" *[3]* beschreibt die Entwicklung von Software in verschiedenen Phasen und wird häufig als eine der Grundlagen dessen gesehen, was wir heute als Wasserfallmodell kennen. Bemerkenswerterweise hat Royce bereits damals erkannt, dass die strikte Einteilung in Phasen riskant und fehleranfällig ist. Er schlägt deshalb zum Beispiel vor, Kunden möglichst früh einzubinden: „It is important to involve the customer in a formal way so that he has committed himself at earlier points before final delivery. To give the contractor free rein between requirement definition and operation is inviting trouble."

Diese frühen Einsichten hatten jedoch wenig Einfluss auf die schnelle und weite Verbreitung des Wasserfallmodells in der Softwareentwicklung. Die Bemühungen zur Standardisierung durch Behörden der US-Regierung, wie dem DOD-STD-2167 aus dem Jahr 1985, legten die Grundlage dafür, dass Softwareentwicklung fast über Jahrzehnte hinweg untrennbar mit dem Wasserfallmodell verknüpft war. Das ändert allerdings nichts an der frühen Erkenntnis von Royce: Beim Wasserfallmodell besteht ein signifikantes Risiko, nicht das zu liefern, was der Kunde haben will.

Ein Versuch zur Lösung dieses Problems wurde 1979 von Barry Boehm zum ersten Mal vorgestellt: das V-Modell. In diesem Entwicklungsmodell werden ganz dediziert zwei Aspekte betrachtet: Validieren (machen wir das Richtige?) und Verifizieren (machen wir es richtig?). In Deutschland wurde dieser Ansatz früh vom Bundesministerium für Verteidigung aufgegriffen und eigenständig weiterentwickelt. Inzwischen erfolgt die Weiterentwicklung durch den Beauftragten der Bundesregierung für Informationstechnik (CIO Bund). Übrigens, die aktuelle Version 2.3 des V-Modells XT *[4]* wurde im Mai 2019 veröffentlicht und erlaubt inzwischen auch eine iterative Vorgehensweise. Es ist daher ein Irrtum zu glauben, dass diese Ansätze heute ausschließlich aus „historischem" Interesse Relevanz besitzen.

Nichtsdestotrotz, auch die erwähnten Regierungsbehörden der USA haben frühzeitig erkannt, dass penibles Festhalten an starren Regeln schnell zu hoher Inflexibilität führen kann. Und dass das Problem der hohen Kosten durch das Wasserfallmodell keinesfalls gelöst wurde. Ein bekanntes Paradebeispiel für „Wasserfall"-Software ist der Code für das Space-Shuttle der NASA. 17 Fehler in 420.000 Zeilen Code ist vermutlich ein Qualitätsrekord *[5]*. Aber wer kann oder will über 1.000 US-Dollar pro Zeile Code ausgeben *[6]*?

Aus der Kostenproblematik ergibt sich eine weitere spannende Frage: Wenn die Entwicklung so viel kostet, wie kann man dann vor der potenziellen Vergabe von Aufträgen das Risiko abschätzen, dass der Lieferant in der Lage ist, die gewünschte Software zum geplanten Termin, mit der gewünschten Qualität und innerhalb eines bestimmten Kostenrahmens zu entwickeln?

## Ein neuer Blickwinkel

Und wieder kam eine prägende Antwort aus den USA. Am Software Engineering Institute (SEI) der Carnegie Mellon University (CMU) wurde ab Mitte der 1980er Jahre am Capability Maturity Model (CMM) gearbeitet. Ziel war es, den Reifegrad eines Unternehmens, das Software entwickelt, zu verstehen und zu verbessern. Wer sich bisher keine Gedanken über seinen Entwicklungsprozess gemacht hat, wird sich in diesem Modell auf Stufe 1 „Initial" finden. Durch geeignete Maßnahmen kann eine Firma sich Stufe für Stufe verbessern, um am Ende idealerweise die (utopische) Stufe 5 „Optimizing" zu erreichen. Mit jeder Stufe steigen dabei Wissen und Kontrolle über den eigenen Entwicklungsprozess, bis hin zum Punkt der kontinuierlichen Verbesserung aller Abläufe. Wir sehen: Die Idee des beständigen Nachdenkens „Was tun wir eigentlich?" und „Wie könnten wir es besser machen?" ist mitnichten eine Erfindung des neuen Jahrtausends und des agilen Manifests.

Doch woran liegt es dann, dass wir heute in Blogs, Magazinen und auf Konferenzen zwar von Agilität und Retrospektiven hören, aber nur selten im Zusammenhang mit CMM?

Meiner Meinung nach liegt eine der Ursachen darin, dass diese Methoden zwar Werkzeuge für Entscheider und Projektmanager liefern, jedoch dem einzelnen Entwickler keine Hilfestellung bieten, um einen guten Job zu machen. Wenig überraschend: Auch dieses Manko wurde nicht erst gestern entdeckt. Am erwähnten SEI entstanden unter Watts Humphrey in den 1990er Jahren der „Personal Software Process" (PSP) und später der „Team Software Process" (TSP). Ziel war konkret, für einzelne Entwickler beziehungsweise Entwicklerteams Methoden bereitzustellen, die es ermöglichen, die verschiedenen Aufgaben, wie Entwurf, Implementierung und Testen, besser zu organisieren und dadurch verlässlich planbar zu machen. In der Theorie spannende und interessante Modelle, die jedoch in der Praxis – zumindest im deutschsprachigen Raum – keine nennenswerte Nutzung generieren konnten.

Speziell der PSP zeichnet sich durch einen hohen Grad an Formalismus auf. Da werden „Lines of Code" abgeschätzt und später gezählt, immer wieder Checklisten erstellt und erweitert und in erheblichem Aufwand Statistik betrieben. Konsequent umgesetzt kann man damit klar formulierte Programmieraufgaben mit hoher Qualität lösen.

## Der Übergang in die Neuzeit

Aber diejenigen Fragen, die sich in einem realen Arbeitsumfeld stellen, wurden auch vom PSP nicht beantwortet. Glücklicherweise begannen Kent Beck, Martin Fowler und andere Praktiker ebenfalls Mitte der 1990er Jahre an dem zu arbeiten, was als „Extreme Programming" (XP) die Sicht auf Softwareentwicklung in gänzlich andere Bahnen lenkte. Auch XP beinhaltet formale Techniken, im Mittelpunkt aller Aktivitäten steht allerdings der Mensch. Das bedeutet: Basierend auf gemeinsamen Werten suchen sich einzelne Entwickler und Teams diejenigen Methoden, die es ihnen ermöglichen, die gemeinsamen Ziele zu erreichen. Zwei Aspekte sind dabei von zentraler Bedeutung: Flexibilität und humane Arbeitsbedingungen. Das erklärte Ziel von XP ist die Fähigkeit eines Teams, flexibel auf Änderungen reagieren zu können. Doch nicht um jeden Preis. Im Gegenteil: Es geht um die Etablierung nachhaltiger Vorgehensweisen, die dauerhaft aufrechterhalten werden können. Vorgehensweisen, die ihre Stärken auch daraus ziehen, dass sie den Menschen als soziales Wesen und Individuum anerkennen.

Aus dieser Haltung heraus ergibt sich eine holistische Herangehensweise, die sowohl technische Details als auch soziale Aspekte explizit zu verknüpfen sucht. Alle wesentlichen Merkmale, die wir heute mit Agilität verbinden (kurze Iterationen, kontinuierliche Integration, offene Kommunikation als Grundlage, beständige Einbeziehung des Kunden etc.), hat XP schon vor über 20 Jahren auf den Weg gebracht. Provokant formuliert könnte man sagen, dass die bekannten Konzepte wie der „Scrum-Prozess" und das Agile Manifest, die nach XP kamen, lediglich unterschiedliche Aspekte von XP weiter verfeinert haben.

Wie bereits eingangs erwähnt: Was diese Methoden vereint, ist die Tatsache, dass sie in der Regel in der Praxis gut funktionieren. Und vielleicht genauso wichtig: Dass die Zufriedenheit der Teilnehmer an agilen Prozessen meistens als gut, gar sehr gut, bewertet wird. Aber wie ebenfalls bereits erwähnt: XP oder Scrum funktionieren nicht in jedem Kontext „einfach gut". Und das ist auch kein Anspruch dieser Methoden und Konzepte. Der Scrum-Prozess beschäftigt sich primär mit der Organisation eines Scrum-Teams. Bereits die Frage, ob Scrum darüber hinaus auch ein geeignetes Framework für Entscheider auf Unternehmensebene darstellt, wird heftig diskutiert und die Antworten fallen unterschiedlich aus. Wer Hilfestellungen zur Organisation von Teams mit Tausenden von Entwickler sucht, wird hier jedenfalls keine befriedigenden Antworten finden.

## Vom Team … zum Konzern

Im Weiteren soll es genau darum gehen: Wie kann man die Vorteile agiler Entwicklungsmodelle auf Teams mit 200 oder 2.000 Personen übertragen? Ich will an dieser Stelle stellvertretend zwei der bekanntesten Methoden vorstellen, beginnen werden wir mit dem „Scaled Agile Framework", kurz SAFe®.

SAFe entwickelte sich aus dem Buch „Agile Software Requirements", das Dean Leffingwell 2011 veröffentlich hat. Das erklärte Ziel: Die bekannten agilen Praktiken mit dem Konzept des „Lean Management" zu verbinden und das Management komplexer Portfolios von Produkten möglich zu machen. Dreh- und Angelpunkt bei SAFe ist das „SAFe Big Picture" *[7] (siehe Abbildung 1)*.

Im „Big Picture" finden sich alle Fragestellungen, für die SAFe Hilfestellungen anbietet. Zu jedem Element in dieser Grafik stellt SAFe ausführliche Artikel bereit, die das spezifische Thema detailliert be-

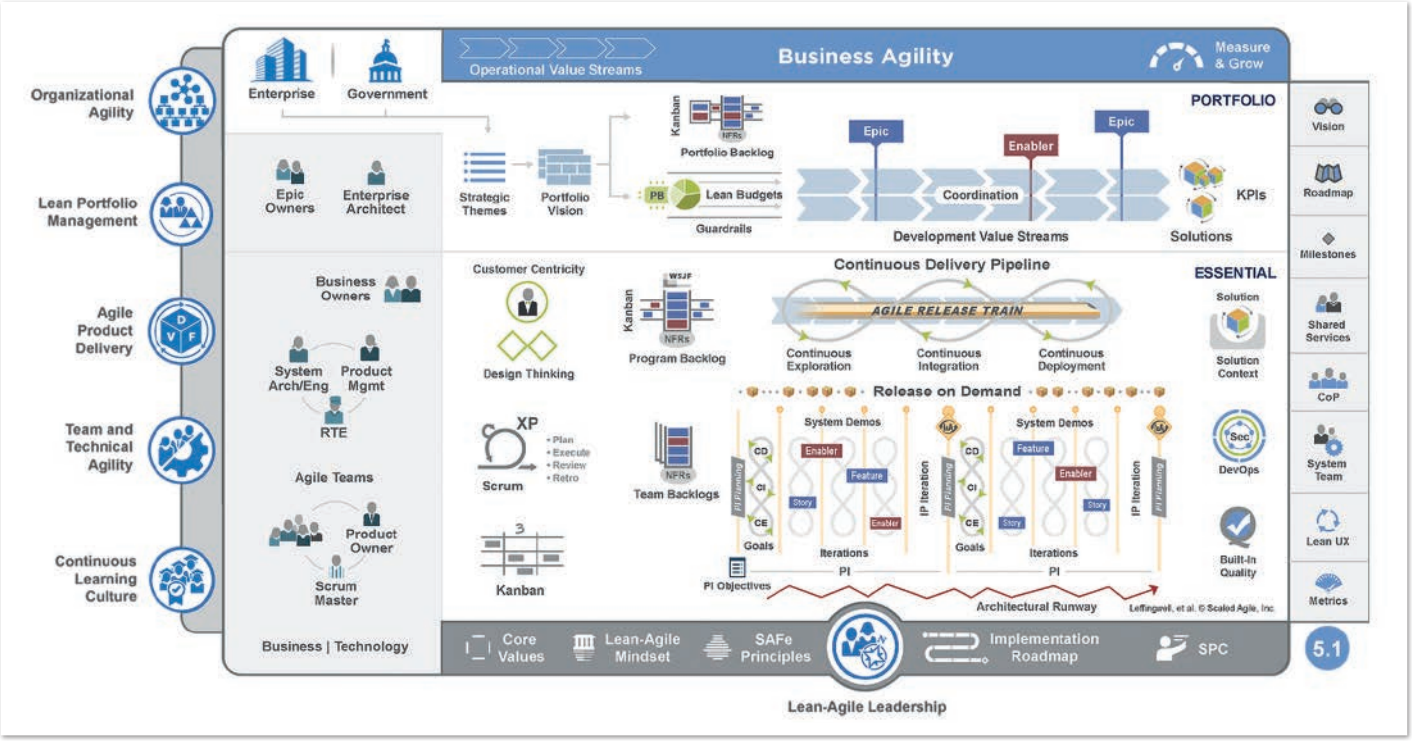



*Abbildung 1: SAFe for Lean Enterprises 5.1, © scaledagileframework.com*

sprechen und innerhalb des Frameworks einordnen. SAFe legt großen Wert darauf, seine Nutzer umfassend zu informieren und ihnen eine vollständige „Landkarte" bereitzustellen.

Um nur ein Beispiel zu nennen: SAFe definiert als zentrale Komponente den „Agile Release Train" (ART). Dabei handelt es sich um eine langlebige Gruppe agiler Teams, die zusammen mit weiteren Akteuren inkrementell „Solutions" innerhalb eines „Value Stream" entwickelt und ausliefert. Der ART steht in Beziehung zu verschiedenen agilen Organisationsformen (XP, Scrum, Kanban etc.). Die Verwaltung der Arbeitspakete erfolgt wenig überraschend durch „Team Backlogs", die auf höherer Ebene durch zentrale „Program Backlogs" koordiniert werden.

Das „Big Picture" definiert auf diese Weise alle relevanten Elemente, die in einer großen Organisation betrachtet werden sollten. Diese Wegmarken werden ausführlich beschrieben und aus unterschiedlichen Richtungen betrachtet. Insgesamt ist SAFe daher besonders für diejenigen geeignet, die sich am Anfang der Reise befinden. Wer sich eher in der Welt des Wasserfallmodells und ausführlich beschriebener Prozesse verortet, für den bietet SAFe konkrete Hilfestellungen mit hohem Detailgrad.

Für jene, die schon länger „auf der Reise" sind und nach neuen Wegen suchen, um eine Organisation, die bereits in vielen Teilen „agil" ist, auf die nächste Stufe der Evolution zu heben, lohnt sich der Blick auf andere Frameworks, zum Beispiel „LargE Scale Scrum", kurz LeSS, das seit 2013 von Bas Vodde entwickelt wird.

Auch für LeSS gibt es eine Übersichtsgrafik, die alle Themengebiete und Grundbestandteile der Methode übersichtlich darstellt. Viel spannender finde ich persönlich aber das „LeSS Complete Picture" *[8]* in *Abbildung 2*.

Was dabei sofort auffällt: „Minimalismus" und „Prioritäten" – die Essenz von LeSS sind nicht Dutzende verschiedener Themenblöcke, sondern die vier Bausteine „Prinzipien", „Frameworks", „Anleitungen" und „Experimente". Und offensichtlich: Den meisten Raum nimmt die Kategorie „Experimente" ein!

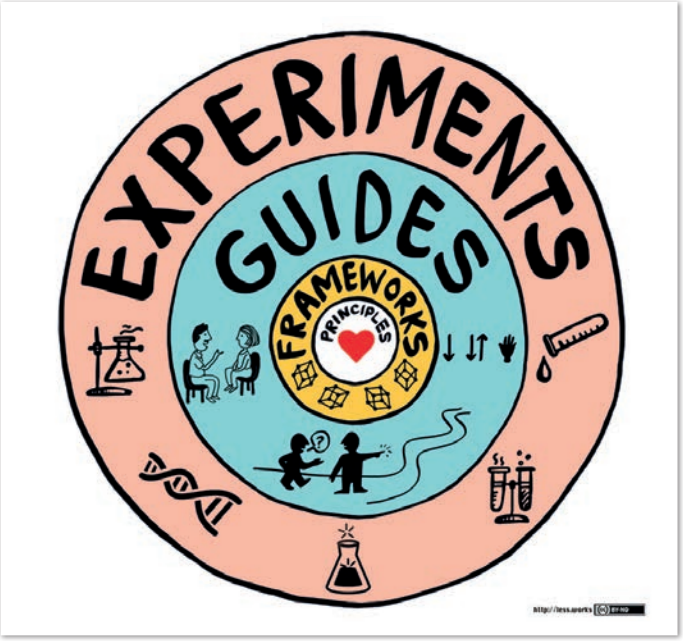



*Abbildung 2: The LeSS complete picture, © https://less.works*

Ziel von LeSS ist es nämlich nicht, die agile Welt komplett neu zu erfinden. Es geht vielmehr darum, die bekannten und etablierten Prinzipien und Elemente des „Scrum"-Teamprozesses im Kontext einer größeren Organisation zu transportieren und zu leben. In diesem Sinne: eine (Rück-)Besinnung darauf, wie Agilität in kleineren Teams erfolgreich gestaltet wird. Die nächsten beiden Kreise machen klar, dass auch LeSS den Nutzern vielfältige Hilfestellungen an die Hand gibt, um jene Kernideen von Scrum in große Teams zu tragen. Am wichtigsten jedoch ist die Ansage: „Du musst selbst herausfinden, was für Dich das Richtige ist." Anders formuliert: Das eigentliche Ziel von LeSS besteht darin, die Nutzer in die Lage zu versetzen, sinnvolle Fragestellungen und Thesen zu entwickeln, um diese dann durch Experimente zu beantworten beziehungsweise zu validieren. LeSS will ihnen nicht sagen, wie sie ihre Organisation umbauen oder ihre Prozesse gestalten sollten. Ziel ist es vielmehr, sie in die Lage zu versetzen, diejenigen individuellen Fragen zu finden, deren Beantwortung genau ihre Situation nachhaltig verbessern wird.

Neben SAFe und LeSS gibt es eine ganze Reihe weiterer Methoden: Disciplined Agile Delivery (DAD), Nexus, Enterprise Scrum, Scrum @ Scale, um nur einige zu nennen. Dabei ist sicherlich zutreffend, dass alle guten Modelle versuchen, den Anwendern nahe zu bringen: „Grau ist alle Theorie."

Daher möchte ich mich jetzt von eher theoretischen Betrachtungen über den Raum der Möglichkeiten verabschieden und konkret darlegen, wie Skalierung von Agilität in meinem Arbeitsumfeld aussieht.

## Exkurs: IBM System Z

Mein Arbeitgeber ist das IBM-Entwicklungslabor in Böblingen, und ich gehöre zum Team „System Z Firmware Development". Vereinfacht kann man sagen: Wir entwickeln jegliche Software, die ein IBM-Z-Mainframe-Kunde zusammen mit der Hardware erhält. Das ist der wesentliche Unterschied im Vergleich zu Betriebssystemen wie z/VM oder z/OS, die getrennt vertrieben werden. Meine Definition des Begriffs Firmware ist deswegen so vage, weil wir ein breites Spektrum an Komponenten betreuen. Das beginnt bei Kern-Funktionen wie dem „BIOS" des Mainframes (geschrieben in C++, PL8, PLX, gerne auch: IBM Z Assembler), geht über umfangreiche Steuer- und Regelsoftware zur Kontrolle der physikalischen Hardware (C, C++) und endet bei komplexer Software zur Verwaltung der logischen Partitionen/virtuellen Maschinen, in denen die Kunden ihre Betriebssysteminstanzen betreiben (Java-Backend und JavaScript-Frontend).

Passend zu diesem hohen Grad an „technischer Diversität" gestaltet sich die organisatorische Seite: Unser Team besteht aus etwa 500 Mitarbeitern und ist global verteilt über Standorte in Deutschland, in den USA und in Indien. Ein sehr heterogenes Umfeld, mit jungen Uni-Abgängern auf der einen Seite und erfahrenen Kollegen, die manchmal seit 30 Jahren die gleiche Komponente betreuen, am anderen Ende des Spektrums.

Die große Herausforderung, die sich für unser Team stellt, ergibt sich aus dem „Ende von Moore's Law". Um das zu erklären, muss ich etwas weiter ausholen.

IBM entwickelt die IBM Z Mainframes kontinuierlich weiter. In der Regel wird alle zwei bis drei Jahre die nächste Generation Mainframes aufgelegt. Das bedeutet konkret, dass beispielsweise die Prozessor-Chips komplett neu entwickelt werden. Bis vor einigen Jahren ergab sich daraus automatisch eine signifikante Steigerung der Leistung des Gesamtsystems. Für viele Kunden war das ein wesentlicher Grund für den Kauf eines neuen Mainframes. In diesem Modell ist die primäre Aufgabe von Firmware, die Geschwindigkeitsvorteile und neuen Funktionen der Hardware effizient nutzbar zu machen. Dementsprechend diktiert die Verfügbarkeit der neuen Hardware die Zeitpläne für die Entwicklung der Firmware. Insgesamt geht es hier um Tausende von Entwicklern, die über Jahre darauf hinarbeiten, zu einem bestimmten Termin die nächste Systemgeneration auf den Markt zu bringen. Diese Vorgehensweise hat für mehr als 20 Jahre sowohl für die IBM als auch für deren Kunden gut funktioniert. Aber John Markoff brachte es bereits 2015 präzise auf den Punkt: „Smaller, Faster, Cheaper, Over" *[9]*. Moore's Law gilt nicht mehr, ein neuer Prozessor in der neuesten Technologie ist nicht automatisch (sehr viel) schneller als das Vorgängermodell.

Natürlich legt die IBM Wert darauf, dass jede „nächste Generation" mehr Leistung erbringt als das aktuelle Modell. Nur ist eben der Zuwachs nicht mehr in den Größenordnungen machbar, die noch vor fünf oder zehn Jahren üblich waren.

Daraus folgt eine hohe Notwendigkeit, neue Kaufanreize zu schaffen. Konkret bedeutet das: Der IBM Z Mainframe muss auf allen Ebenen attraktiver werden. Zum Beispiel dadurch, dass die Firmware nicht mehr nur die grundlegenden Funktionen der unterliegenden Hardware verfügbar macht, sondern darüber hinaus komplexe Funktionen bereitstellt, die das Management des Mainframes in allen Bereichen für die Kunden massiv vereinfacht und verbessert.

Dabei ist es von großer Bedeutung, Lösungen nicht im Elfenbeinturm zu entwerfen, die an den vielfältigen Bedürfnissen unserer Kunden vorbeigehen. Um „das Richtige" zu liefern, setzt IBM auf die „IBM Enterprise Design Thinking"-Methode *[10]*. Deren Ziel ist es, die Benutzung jedweden Produkts auf die Bedürfnisse der Endanwender zu optimieren.

Zusammengefasst ergibt sich daher für unser Team die Notwendigkeit, einerseits vorausschauend die wichtigen technischen Trends frühzeitig zu erkennen und mit hoher Qualität in unser Produkt einfließen zu lassen. Das geschieht in intensiver Zusammenarbeit mit Kunden, jedoch unter Beibehaltung der erwähnten langen Release-Zyklen.

Allen Beteiligten war klar, dass die existierenden Prozesse und Strukturen, die sich sehr stark am Wasserfallmodell orientieren und seit 15 bis 20 Jahren immer weiter verfeinert wurden, keine angemessene Grundlage für die notwendige Neuausrichtung des Teams bilden.

## Agilität für den Mainframe

Unsere „Antwort" basiert auf drei Säulen:

1. Seit über zehn Jahren werden regelmäßig Schulungen zu agilen Entwicklungsmethoden durchgeführt. Ziel ist es, die breite Masse der Entwickler mit neuen Konzepten und Ansätzen vertraut zu machen. Dabei wurde und wird großer Wert daraufgelegt, die verschiedenen existierenden Rollen (beispielsweise Entwicklung, Teamführung, Projektmanagement) gleichermaßen anzusprechen und gemeinsam neue Wege zu entwickeln.
2. Die bereits erwähnte Methode „IBM Enterprise Design Thinking" mit dem Ziel, neue Ideen möglichst schnell mit Kunden zu prüfen und weiterzuentwickeln. Besonders wichtig ist uns dabei, die Aktivitäten der Design-Teams möglichst eng mit der eigentlichen Produktentwicklung zu verbinden.
3. Einer dediziert neuen Organisationsform, basierend auf dem „Spotify-Modell" *[11]*.

Oberflächlich betrachtet ist das Spotify-Modell eine Methode, um die Struktur einer größeren Organisation so zu gestalten, dass ein hohes Maß an Autonomie entstehen kann, während gleichzeitig darauf geachtet wird, dass die einzelnen Teams auf die gleichen Ziele hin ausgerichtet sind. Auf der untersten Stufe stehen dabei die sogenannten „Squads". Das sind agile Teams, die idealerweise die vollständige Verantwortung für eine autarke Komponente haben, sich also holistisch um Design, Entwicklung und Test kümmern. Alle Squads, die am gleichen Produkt arbeiten, kommen im „Tribe" zusammen. Weiterhin existieren innerhalb eines Tribe die sogenannten „Chapter". Deren Sinn besteht darin, den Austausch über bestimmte Interessengebiete innerhalb des Tribe zu voranzubringen. Ein gängiges Beispiel sind Chapter für „Frontend"- beziehungsweise. „Backend"-Entwicklung. Daneben gibt es noch „Guilds", das sind Interessenverbände, die über mehrere Tribes hinweg agieren. Das alles fließt in unserem Team im sogenannten Alliance Board zusammen: Hier kommen regelmäßig Management, Tribe Leader, Projektmanagement und technische Führungskräfte zusammen, damit die gemeinsamen Ziele nicht aus dem Fokus geraten.

Entscheidend ist aber an dieser Stelle die Haltung aller Beteiligten. Das primäre Ziel ist es, neue Freiheitsgrade zu schaffen. „Squad" ist nicht nur eine andere Bezeichnung für „Team". Es geht vielmehr darum, den Mitgliedern der Squads die Möglichkeit zu geben, die eigenen Interessen und Stärken konstruktiv einzubringen. Und das gelingt weder über Nacht noch mit wenigen, großen Würfen. Vielmehr sind Kontinuität und Beständigkeit gefragt. Gleichzeitig bedarf es hier der Balance: Unsere „Basisaufgaben" müssen weiterhin mit der Qualität erledigt werden, die unsere Kunden von uns erwarten. Andererseits soll jedes Mitglied unseres Teams die Möglichkeit haben, neue Ansätze zu erproben, zu verfeinern, zu verwerfen.

Es wird Sie vermutlich nicht überraschen: In der Realität landet man irgendwo in der Mitte. Es gibt Gruppen, die sich (noch) in der alten Welt wohler fühlen und sich nur mit kleineren Schritten auf diese neue, weniger formal definierte Zukunft einlassen. Es gibt allerdings auch Squads, die das neue Modell jeden Tag neu leben und dadurch in der Lage sind, die potenziellen Konflikte zwischen Innovationsfreude, Qualität und Termindruck weitestgehend aufzulösen. Außerdem zeigt sich: Rosinenpicken ist erlaubt und sinnvoll – unsere Projektmanager orientieren sich beispielsweise an verschiedenen Elementen aus dem SAFe-Kosmos, was die Neugestaltung unseres Portfolio-Backlog-Managements angeht.

Die Möglichkeiten zur Einflussnahme durch das höhere Management sind eher begrenzt: Es geht in erster Linie darum, positive

Anreize zu schaffen, da ein erfolgreicher Kulturwandel hin zum „autonomen Entwickler" nicht von oben verordnet werden kann. Gelebte Führungsverantwortung bedeutet hier vor allem, aktiv an der Lösungsfindung mitzuwirken, wenn Konflikte mit externen Gruppen (innerhalb der IBM) offenbar werden, die ihre Ursache in unterschiedlichen Grundhaltungen zu Agilität haben.

## Fazit

Zusammenfassend möchte ich nochmals betonen, worin meines Erachtens der wichtigste Aspekt bei der Organisation von Softwareentwicklung in großen Teams liegt: Es geht nicht darum, in einem YouTube-Video, einem Artikel in der Java aktuell oder einem guten Buch die allgemein gültigen Regeln und Antworten zu suchen, die alle Probleme auf Jahre hinaus lösen. Das Ziel ist stattdessen, kontinuierlich die richtigen Fragen zu identifizieren! Herauszufinden, wie die eigene Organisation tickt und welche ihrer Stärken dazu geeignet sind, die aktuellen Herausforderungen zu meistern. Und sich immer wieder klarzumachen, dass auch gute, funktionierende Lösungen nicht für die Ewigkeit sind, sondern regelmäßig auf den Prüfstand gehören.

Schließen möchte daher ich dementsprechend mit einem Gedankenexperiment, das dem Konzept des „Intentional Living" *[12]* entlehnt ist:

*Tun Sie die Dinge, weil Sie es können? Oder agieren Sie, weil Sie sich bewusst entschieden haben, genau so zu handeln?*

Soll heißen: Sowohl in der Software-Entwicklung als auch im realen Leben kann man sich treiben lassen, um dann dort zu landen, wohin die Umstände einen führen. Man kann sich aber auch überlegen „Wo will ich eigentlich hin?", um sich bewusst für den Weg zu entscheiden, der in Richtung dieses Ziels führt. Welchen Weg gehen Sie heute, und ist das wirklich der Weg, den Sie gehen wollen?

## Quellen


[1] *https://agilemanifesto.org/iso/de/manifesto.html*
[2] *http://homepages.cs.ncl.ac.uk/brian.randell/NATO/nato1969.PDF*
[3] *http://www-scf.usc.edu/~csci201/lectures/Lecture11/royce1970.pdf*
[4] *https://www.cio.bund.de/Web/DE/Architekturen-und-Standards/V-Modell-XT/vmodell_xt_node.html*
[5] *https://space.stackexchange.com/questions/9260/how-often-if-ever-was-software-updated-in-the-shuttle-orbiter/9271#9271*
[6] Hoffentlich: NASA, Boeing, Airbus, jeder AKW-Betreiber, …
[7] *https://www.scaledagileframework.com/*
[8] *https://less.works/*
[9] *https://www.nytimes.com/2015/09/27/technology/smaller-faster-cheaper-over-the-future-of-computer-chips.html*
[10] *https://www.ibm.com/design/thinking/*
[11] *https://engineering.atspotify.com/2014/03/27/spotify-engineering-culture-part-1/*
[12] *https://en.wikipedia.org/wiki/Intentional_living*



**Edwin Günthner**
IBM Deutschland Research & Development GmbH
*edwin.guenthner@de.ibm.com*


Ich habe an der Universität Karlsruhe studiert und dort 1999 meinen Abschluss als Diplom-Informatiker gemacht. Danach war ich zunächst zwei Jahre als Java-Entwickler beim Compart Systemhaus in Böblingen beschäftigt, bevor ich 2001 zur IBM Deutschland Research & Development GmbH gewechselt habe. Dort bin ich seither in wechselnden Rollen in den Bereichen zSystems-Hardware und Firmware-Entwicklung im Labor Böblingen tätig. In den letzten Jahren habe ich mich intensiv mit der Frage beschäftigt, wie im Kontext eines komplexen Produkts, der IBM z Systems (aka „Mainframe"), und eines großen, global verteilt agierenden Teams von Entwicklern agile Entwicklungsprozesse gestaltet werden können. Jetzt ist es mir ein Anliegen, diese Erfahrungen an alle Interessierten weiterzugeben.


## Mitglieder des iJUG

01 Android User Group Düsseldorf
02 BED-Con e.V.
03 Clojure User Group Düsseldorf
04 DOAG e.V.
05 EuregJUG Maas-Rhine
06 JUG Augsburg
07 JUG Berlin-Brandenburg
08 JUG Bremen
09 JUG Bielefeld
10 JUG Bonn
11 JUG Darmstadt
12 JUG Deutschland e.V.
13 JUG Dortmund
14 JUG Düsseldorf rheinjug
15 JUG Erlangen-Nürnberg
16 JUG Freiburg
17 JUG Goldstadt
18 JUG Görlitz
19 JUG Hannover
20 JUG Hessen
21 JUG HH
22 JUG Ingolstadt e.V.
23 JUG Kaiserslautern
24 JUG Karlsruhe
25 JUG Köln
26 Kotlin User Group Düsseldorf
27 JUG Mainz
28 JUG Mannheim
29 JUG München
30 JUG Münster
31 JUG Oberland
32 JUG Ostfalen
33 JUG Paderborn
34 JUG Passau e.V.
35 JUG Saxony
36 JUG Stuttgart e.V.
37 JUG Switzerland
38 JSUG
39 Lightweight JUG München
40 SOUG e.V.
41 SUG Deutschland e.V.
42 JUG Thüringen
43 JUG Saarland

iJUG Verbund

www.ijug.eu

# Impressum

Java aktuell wird vom Interessenverbund der Java User Groups e.V. (iJUG) (Tempelhofer Weg 64, 12347 Berlin, *www.ijug.eu*) herausgegeben. Es ist das User-Magazin rund um die Programmiersprache Java im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Java-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

Java aktuell wird verlegt von der DOAG Dienstleistungen GmbH, Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Björn Bröhl. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führen einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
ViSdP: Fried Saacke
Redaktionsleitung: Lisa Damerow
Kontakt: redaktion@ijug.eu

Redaktionsbeirat:
Andreas Badelt, Melanie Feldmann, Marcus Fihlon, Markus Karg, Manuel Mauky, Bernd Müller, Benjamin Nothdurft, Daniel van Ross, André Sept

Titel, Gestaltung und Satz:
Alexander Kermas,
DOAG Dienstleistungen GmbH

Bildnachweis:
Titel: Bild © storyset
*https://freepik.com*
S. 11: Bild © JulsIst
*https://stock.adobe.com*
S. 16+17: Bild © mast3r
*https://stock.adobe.com*
S. 22: Bild © metamorworks
*https://stock.adobe.com*
S. 28+29: Bild © VectorMine
*https://stock.adobe.com*
S. 33: Bild © asrulaqroni
*https://freepik.com*
S. 38: Bild © royyimzy
*https://stock.adobe.com*
S. 47: Bild © alphaspirit
*https://stock.adobe.com*
S. 52+53: Bild © leremy
*https://123rf.com*

Anzeigen:
DOAG Dienstleistungen GmbH
Kontakt: sponsoring@doag.org

Mediadaten und Preise:
*www.doag.org/go/mediadaten*

Druck:
WIRmachenDRUCK GmbH
*www.wir-machen-druck.de*



# Inserentenverzeichnis


DOAG 2022
Konferenz + Ausstellung
In Nürnberg
20.-23. SEPT.
Die Oracle-
ANWENDERKONFERENZ
anwenderkonferenz.doag.org
Eventpartner:
AOUG
AUSTRIAN ORACLE USER GROUP
SOUG
swiss oracle user group
iJUG
Verbund