RISC Software GmbH
www.risc-software.at

# INSIGHTS²

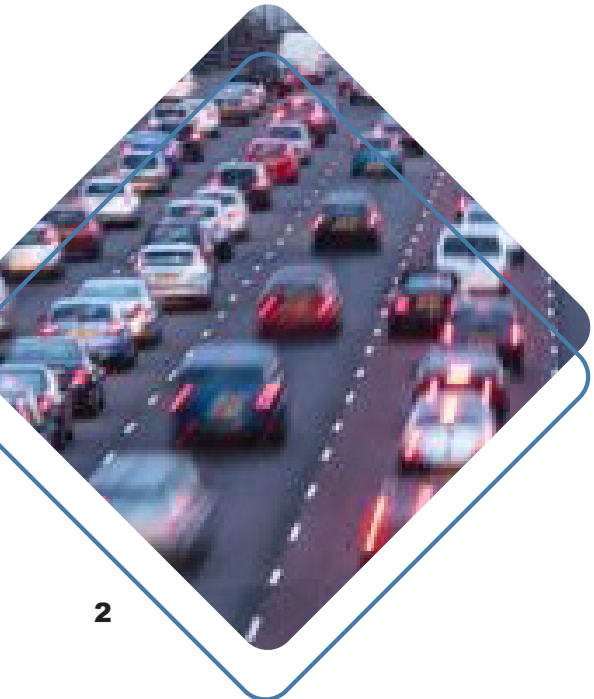
Das Fachmagazin der RISC Software GmbH
zu aktuellen Forschungs- und Entwicklungsthemen.

Agile Softwareentwicklung

Software-Reengineering

Data Science und Prescriptive Analytics

Intelligente Transportsysteme



# INHALT

## Liebe Leserin, lieber Leser,

es freut mich, dass wir Ihnen mittlerweile die zweite Ausgabe unserer gesammelten Fachbeiträge präsentieren können. Wir möchten Ihnen damit wieder allgemeine Themeneinstiege und tiefergehende Einblicke in einige unserer Arbeitsgebiete und Expertisen geben. In diesem Magazin finden Sie folgende Themenblöcke:

- **Agile Softwareentwicklung**: Agile Vorgehensmodelle setzen vor allem auf der organisatorischen Seite an. Agile Workflows und Test-Driven-Development richtig eingesetzt, bringen rasches Feedback und schnelle Reaktionszyklen.
- **Data Science und Prescriptive Analytics:** Die Verschmelzung der beiden Welten „klassische Optimierung" und „lernende Systeme" zeigt, wie Modelle und Daten zu Lösungen kombiniert und daraus neue Erkenntnisse gewonnen werden.
- **Software-Reengineering** – weg von den Legacy-Systemen!
- **Intelligente Transportsysteme** – wir brauchen einen Mobilitätsrevolution ... eine Initiative unserer (jungen) Mitarbeiter*innen für eine nachhaltige und selbstbestimmende Mobilität.

Vieles, das Sie hier lesen, ist langjährig aufgebautes und in Forschungsprojekten erarbeitetes Know-how gepaart mit den modernsten wissenschaftlichen Methoden, die erfolgreich in Kund*innenprojekten umgesetzt wurden. Denn was uns auszeichnet, ist die an uns herangetragenen Problemstellungen gemeinsam mit unseren Kund*innen so zu lösen, dass über die Digitalisierung hinaus auch langfristig ein entscheidender Wettbewerbsvorteil und oft eine langjährige gute Partnerschaft entsteht. Der weite Begriff der „digitalen Transformation" ist in aller Munde, muss allerdings von jeder Organisation individuell durchgeführt werden. Auf diesem Weg können wir Sie mit unserer Expertise und unserem Know-how auf vielfältige Weise unterstützen, aber gehen müssen Sie den Weg im Endeffekt selbst.

Ein großes Thema, das derzeit viele Industrieunternehmen beschäftigt, ist die Re-Globalisierung: Damit ist gemeint, dass die Lieferketten sich in Richtung Stabilität und Redundanz verändern und Produktionen wieder zurück nach Europa geholt werden. In Kombination mit dem allgegenwärtigen Fachkräftemangel ist dies eine enorme Herausforderung, welche nur mit verstärkter Automatisierung (=Digitalisierung) bewältigt werden kann. Ganz egal, wo Sie stehen - ob Sie komplexe Prozesse digitalisieren, alte Bestandssoftware am laufenden Stand der Technik bringen, Ihre Fertigungsdaten für Prognosen nutzen wollen oder irgendwo anders - unser Expert*innenteam unterstützt Sie bei der Umsetzung Ihrer F&E-Vorhaben mit vielfältigem Fachwissen!

Viel Spaß beim Lesen!

Wolfgang Freiseisen
CEO Software GmbH

# Methoden und Werkzeuge für die Datenaufbereitung im Big Data Bereich

– DI Paul Heinzlreiter
Senior Data Engineer in der Unit Logistics Informatics

In den letzten Jahren hat die Rolle von Big Data in zahlreichen Wirtschaftsbereichen wie beispielsweise der produzierenden Industrie, Logistik oder Handel stark an Bedeutung gewonnen. Unter Einsatz verschiedenster Sensor-Systeme werden große Datenmengen gesammelt, die in weiterer Folge zur Optimierung von Maschinen oder Geschäftsprozessen herangezogen werden können. Hierbei kommen oft Methoden aus den Bereichen Künstliche Intelligenz, maschinelles Lernen oder Statistik zum Einsatz.

All diese Methoden benötigen als Grundlage allerdings eine größere Menge an qualitativ hochwertigen und validen Daten. In diesem Kontext kommt Data Engineering zum Einsatz, um die Rohdaten zu sammeln, zu bereinigen und zu einer integrierten Datenbasis zusammenzuführen. Während in einem früheren Artikel (im Magazin INSIGHT #1) die generelle Rolle und Ziele von Data Engineering beleuchtet wurden, soll hier der Fokus auf Methoden und bewährte Werkzeuge gelegt werden sowie ein beispielhafter Einblick in die algorithmische Umsetzung von Data Engineering Aufgaben gegeben werden.

## Datenstrom- und Batch-Verarbeitung

Wenn beispielsweise industrielle Sensordaten über die Zeit gesammelt werden, fallen pro Zeiteinheit (z.B. alle paar Sekunden) keine großen Datenmengen an, über Monate und Jahre steigen die gespeicherten Datenmengen aber oft in den Terabyte-Bereich. Wenn Daten in dieser Größenordnung verarbeitet werden sollen, kann dies im Wesentlichen durch zwei verschiedene Paradigmen erfolgen, hier beschrieben für die Konvertierung des Datentyps einer Tabellenspalte:

- Batch-Verarbeitung:

Hierbei werden alle Zeilen in einer Tabelle parallel verarbeitet, um eine Spalte zu konvertieren.

- Datenstromverarbeitung (Data Streaming):

Hierbei werden die Zeilen der Tabelle sequenziell eingelesen und die Spaltenkonvertierung pro Zeile durchgeführt.

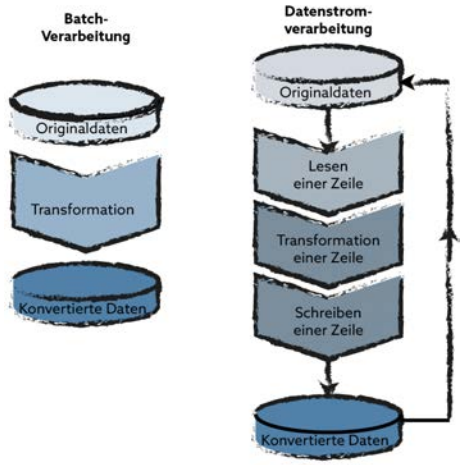



Abbildung 1: Vergleich Batch- und Datenstromverarbeitung

Der wesentliche Unterschied zwischen den beiden Datenverarbeitungsansätzen ist, dass beim Data Streaming die notwendigen Datentransformationen – wie beispielsweise die Konvertierung von Datenfeldern auf andere Datentypen – direkt auf dem aktuell gelieferten Datensatz durchgeführt werden, während bei der Batch-Verarbeitung die Daten zuerst gesammelt werden, und in der Folge die Datentransformationen auf der Gesamtheit der Daten durchgeführt werden. Welcher Ansatz gewählt wird, ist von den Anforderungen an die Datentransformation abhängig:

- Wenn die Transformation lokal auf den aktuell abgefragten oder empfangenen Daten durchgeführt werden kann, ist oft der Einsatz des Data Streaming Ansatzes vorzuziehen, da er meist eine simple und lokale Operation darstellt, die aufgrund der geringeren Eingabedaten auch schneller abgearbeitet werden kann. Ein typischer Einsatzbereich von Data Streaming ist die direkte Konvertierung von über die Zeit verteilt eintreffenden Sensordaten, da diese dann einzeln konvertiert und gespeichert werden können.

- Wenn allerdings die Datentransformation Eingabedaten aus dem gesamten bereits gespeicherten Daten benötigt oder bereits alle Daten vorliegen, eignet sich ein Batch-Ansatz besser. Hierbei ist auch oft eine parallele Verarbeitung der Daten einfacher umzusetzen, da diese von Frameworks wie Apache Hadoop (durch den Map-Reduce Ansatz) oder Apache Spark direkt unterstützt wird.

Generell sollten die erhaltenen Daten einmal im Rohformat gespeichert werden, um keine Daten zu verlieren, die als Grundlage für zukünftige Analysen noch benötigt werden könnten. Die Weiter-

verarbeitung so gespeicherter Daten kann dann durch Batch-Verarbeitung oder Data Streaming erfolgen. Im zweiteren Fall wird aus den gespeicherten Daten durch kontinuierliches Lesen wieder ein Datenstrom erzeugt. Umgekehrt kann ein Datenstrom kontinuierlich gespeichert werden und somit als Ausgangsbasis für Batch-Verarbeitung dienen.

### Datenstromverarbeitung: Apache NiFi

NiFi stellt ein Werkzeug zur Datenstromverarbeitung dar, welches es ermöglicht, in einer grafischen, webbasierten Benutzerschnittstelle Datentransformationen zu einer durchgängigen Datenpipeline zu verbinden, durch die die Quelldaten fließen und schrittweise transformiert werden. Die Stärken von Apache NiFi liegen in der breiten Palette an bereits verfügbaren Modulen, die beispielsweise das Einlesen und Speichern von zahlreichen Datenformaten ermöglichen. Durch den Open-Source-Charakter von NiFi und die objektorientierte Struktur seiner Module ist es leicht möglich, selbst Module zu entwickeln und diese in Datenpipelines zu integrieren. Weiters werden durch NiFi auch Themen wie die automatisierte Behandlung von verschiedenen Verarbeitungsgeschwindigkeiten der Module geregelt.

### Batch-Verarbeitung: Apache Hadoop

Hadoop ist ein Software-Framework, das auf dem Grundprinzip der parallelen Datenverarbeitung in einer Cluster-Umgebung beruht. Innerhalb der verteilten Verarbeitung übernimmt jeder Cluster-Rechner die Verarbeitung der dort lokal vorliegenden Daten, womit vor allem Kommunikationsaufwand während der Berechnungen eingespart werden kann. Hadoop unterscheidet hierbei zwischen Controller- und Responder-Diensten im Cluster, wobei die Responder-Dienste die Verarbeitung der lokal vorliegenden Daten übernehmen, während den Controller-Diensten die Koordination des Clusters obliegt. Teile der in Hadoop implementierten Algorithmen wurden von Google entwickelt und die Konzepte in Forschungspapers veröffentlicht, wie beispielsweise das Google File System, Map-Reduce und Google Bigtable. Bei Google werden diese Lösungen für den Betrieb der weltweiten Suchinfrastruktur eingesetzt, während das Hadoop-Projekt eine Open-Source-Implementierung dieser Konzepte darstellt.

Im Kern besteht ein Hadoop-System aus einem üblicherweise Linux-basierten Cluster, auf dem das Hadoop File System (HDFS) und YARN als Implementierung des Map-Reduce Algorithmus laufen. Ein Hadoop-Cluster mit den Diensten HDFS und YARN stellt eine solide technologische Basis für verschiedenste Big-Data-Dienste wie BigTable-Datenbanken wie HBase – siehe weiter unten – oder Graph-Datenbanken wie beispielsweise JanusGraph dar.

### Hadoop File System (HDFS)

HDFS ist eine Open-Source-Implementierung des Google Filesystems. Es besteht, genauso wie andere Hadoop-Teilsysteme, aus Controller- und Responder-Komponenten, im Fall von HDFS Namenodes (Controller) und Datanodes (Responder). Während ein Namenode speichert, wo am Cluster die Daten für einzelnen Dateien hinterlegt sind, übernehmen die Datanodes die Speicherung der Datenblöcke. Grundsätzlich ist HDFS für große Dateien optimiert, die Blockgröße für die Speicherung beträgt üblicherweise 128 Megabyte. Einerseits kann eine Datei aus vielen einzelnen Blöcken bestehen, andererseits werden die Datenblöcke aus Redundanz- und Performancegründen über mehrere Clusterknoten repliziert. Die Zugriffssemantik von HDFS unterscheidet sich von der üblichen Posix-Semantik, da an HDFS-Dateien nur Daten angehängt werden können, diese aber nicht editiert werden können. Um eine neue Version einer Datei zu erzeugen, muss diese ersetzt werden. Dies kann auch für große Dateien mit dem unten beschriebenen Map-Reduce Algorithmus sehr effektiv durchgeführt werden.

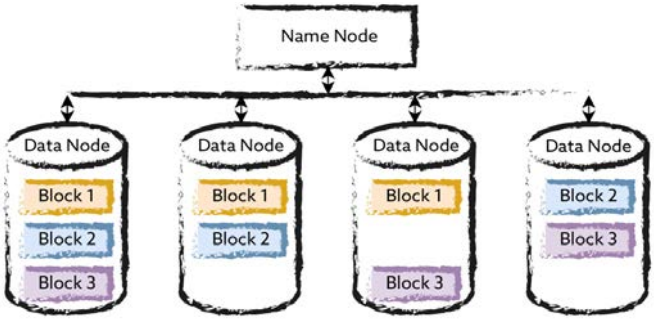



Abbildung 2: Blockreplikation im Hadoop-File-System (HDFS).

Im Kontext eines Hadoop-Systems können nun beispielsweise im CSV-Format abgelegte Textdateien mit Map-Reduce Jobs verarbeitet werden, wobei die Verteilung der Teil-Jobs über den Cluster basierend auf der Verteilung der HDFS-Dateiblöcke automatisch vom Hadoop-Framework übernommen wird. Neben reinen Textdateien können auch strukturierte Binärdaten wie zum Beispiel die Formate ORC, Parquet oder AVRO direkt von Hadoop verarbeitet werden. Für neue Formate können darüber hinaus spezifische Splitter-Klassen für Map-Reduce implementiert werden. Weiters ist es im Rahmen eines Map-Reduce-Algorithmus ohne Probleme möglich, beispielsweise nur eine Map-Stufe durchzuführen, um neue Spalten zu einer CSV-Datei hinzuzufügen.

### Map-Reduce-Framework (YARN)

Basierend auf der oben gezeigten Datenverteilung in HDFS kann nun durch den Dienst YARN ein datenparalleler Batch-Job ausgeführt werden, wobei jeder Responderknoten die lokal vorliegenden Datenblöcke bearbeitet. Konzeptionell folgt die Ausführung dabei dem Map-Reduce-Algorithmus. Ein klassisches Anwendungsbeispiel für den Map-Reduce-Algorithmus ist das Zählen von Wörtern in Textdokumenten. Hier emitiert der Map-Schritt pro Zeile eine Menge an Paaren der Form (Wort, Anzahl des Auftretens in der Zeile). Im Shuffle-Schritt werden diese Wertpaare nach den Worten zusammengefasst, da diese den Schlüssel repräsentieren. Im abschließenden Reduce-Schritt werden die Worthäufigkeiten pro Wort aufsummiert. Eine beispielhafte Ausführung könnte wie folgt ablaufen:

- Der Eingabetext wird in einzelne Textzeilen aufgeteilt. (Splitting)
- Der Map-Schritt, welcher für jede Zeile einzeln parallel ausgeführt wird, erzeugt für jedes Wort in der Zeile ein Paar aus dem Wort und der Zahl 1. (Mapping)
- Die Paare werden nach den Worten sortiert und zu einer Liste je Wort zusammengefasst. (Shuffling)
- Für jedes Wort wird durch Aufsummieren der Zahlen die Anzahl im Gesamttext bestimmt. (Reducing)

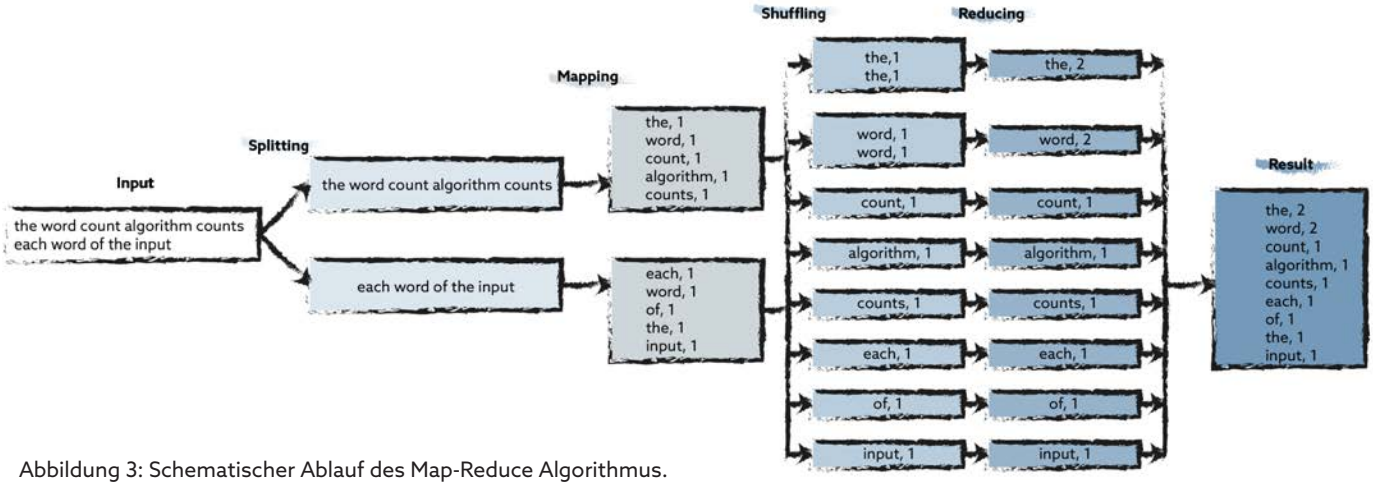



Abbildung 3: Schematischer Ablauf des Map-Reduce Algorithmus.

Während der Map-Schritt und der Reduce-Schritt jeweils ausprogrammiert werden müssen, wird der globale Shuffle-Schritt automatisch vom Map-Reduce-Framework übernommen. Die Implementierung des Map- und Reduce-Schritts erfordert in der Praxis beispielsweise das objektorientierte Überschreiben von jeweils einer Map- und einer Reduce-Methode, deren Schnittstellen bereits vorgegeben sind. Dies ermöglicht den Fokus auf die Transformation eines Wertepaares zu setzen, während sich das Framework in weiterer Folge um die skalierte Ausführung am Cluster kümmert.

## Batch- und Datenstromverarbeitung: Apache Spark

Spark stellt eine flexible Datenverarbeitungsschicht dar, die auf verschiedene Infrastrukturen wie beispielsweise Hadoop aufgesetzt und für verschiedene Aufgaben im Bereich von Data Engineering aber auch Data Science genutzt werden kann. Als allgemeines Datenverarbeitungsframework kann Spark Aufgaben der Datenvorverarbeitung genauso wie Machine-Learning-Aufgaben übernehmen.

Apache Spark kann beispielsweise auf einem bestehenden Hadoop-Cluster installiert werden und direkt auf die dort gespeicherten Daten zugreifen und diese parallel verarbeiten. Ein Ansatz dazu ist der oben genannte Map-Reduce-Algorithmus, wobei allerdings Spark noch andere flexible Methoden wie beispielsweise Datenfilterung anwenden kann. Spark legt Zwischenergebnisse als verteilte Datensets (resilient distributed datasets, RDDs) im Hauptspeicher ab, wodurch langsame wiederholte Festplattenzugriffe – wie häufig bei klassischen Datenbanken – vermieden werden können.

Wichtige Features von Spark umfassen unter anderem:

- Parallele Batchverarbeitung, zum Beispiel unter Einsatz des Map-Reduce-Algorithmus.
- Unterstützung von SQL-Abfragen auf beliebig (z.B. im HDFS) gespeicherte Daten. Hierzu muss nur interaktiv eine Tabelle angelegt werden, die das zu verwendende Datenschema definiert und auf die zugrundeliegenden Daten verweist.
- Basierend auf der sequenziellen Verarbeitung von mehreren RDDs kann Datenstromverarbeitung durchgeführt werden.

Genauso wie ein zugrundeliegender Hadoop-Cluster kann eine Spark-Installation durch eine einfache Hardware-Erweiterung für die Verarbeitung größerer Datenmengen fit gemacht werden.

## Die Wahl der passenden Werkzeuge für Big Data Engineering

Wie aus den folgenden gezeigten Anwendungsbeispielen (Datenüberführung aus einer CSV-Datei in eine SQL-Datenbank) ersichtlich ist, führen im Bereich Data Engineering oft verschiedene Wege zum selben Ziel. Welche Methoden zum Einsatz kommen sollten, hängt oft von den speziellen Anforderungen der Kunden sowie deren Systemumgebung ab:

- Wenn beispielsweise bereits ein Hadoop-Cluster im Einsatz oder geplant ist, kann dieser beim Entwurf einer Lösung bereits eingebunden werden.
- Public-Cloud-Angebote wie beispielsweise Amazon AWS bieten wiederum Alternativen zu den oben beschriebenen Open-Source-Lösungen an, welche vor allem den Betrieb der Lösung vereinfachen, aber auch zu Vendor-Lock-In führen können.
- Weitere Kriterien für eine Technologieentscheidung sind Anforderungen an Skalierbarkeit und die geplante Integration zusätzlicher Werkzeuge.
- Nicht zuletzt bieten Open-Source-Lösungen oft Kostenvorteile, da auch für hoch skalierende Lösungen keine Lizenzkosten anfallen.

## Anwendungsbeispiel: Aufbereitung von industriellen Sensor- und Logdaten

Im Rahmen des Forschungsprojekts VPA4.0 wurde in eine Datenpipeline zur Vorverabeitung der Produktions-Sensordaten eingerichtet. Diese stellt ein gutes Beispiel für die Verknüpfung von Streaming- und Batch-Verarbeitung dar. Apache NiFi wurde als Streaming-Lösung verwendet, um die Daten direkt vom Projektpartner verschlüsselt über das Internet zu übertragen, bevor sie lokal am Hadoop-Cluster gespeichert wurden. Die weitere Datenaufbereitung wurde dann mithilfe von Spark parallel auf dem Hadoop-Cluster durchgeführt und umfasste folgende Schritte:

- Entpacken der erhaltenen Datenarchive und Entfernung nicht benötigter Dateien
- Aufbereitung und Speicherung der Daten als CSV-Dateien im HDFS
- Anlegen virtueller Tabellen basierend auf CSV-Dateien ermöglicht die weitere Verarbeitung mit SQL
- Datenfilterung und Speicherung im optimierten Parquet-Format für interaktive SQL-Abfragen

## Anwendungsbeispiel: Bereinigen von Sensordaten und Ablegen in einer SQL-Datenbank

Dieses Beispiel umfasst Sensordaten, die an einer Wärmekraftmaschine gesammelt wurden. Man sieht in folgendem Beispiel in der

Spalte power_dynamo negative Werte, welche durch eine Messungenauigkeit entstanden sind. Zeilen mit solchen Werten sollen nun als fehlerhaft ausgefiltert und die bereinigten Daten in einer Datenbank gespeichert werden.

```
timestamp;temperature_heater;temperature_boiler;pressure_boiler;rpm;power_dynamo;power_heating;valve_aperture;water_level
2019-06-14T12:43:00;39.404514;267.222229;1292.380981;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:01;38.194447;268.361115;1292.380981;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:02;38.194447;267.222229;1292.326538;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:03;38.194447;267.222229;1292.380981;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:04;39.404514;268.361115;1305.005615;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:05;38.194447;267.222229;1292.380981;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:06;38.194447;267.222229;1317.630371;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:07;39.404514;267.222229;1305.005615;0.000000;-0.019452;446.973846;0.0;21.00
2019-06-14T12:43:08;38.194447;268.361115;1330.200562;0.000000;-0.019452;446.973846;0.0;21.00
...
```

**Umsetzung in Spark:**
In Spark können die Daten in einem ersten Schritt eingelesen, auf die richtigen Datentypen umgewandelt und in einem korrekt typisierten Dataframe abgelegt werden. Dieser stellt eine Spark-Standarddatenstruktur dar, in der Daten im Hauptspeicher gehalten werden. Dies kann mit einem Kommando in einer interaktiven pyspark-Shell umgesetzt werden, welche Python als Umsetzungssprache verwendet:

```
>>> df = spark.read.csv("file:///tmp/datacollection.csv", sep=';', header=True).selectExpr("cast(timestamp as timestamp)",
      "cast(temperature_heater as double)",
      "cast(temperature_boiler as double)",
      "cast(pressure_boiler as double)",
      "cast(rpm as double)",
      "cast(power_dynamo as double)",
      "cast(power_heating as double)",
      "cast(valve_aperture as double)").show()
+-------------------+-----------------+-----------------+---------------+---+------------+-------------+--------------+
|          timestamp|temperature_heater|temperature_boiler|pressure_boiler|rpm|power_dynamo|power_heating|valve_aperture|
+-------------------+-----------------+-----------------+---------------+---+------------+-------------+--------------+
|2019-06-14 12:43:00|        39.404514|        267.222229|    1292.380981|0.0|   -0.019452|   446.973846|           0.0|
|2019-06-14 12:43:01|        38.194447|        268.361115|    1292.380981|0.0|   -0.019452|   446.973846|           0.0|
|2019-06-14 12:43:02|        38.194447|        267.222229|    1292.326538|0.0|   -0.019452|   446.973846|           0.0|
```

Mit dem folgenden Kommando können die Daten direkt in der SQL-Tabelle EngineData abgelegt werden:

```
>>> df = spark.read.csv("file:///tmp/datacollection.csv", sep=';', header=True).selectExpr("cast(timestamp as timestamp)",
      "cast(temperature_heater as double)",
      "cast(temperature_boiler as double)",
      "cast(pressure_boiler as double)",
      "cast(rpm as double)",
      "cast(power_dynamo as double)",
      "cast(power_heating as double)",
      "cast(valve_aperture as double)").createOrReplaceTempView("EngineData")
```

Um Zeilen mit fehlerhaften Werten herauszufiltern, können nun basierend auf der SQL-Tabelle Abfragen eingesetzt werden.
In diesem Fall werden negative power_dynamo-Werte herausgefiltert:

```
>>> df_cleaned = spark.sql("select * from EngineData where power_dynamo >= 0")
```

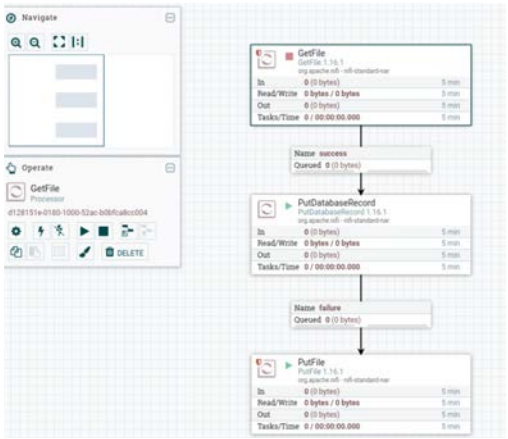
Ein Dataframe kann nach der Bereinigung wieder als CSV-Datei gespeichert werden. Die Einbindung der repartition-Funktion gewährleistet dabei, dass das Ergebnis in einer Datei gespeichert wird, auch wenn der Dataframe vorher partioniert gewesen sein sollte. Dies kann die Folge von parallel ausgeführten Verarbeitungsschritten sein.

```
>>> df_cleaned.repartition(1).write.format('com.databricks.spark.csv').save("/tmp/cleaned.csv",header = 'true')
```

Als Alternative kann der Dataframe auch über die JDBC API in einer Datenbank gespeichert werden.
Durch das folgende Kommando werden die Daten beispielsweise in einer SQLite Datenbank gespeichert:

```
>>> df_cleaned.write.jdbc(url="jdbc:sqlite:/tmp/engine.db", table="data", mode="overwrite")
```

**Umsetzung in NiFi:**
Das hier gezeigte Beispiel zeigt wiederum das Einlesen der CSV-Datei mit den Wärmekraftmaschinendaten und deren Speicherung in einer SQLite-Datenbank. Hier wird die CSV-Datei mithilfe des GetFile-Prozessors eingelesen und in NiFi-Flowfiles umgewandelt. Diese werden in einen PutDatabaseRecord-Prozessor geleitet, der für das korrekte Parsing der CSV-Datei und den Zugriff auf die Datenbank konfiguriert wird. Genauso wie das Verbinden der einzelnen Module, erfolgt auch deren Konfiguration interaktiv in der NiFi-Weboberfläche.




Der abschließende PutFile-Prozessor dient zum Abfangen und Speichern von Fehlerbedingungen, wie beispielsweise falsch formatierte Zeilen in der Eingabedatei. Dadurch wird ermöglicht, dass Fehlerbedingungen leicht in der gespeicherten Textdatei nachvollzogen werden können. ◆

# Agile Softwareentwicklung mittels DevOps-Workflows

– Florian Haßler
Software Engineer in der Unit Domain-specific Applications

Das Entwickeln und Veröffentlichen von Software kann sich aufwendig gestalten. Manuelle Test-, Integrations- und Veröffentlichungsschritte nehmen sehr viel Zeit in Anspruch und sind fehleranfällig. Oft traten deshalb in der Vergangenheit Showstopper erst sehr spät im Projektverlauf in Erscheinung und zwangen alle Beteiligten, einige Schritte zurückzugehen. Um diese Problematik zu vermeiden, kann ein CI/CD-DevOps-Workflow eingesetzt werden.

## Workflow

Die Entwickler*innen wählen ein vom Kunden erfasstes und final spezifiziertes Ticket. Es wird damit begonnen, Tests zu schreiben und in weiterer Folge das Feature in Code zu gießen. All das geschieht fernab der aktiven Codebasis der Applikation in einem abgetrennten Code-Bereich. Bevor das Feature nach Erfüllung aller Akzeptanzkriterien in die aktive Codebasis integriert wird, wird es automatisiert auf Herz und Nieren geprüft. Erst, wenn es geprüft wurde, wird es auf das Testsystem und anschließend auf das Produktivsystem veröffentlicht. Warum automatisiert?

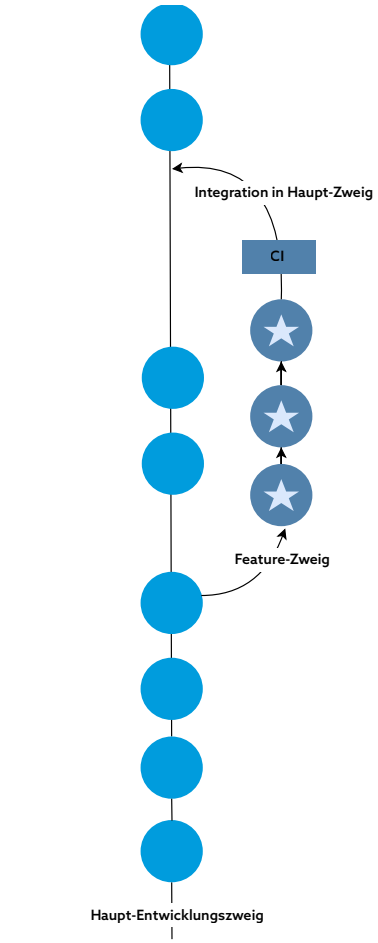
In den Automatisations-Pipelines können Tests einmal definiert werden und damit die verbundenen Anforderungen immer ohne menschliches Zutun sichergestellt werden. Containerisierung hilft zusätzlich noch dabei, die Tests reproduzierbar zu machen. Auch das Veröffentlichen erfolgt automatisiert. Wenige Minuten nach der Umsetzung ist ein neues Feature bereits auf dem Testsystem unserer Kund*innen sichtbar und steht für Anwender*innen-Tests zur Verfügung. Programmierfehler dringen dabei aufgrund automatisierter Tests nur äußerst selten bis zum Testsystem durch. Dadurch müssen sich Kund*innen nicht mit diesen Fehlern beschäftigen und können sich auf die Inhalte der Features konzentrieren. Im weiteren Verlauf beschreibt dieser Artikel nun, wofür die Abkürzungen CI und CD bei den Entwickler*innen der RISC Software GmbH stehen. Mit den hier vorgestellten vier Strategien wird der Arbeitsalltag sowohl für Entwickler*innen als auch Anwender*innen stark vereinfacht.

## Erfahrung mit Pipelines

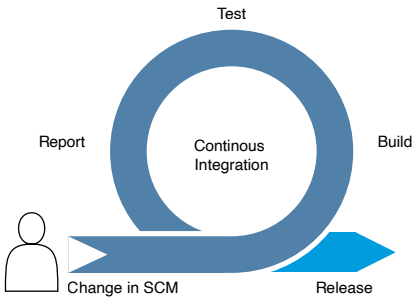
Plattformen zur Umsetzung solcher Automatisations-Pipelines gibt es einige. In der RISC Software GmbH haben wir Erfahrung mit Concourse, Jenkins und GitLab. Speziell bei den Web-Entwicklungs-Teams hat sich gezeigt, dass sich GitLab am besten in ihre Workflows integrieren ließ.

## CI – Continuous Integration

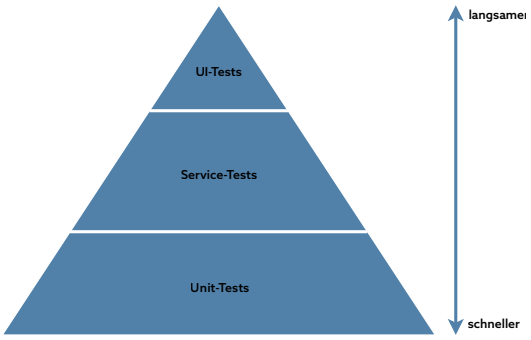
Eine Idee von Continuous Integration ist es, neue Features so schnell und so oft wie möglich automatisiert zu integrieren. Es geht hier um kurze Feedback-Schleifen. Sie helfen den Entwickler*innen, Probleme zu identifizieren und zu beheben, bevor der Kontext gewechselt (also die nächste Aufgabe gestartet) wird. Die Kernfrage lautet: Lässt sich das entwickelte Feature in die bisherige Applikation integrieren, ohne negative Auswirkungen auf bestehende Funktionen zu haben?




Vereinfacht dargestellt spielen Entwickler*innen ihre Änderungen in unserem Source-Code-Management-System (SCM) ein und die CI-Pipeline beginnt zu arbeiten. Wenn sie fertig ist, benachrichtigt sie die Entwickler*innen über Erfolg oder Misserfolg. Abhängig vom Typ eines Projekts kommen verschiedene Tools in der Integrations-Pipeline zum Einsatz. Fast immer wird damit gestartet, dass die Test-Suits entsprechend ihrer Stellung in der Test-Pyramide (lt. Mike Cohn) durchlaufen. Unit-Tests bilden dabei die Basis der Test-Pyramide. Getestet werden kleinstmögliche Verhaltenseinheiten. Sie stellen sicher, dass der Code wie erwartet funktioniert. Anschließend wird mithilfe von Integrationstests sichergestellt, dass die Interaktion zwischen den verschiedenen Teilen der Applikation oder mit externen Komponenten (z.B. Datenbanken) auch funktioniert. Schlägt ein Test

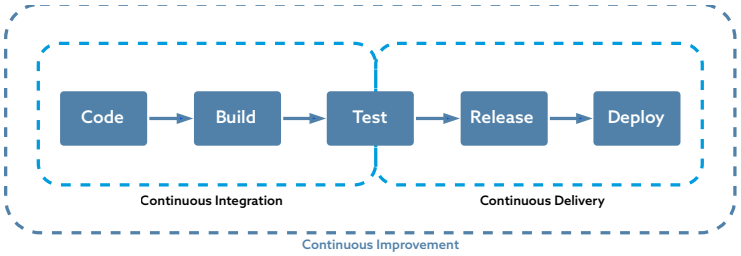


fehl, wird die Test-Pipeline abgebrochen und das Dev-Team erhält eine Benachrichtigung. Beinahe immer ist der letzte Schritt in den RISC-Software-GmbH-CI-Pipelines eine statische Code-Analyse durch ein entsprechendes Tool. Dieses hilft dabei, Schwachstellen und Inkonsistenzen mit allgemein üblichen Entwicklungspraktiken zu erkennen. Das in der RISC Software GmbH verwendete Tool enthält viele eingebaute Regeln für verschiedene Programmiersprachen. Das Ergebnis der Analyse ist eine Bewertung der technischen Qualität des Source-Codes. Es werden auch die Anzahl der Fehler, Schwachstellen, Testabdeckung, Anzahl der geschriebenen Codezeilen und noch vieles mehr ausgegeben. Die Entwickler*innen erhalten eine automatisierte Bewertung ihres Codes.




## CD – wie: Continuous Delivery

Ist die CI-Pipeline erfolgreich durchlaufen und der Report des Code-Analyse-Tools zufriedenstellend ausgefallen, wird die Continuous-Delivery-Pipeline manuell aktiviert. Sie verfolgt das Ziel, den Veröffentlichungs-Prozess automatisiert, schnell und zuverlässig auszuführen. Die in der CI-Pipeline gebauten Artefakte werden gesammelt und auf dem Testsystem veröffentlicht. Dieser Veröffentlichungs-Schritt kann simpel sein, indem nur Source-Code auf einen Server kopiert wird, oder auch komplexer. Das Zielsystem und der Ort des Zielsystems sind für die Komplexität ausschlaggebend.




Ein Vorteil, der im Zusammenhang mit Continuous Delivery oft genannt wird ist, dass die Applikation zu jeder Zeit veröffentlicht werden kann. Es gibt keinen Zeitpunkt, in dem sich in der aktiven Code-Basis des Versionsverwaltungssystems Code befindet, der nicht funktionsfähig ist. Melden sich Kund*innen, weil ein Problem aufgetreten ist, kann das Team zeitnahe darauf reagieren und eine neue Version veröffentlichen. Ein weiterer Vorteil ist natürlich, dass keine klassischen Releases mehr nötig sind, sondern Feature für Feature direkt von Kund*innen getestet und abgenommen werden kann.

## CD – wie: Continuous Deployment

Continuous Deployment führt die Schritte von Continuous Integration und Continuous Delivery einen Schritt weiter. Hier werden alle Änderungen, welche die CI-Pipeline erfolgreich durchlaufen, sofort auch freigegeben. Dieser Prozess ist vollständig automatisiert und nur ein fehlgeschlagener Integrationsschritt verhindert, dass die Änderungen auf das Test- und in weiterer Folge Produktionssystem übertragen werden. Das funktioniert, wenn das Vertrauen in die Entwickler*innen, den Continuous Integration- und den Continuous-Delivery-Prozess sehr groß ist. Continuous Deployment hat seinen Platz hauptsächlich im groß angelegten Produktentwicklungsgeschäft; im klassischen Projektgeschäft sind die dafür notwendigen Voraussetzungen oft nicht wirtschaftlich sinnvoll umzusetzen.

## CI – wie Continuous Improvement

Abbildung 1 zeigt die Benachrichtigung, die einige Dev-Teams am Morgen nach Bekanntwerden der Sicherheitslücke in der Java-Logging-Bibliothek Log4j ereilte. Die Sicherheitschecks der verwendeten Bibliotheken laufen jede Nacht automatisch, um die von uns entwickelten Applikationen sofort nach Bekanntwerden von Sicherheitslücken mit Patches zu versorgen. Außerdem helfen sie, regelmäßig die verwendeten Softwarebibliotheken in von der RISC Software GmbH entwickelten Applikationen zu aktualisieren und so diese up-to-date zu halten. Verbesserungen werden dabei inkrementell umgesetzt, evaluiert und anschließend weiter verbessert. Das Resultat sind schlankere Arbeitsabläufe und damit einhergehend mehr Zeit für die Umsetzung neuer Features.

```
+-----------------------------+---------------+----------+-------------------+---------------+-----------------------------------+
|           LIBRARY           | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |               TITLE               |
+-----------------------------+---------------+----------+-------------------+---------------+-----------------------------------+
| org.apache.logging.log4j:log4j-api | CVE-2021-45046 | CRITICAL | 2.15.0 | 2.16.0 | log4j-core: DoS in log4j 2.x |
|                             |               |          |                   |               | with thread context message       |
|                             |               |          |                   |               | pattern and context...            |
|                             |               |          |                   |               | --->avd.aquasec.com/nvd/cve-2021-45046 |
+-----------------------------+---------------+----------+-------------------+---------------+-----------------------------------+
```

Abbildung 1: Benachrichtigung nach Bekanntwerden der Sicherheitslücke in der Java-Logging-Bibliothek Log4j

# Vorteile

Durch Continuous-Integration, Delivery, Deployment und Improvement hat die RISC Software GmbH die Fähigkeit gewonnen, häufige Releases ohne Qualitätskompromisse zu veröffentlichen. – Die Vorteile unseres DevOps-Workflows für Kund*innen:

- Minimiertes Fehlerrisiko (und Fehler können schneller korrigiert werden)
- Kürzere Reaktionszeiten (Kund*innen-Feedback kann rascher eingebaut werden)
- Geringere Kosten für manuelle Tests (mehr Features für gleich viel Geld)
- Fortschritte im Entwicklungsprozess sind sichtbar
- Sicherheitslücken werden zügig erkannt und beseitigt
- Komponenten, Frameworks, Bibliotheken werden effizienter aktualisiert ◆

# Transformer-Modelle erobern Natural Language Processing

– Fabian Jetzinger, BSc
Data Scientist in der Unit Logistics Informatics

## Wie Sie vortrainierte Modelle wie Google T5 optimal nutzen

Künstliche Intelligenz (KI) ist mittlerweile zum festen Bestandteil unseres Alltags geworden. Täglich sind wir mit zahlreichen Systemen in Kontakt, welche auf KI aufbauen – auch wenn wir uns dessen nicht immer bewusst sind. Auffällig wird es jedoch dann, wenn Maschinen in unserer Sprache mit uns kommunizieren, wie das bei Sprachassistenten der Fall ist. Beim KI-getriebenen Verständnis natürlicher Sprache konnten in den vergangenen Jahren erhebliche Fortschritte erreicht werden, unter anderem durch sogenannte Transformer-Modelle – darunter auch die von Google entwickelte T5-Architektur, welche in diesem Beitrag vorgestellt wird.

Das Feld der natürlichen Sprachverarbeitung (Natural Language Processing, kurz NLP) beschäftigt sich mit dem Verständnis natürlicher, menschlicher Sprache. Einen Einstieg in das Thema bietet der Fachartikel des ersten Insights Magazins „OK Google: Was ist Natural Language Processing?".

Da zum maschinellen Aufbau von Verständnis für natürliche Sprache überaus große Mengen an Textdaten von mehreren Gigabyte nötig sind (bspw. der gesamte Text aus Wikipedia), ist das Training von NLP-Systemen von Grund auf mit erheblichen Kosten (bis zu sechsstelligen Euro-Beträgen oder mehr) und Zeitaufwand verbunden[1], bis eine akzeptable Qualität der Ergebnisse erreicht werden kann. Deswegen stellen zahlreiche Forscher*innen sowie auch Großunternehmen wie Google oder Facebook vortrainierte Sprachmodelle öffentlich zur Verfügung. Da diese sogenannten Basismodelle bereits ein grundlegendes Verständnis für Sprache erlernt haben, können andere Forscher*innen auf diesen Modellen aufbauen und sie für ein konkretes Vorhaben anpassen, erweitern und wiederum mit der NLP-Community teilen.

Dieses Konzept wird als Transfer-Learning bezeichnet. Hierbei werden Modelle mit riesigen Datenmengen darauf trainiert, ein allgemeines Verständnis für grundlegende Konzepte (in diesem Fall natürliche Sprache) zu erlangen, und anschließend mit spezielleren Daten darauf trainiert, eine konkrete Aufgabe zu erledigen. Dies erlaubt nicht nur die Wiederverwendung von Modellen, sondern verringert auch die Größe der für eine konkrete Aufgabe benötigten Datenbasis.

Wenn nur wenige Daten zur Verfügung stehen, kann man auf Zero-Shot-Learning zurückgreifen. Hierbei wird ein Modell dazu gebracht, eine Aufgabe zu erfüllen, auf welche es nicht explizit trainiert wurde. Nach diesem Prinzip arbeiten auch One-Shot- und Few-Shot-Modelle. Somit können Modelle mit keinen oder nur wenigen Daten auf eine bestimmte Aufgabe getrimmt werden. Leider ersparen diese Ansätze das Sammeln von Trainingsdaten in sehr vielen Fällen nicht völlig: erstens dienen diese Zero-Shot-Modelle oftmals nur als Prototypen bzw. Baseline-Modelle, da diese häufig nicht ausreichend gut generalisieren können (d.h. sinkende Qualität der Ergebnisse bei neuen, ungesehenen Daten) und sehr anfällig für Fehler in den (Trainings-)Daten sind. Zweitens sind zusätzlich Evaluierungsdaten nötig, um die Qualität der Zero-Shot-Modelle überhaupt erst quantifizieren zu können und eine Einschätzung für den durch ihren Einsatz erzielten Mehrwert treffen zu können.

Einen der größten Durchbrüche der letzten Jahre im NLP-Bereich stellt die Entwicklung sogenannter Transformer-Modelle dar. Hierbei handelt es sich um eine besondere Architektur von neuronalen Netzen, welche sogenannte Attention-Mechanismen verwendet und die zuvor vorherrschenden rekurrenten neuronalen Netze (RNN) durch tiefe Feed-Forward-Netze ersetzt. Diese neuartige, 2017 entwickelte Architektur konnte die Schwächen der Vorgängermodelle weitgehend ausmerzen. Sie ermöglicht u.a. ein (besseres) Verständnis über den Kontext bestimmter Wörter und erleichtert den performanten Umgang mit größeren Datenmengen. Das wohl bekannteste Beispiel einer Gruppe von Transformer-Modellen stellt BERT[2] (Bidirectional Encoder Representations from Transformers) dar. Die grundlegenden BERT-Modelle können um maßgeschneiderte Erweiterungen ergänzt werden, welche auf die gewollte Aufgabe (z.B. Klassifizierung von Sätzen auf negative, neutrale oder positive Stimmung) trainiert werden. Auf der BERT-Architektur basierende Systeme konnten seit der Publikation im Jahr 2018 in zahlreichen Aufgaben rekordbrechende Ergebnisse erzielen und sind mittlerweile fester Bestandteil der Google-Suche.

### Was ist Google T5?

Die von Google entwickelte T5-Architektur[3] (Text-To-Text-Transfer-Transformer) funktioniert sehr ähnlich wie BERT, weist aber einige Unterschiede auf. Das namensgebende Text-To-Text-Prinzip bedeutet, dass bei T5-Modellen Ein- und Ausgabe aus reinen Textdaten bestehen. Dies erlaubt das Trainieren der T5-Modelle auf beliebige Aufgaben, ohne die Modellstruktur selbst auf diese Aufgabe anpassen zu müssen. So kann jedes Problem, welches als Text-Eingabe zu Text-Ausgabe formulierbar ist, durch T5 bearbeitet werden. Dazu zählen beispielsweise Klassifizierung von Texten, Zusammenfassung eines langen Textes, oder die Beantwortung von Fragen über den Inhalt eines Textes. Eine weitere Besonderheit der T5-Architektur ist die Möglichkeit, mit einem einzigen Modell meh-



*[1] Sharir, Or, Barak Peleg, and Yoav Shoham. „The cost of training nlp models: A concise overview." arXiv preprint arXiv:2004.08900 (2020).*
*[2] Devlin, Jacob, et al. „Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).*

rere unterschiedliche Aufgaben lösen zu können, wie in Abbildung 1 dargestellt. So wurden die vortrainierten T5-Modelle bereits auf 17 verschiedenen Aufgaben trainiert. Darunter stellt vor allem die Aufgabe, Fragen zu einem gegebenen Text zu beantworten, eine besondere Stärke von T5 dar. Wird dem Modell der gesamte Wikipedia-Artikel zur Geschichte Frankreichs zur Verfügung gestellt, so kann bspw. auf die Frage „Wer wurde 1643 König von Frankreich?" erfolgreich die richtige Antwort „Ludwig XIV" zurückgegeben werden.

T5 konnte in unterschiedlichen Bereichen bereits beeindruckende Resultate erzielen, ist allerdings wie in allen Bereichen der Künstlichen Intelligenz keine Lösung für jedes Problem (siehe auch No-Free-Lunch-Theorem). Ein gängiger Ansatz zur Bestimmung der besten Lösung eines Problems ist das Testen mehrerer Modellarchitekturen für eine Aufgabe mit anschließendem Vergleich der Resultate.

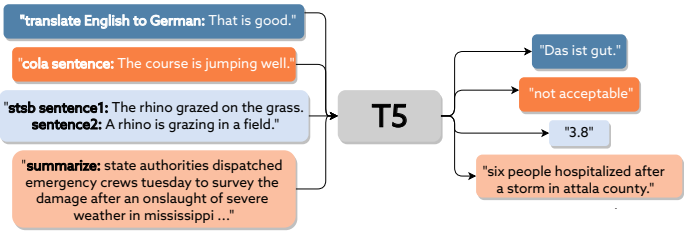



Abbildung 1: T5-Modell

## Parlez-vous français? Oui!

Eine Erweiterung der T5-Modelle stellen die mT5-Modelle[4] dar, welche auf riesigen Mengen von Texten in insgesamt 108 unterschiedlichen Sprachen (Stand 2022-01) trainiert wurden. Dies erlaubt es einem einzigen Modell Aufgaben in unterschiedlichen Sprachen zu lösen. Hierbei ist allerdings auch ein deutlicher Unterschied in der Qualität der Resultate in unterschiedlichen Sprachen erkennbar – je weniger eine Sprache in den Trainingsdaten vorhanden ist, umso schlechter sind die darin erreichten Ergebnisse.

Besonders nützlich ist dabei auch, dass Sprachen, für die weniger Daten zur Verfügung stehen, von den Trainingsdaten in anderen Sprachen profitieren können. Dies ist vor allem bei der Datenakquise hilfreich, da englischsprachige Texte oft in größeren Mengen verfügbar sind als Texte in anderen Sprachen. Selbst wenn Aufgaben nur in einer Sprache gelöst werden sollen, können mT5-Modelle somit dennoch hilfreich sein, falls in der Zielsprache nicht genügend Trainingsdaten vorhanden sind. Die dabei erzielten Resultate sind zumeist allerdings eher schlechter als wenn mit einer ähnlichen Menge an Gesamtdaten nur auf der Zielsprache trainiert wird.

## Wie verwende ich T5 für meinen Use-Case?

In einem ersten Schritt sollten einige allgemeine, wichtige Fragen über den konkreten Use-Case geklärt werden:

- In welchen Sprachen ist das Problem zu lösen?
- Woher stammen die für das Training nötigen Daten? Sind diese für meinen Use-Case geeignet?
- Wie kann ich die Qualität des Modells überprüfen?
- Sind Testdaten vorhanden?
- Welche Ressourcen (Rechenleistung, Zeit, finanzielle Mittel etc.) stehen für ein Training bzw. Fine-Tuning zur Verfügung?
- Welche Modelle sind für meinen Use-Case anwendbar?

Für den potenziellen Einsatz von T5 muss zusätzlich noch evaluiert werden, ob der Use-Case als Text-To-Text-Aufgabe formuliert werden kann.

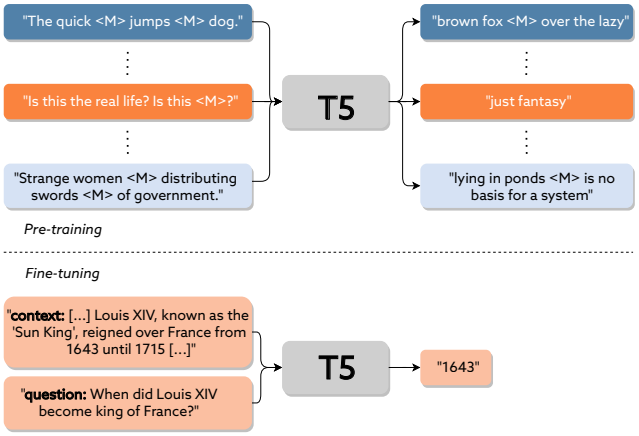



Abbildung 2: Pre-training und Fine-tuning

Als nächstes wird ein passendes, vortrainiertes T5-Modell gewählt. Eine der besten und bekanntesten Quellen dafür ist die Plattform Hugging Face, welche eine Vielzahl an vortrainierten State-of-the-Art-Modellen und auch Datensätze als Open-Source-Lösungen für die Öffentlichkeit bereitstellt. Diese stehen häufig in unterschiedlichen Größen zur Verfügung, wobei eine Balance zwischen der benötigten Rechenleistung bzw. -zeit und der Qualität des Modells gefunden werden muss. Größere Modelle bieten zwar im Allgemeinen bessere Resultate, stellen allerdings auch signifikant größere Anforderungen an Ressourcen, weswegen nicht automatisch immer das größte Modell die beste Wahl ist.

Nun geht es ans sogenannte Fine-Tuning. Dabei wird das vortrainierte Modell (welches zu diesem Zeitpunkt bereits über ein grundlegendes Verständnis für Sprache verfügt) darauf trainiert, eine konkrete Aufgabe, wie in Abbildung 2 dargestellt, zu lösen. Hierfür sind Trainingsdaten notwendig, welche dem Modell vorzeigen, welche Eingabe zu welcher Ausgabe führen soll. Dabei ist nicht nur die Menge, sondern auch die Qualität der Daten von entscheidender Bedeutung.

Herausforderungen sind unter anderem die Wahl des konkreten Basismodells, die Beschaffung und Qualitätskontrolle der Trainingsdaten und das Aufbereiten der Daten in ein vom Modell lesbares Format. Zusätzliche Schwierigkeiten kommen beispielsweise bei längeren Textsequenzen hinzu, da Transformer-Modelle über ein Limit bezüglich Textlänge verfügen.

In der RISC Software GmbH wird der Einsatz von T5-Modellen seit längerem erforscht. So konnte ein System zur Erkennung und Zuordnung von Eigennamen in Texten (Named Entity Recognition, kurz NER) durch T5 erweitert und verbessert werden. Hierzu kommt die bereits vortrainierte Fähigkeit von T5, Fragen zu einem gegebenen Text zu beantworten, zum Einsatz. So kann das T5-Modell Fragen wie „Welche Person ist betroffen?" oder „Welches Unternehmen ist involviert?" beantworten und somit dem NER-System unter die Arme greifen.

Die T5-Architektur eignet sich aufgrund ihres Designs für eine Vielzahl unterschiedlichster Aufgaben (oder Kombination dieser) und konnte bereits in zahlreichen Anwendungsgebieten beeindruckende Resultate liefern. Es bleibt weiterhin äußerst spannend, wie sich diese Technologien in der Zukunft weiterentwickeln werden und welche Erfolge damit noch erzielt werden können. ◆

*[3] Raffel, Colin, et al. „Exploring the limits of transfer learning with a unified text-to-text transformer." arXiv preprint arXiv:1910.10683 (2019).*

*[4] Xue, Linting, et al. „mT5: A massively multilingual pre-trained text-to-text transformer." arXiv preprint arXiv:2010.11934 (2020).*

# Technische Schuld und Legacy-Systeme

– DI (FH) Andreas Lettner
Head of Unit Domain-specific Applications und Head of Coaches

## Wann fördern Verträge diese Entwicklung?

Überall, wo Software zum Einsatz kommt, ist das Risiko vorhanden, dass Alt-Systeme zu Legacy-Systemen mutieren. Dies bedeutet, dass sie nicht mehr im Rahmen vertretbarer Kosten und ohne vorhersehbares technisches Risiko weiterentwickelt oder gewartet werden können. Wie kann also vermieden werden, dass ein System oder Alt-System zum Legacy-System wird? Lesen Sie im folgenden Artikel, wie Legacy-Systeme rechtzeitig erkannt werden, wie man diese entfernt und letztendlich vermeidet.

### Wie werden Legacy-Systeme erkannt?

Ein essenzielles Qualitätskriterium für Software ist eine langfristige Wart- und Erweiterbarkeit. Durch einen Fokus auf dieses Kriterium kann sichergestellt werden, dass künftige Erweiterungen in Bezug auf Kosten, Umsetzungszeit, Wirtschaftlichkeit und Risiko akkurat eingeordnet werden können. Abbildung 1 zeigt anhand der grünen Linie eine optimale Kostenentwicklung in Bezug auf das Alter der Software. Ein initialer Anstieg der Kosten ist damit verbunden, dass Investitionen in die Qualität getätigt werden, welche mittelfristig die Kosten stabilisieren, sowie die Software am Leben erhalten.

Im Unterschied dazu zeigt die rote Linie eine mögliche Entwicklung für ein System, wo Qualitätskriterien vernachlässigt werden. Jeder rote Punkt zeigt eine Entscheidung in der Entwicklung, wo ein Kompromiss zwischen Qualität und Kosten oder eventuell Umsetzungszeit eingegangen wurde. Dadurch entsteht in der Software etwas, das unter dem Begriff „technische Schuld" bekannt ist. Diese technischen Schulden führen dazu, dass die Komplexität künftiger Wartungen und Weiterentwicklungen erhöht wird und somit auch die Änderungskosten, sowie das damit verbundene Risiko. Technische Schulden führen erfahrungsgemäß schnell dazu, dass Teile eines Systems oder das System als Ganzes nicht mehr wart- oder erweiterbar werden.

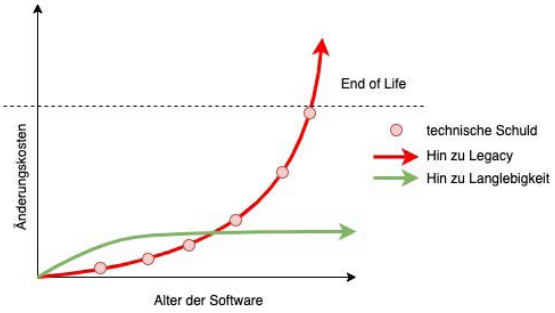



Abbildung 1: Änderungskosten

Die Entwicklung eines Systems hin zu einem Legacy-System ist allgemein gut messbar. Unter anderem können hier folgende Anzeichen beobachtet werden:

- Die Kosten und Umsetzungszeiten für Änderungen steigen über die Zeit an. Vor allem bei Aufgaben mit vergleichbarem Inhalt ist dies gut beobachtbar.
- Es ist ein Anstieg der Fehlerquote im Produktivsystem bei bzw. nach Versionsupdates messbar.
- Die Termine für Produktivsetzungen können nicht konstant eingehalten werden.
- Die Entwickler*innen zeigen Unsicherheiten bei den Schätzungen.
- Es häufen sich Fragen nach der Machbarkeit.
- Es entwickelt sich bei Produktverantwortlichen und Entwickler*innen eine Scheu, Veränderungen voranzutreiben.
- Es wird vermehrt von Workarounds gesprochen.

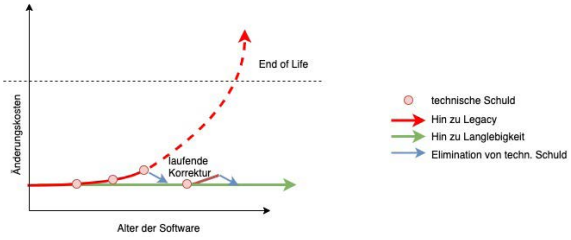



Abbildung 2: konstantes Qualitätsmanagement

### Wie geht man mit einem bestehenden Legacy-System um?

Technische Schuld wird im Rahmen einer Softwareentwicklung immer anfallen. Dies ist nicht vermeidbar. Jedoch können die Prozesse in der Softwareentwicklung derart priorisiert werden, dass technische Schulden abgebaut werden können. Hierfür ist ein gemeinsames Mindset im Projektteam notwendig. Eine konstante Überwachung der technischen Schulden und die Ermächtigung der Entwickler*innen, diese wieder entfernen zu können/dürfen, sind unumgänglich

hierfür. Methoden, welche hier Anwendung finden, sind unter anderem Code Reviews, Coding Guidelines, Pair Programming, Refactoring und Test Driven Development. Abbildung 2 zeigt die konstante Korrektur der technischen Schulden durch kleine Anpassungen.

Je nachdem, wie weit sich das Softwareprodukt auf der Reise in Richtung „End of Life" befindet, können kleinere Anpassungen ggf. nichts mehr bewirken und größere Refactorings oder Reengineerings notwendig werden. In diesem Fall ist es hilfreich, sich Reengineering-Spezialist*innen ins Projektteam zu holen, die nach und nach ein konsistentes Reengineering der bestehenden Alt-Systeme durchführen. Abbildung 3 zeigt das späte Erkennen technischer Schulden. Hier muss mit deutlich mehr Aufwand und Zeit gegengesteuert werden, um das System wieder auf den richtigen Pfad zu bringen.

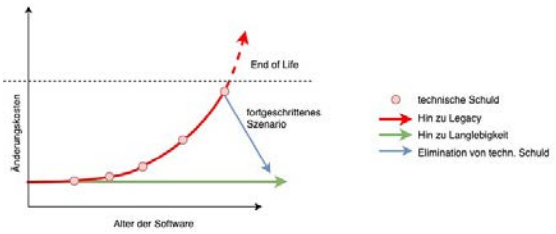



Abbildung 3: Reengineering, größeres Refactoring

## Wie vermeidet man ein Legacy-System?

Neben der Beobachtung technischer Schulden stellt sich die Frage, wie technische Schulden vermieden werden können. Dafür muss analysiert werden, wie technische Schulden zustande kommen. Wo liegen die Ursachen hierfür und was kann getan werden, um diese Ursachen zu entschärfen?

Wie im vorigen Abschnitt festgehalten, sind die Entwickler*innen diejenigen, welche für die technische Qualität eines Softwareproduktes zuständig sind. Technische Schuld wird daher nicht vorsätzlich in ein System integriert, sondern durch externe Faktoren gefördert. Betrachtet man den zeitlichen Verlauf einer Softwareentwicklung, so kann folgender Zusammenhang festgestellt werden: Die blaue Linie zeigt die lineare Projektabwicklung, d.h. die Erfüllung des geplanten Projektumfanges bis zum Lieferzeitpunkt bei V1.0. Die Ideallinie in der Realität sollte der grünen Linie entsprechen. Es gibt kontinuierliche Anpassungen in der Umsetzung, wodurch die Produktentwicklung „auf Kurs" gehalten und so sichergestellt wird, dass ein wertvolles, lauffähiges Produkt zum geplanten Termin zur Verfügung steht.

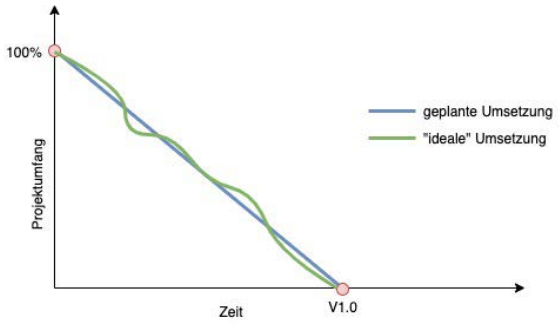



Abbildung 4: Entwicklungsverlauf

Abbildung 5 zeigt ein Bild, welches in der Realität jedoch meist vorzufinden ist. Eine Abweichung vom Plan wird bemerkt. Wie es dazu kam, kann unterschiedliche Gründe haben. Eventuell war eine initiale Schätzung nicht akkurat, das Team hat an Effizienz verloren, es gibt äußere Einflüsse wie z.B. Krankenstände, die Komplexität der Software hat sich geändert, der Projektumfang hat sich geändert, etc. Die Gründe hierfür können vielfältig sein. An dieser Stelle wird jedoch interessant, welche Optionen ein Projektteam hat. Folgende primären Entscheidungen können unter Berücksichtigung der Gegebenheiten getroffen werden.

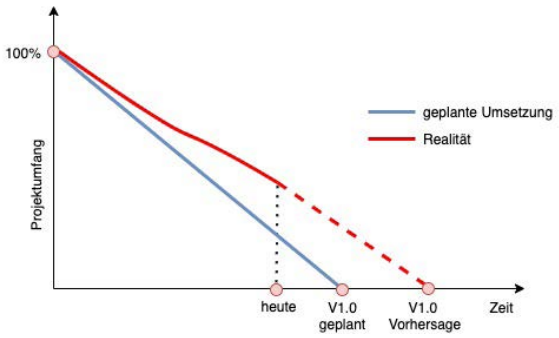



Abbildung 5: Realität

## Option 1: Die Produktverantwortlichen verschieben das Fertigstellungsdatum

Die Verschiebung des Fertigstellungsdatums erfolgt nur, sofern die Produktverantwortlichen die Möglichkeit dazu haben. Hinter einer Verschiebung des Fertigstellungstermins stehen üblicherweise andere Personen oder Systeme, welche davon abhängig sind und mehr oder weniger auf den ursprünglich geplanten Termin bestehen. Dazu kommt noch der Nachteil, dass nicht nur eine Produktivsetzung und somit der (eventuell wirtschaftliche) Nutzen zeitlich verschoben wird, sondern auch noch das Projektteam für einen längeren Zeitraum finanziert werden muss. Dies führt zu zusätzlichen Kosten und späteren Einnahmen. Eine direkte, negative Auswirkung auf den ROI ist die Folge.

## Option 2: Die Entwicklungsgeschwindigkeit wird erhöht

Die wenigsten Widerstände dürfte man sich erwarten, wenn man es schafft, die Arbeitsgeschwindigkeit der Entwickler*innen dahingehend zu optimieren, dass der Fertigstellungszeitpunkt und die Entwicklungskosten eingehalten werden. Tatsächlich ist das eine Option, wobei man hier zwei unterschiedliche Szenarien unterscheiden muss.

**Option 2a**: Es gibt Einflussfaktoren, welche das Team in der Effizienz stört, welche auch tatsächlich behoben werden können. Diese Maßnahmen funktionieren und sind auch jene, welche bei einer kontinuierlichen Prozesskontrolle und -verbesserung stattfinden. Das Szenario in Abbildung 4 setzt eben diese

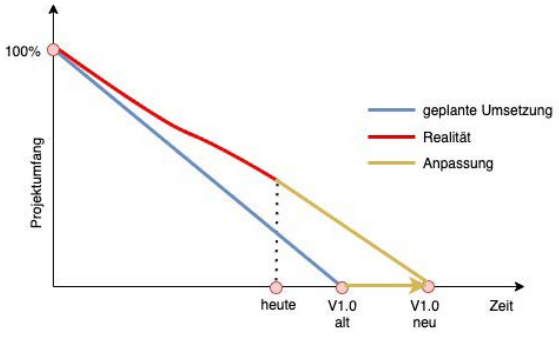



Abbildung 6: Releasedatum verschieben

voraus. Voraussetzung für diese Maßnahmen sind folglich aber auch, dass die Abweichungen nicht zu spät erkannt werden und auch nicht zu groß sind. In Abbildung 7 könnte es eventuell schon zu spät für eine solche Maßnahme sein.

**Option 2b**: Es gibt zwar störende Einflussfaktoren, diese werden jedoch nicht behandelt oder es ist zu spät für entsprechende Maßnahmen. In diesem Fall wird eine Erhöhung der Entwicklungsgeschwindigkeit lediglich durch den Aufbau von Druck auf die Entwickler*innen erreicht. Druck führt üblicherweise dazu, dass in den Methoden eingespart wird, welche keinen offensichtlich direkten Einfluss auf den Produktumfang einer Software haben. Dies sind jene, welche zur Erhaltung der Qualität notwendig sind und somit kommt es sukzessive zu einem Aufbau von technischer Schuld. Da es auch an Zeit und Kapital für den Abbau der technischen Schulden fehlt, schreitet dieser Aufbau gegebenenfalls mit entsprechender Geschwindigkeit voran.

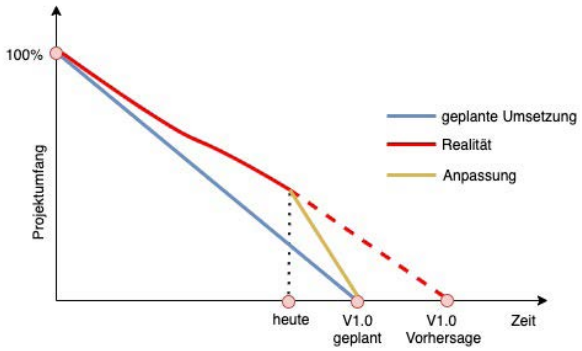



Abbildung 7: Erhöhung der Entwicklungsgeschwindigkeit

### Option 3: Verminderung des Projektumfangs

Als dritte Option besteht die Möglichkeit, die Inhalte des Softwareproduktes anzupassen. Eine entsprechende Befugnis der Produktverantwortlichen ist hierfür jedoch Voraussetzung, bzw. werden hierfür gegebenenfalls entsprechend intensive Verhandlungen zwischen dem bzw. der Produktverantwortlichen und den Stakeholdern benötigt. Anpassungen dieser Art werden durch diverse Maßnahmen erleichtert:

- Variabler Projektumfang – variable Inhalte
- Laufende Priorisierung der benötigten Funktionen
- Entwicklung der Funktionen entsprechend ihrer Prioritäten
- Kontinuierliche Beobachtung der Geschwindigkeit, Timeline, Inhalte, ...

Tatsächlich ist eine solche Anpassung im Projektumfang zu jeder Zeit in der Umsetzung möglich. Rechtzeitig erkannt, sind die Änderungen marginal, spät erkannt, sind die Änderungen entsprechend umfangreicher. Hier gilt: Sofern das Produkt bereits die wertvollsten Funktionen beinhaltet, sollte eine Produktivsetzung mit vermindertem Funktionsumfang kein Hindernis sein. Durch entsprechend gut geschulte und erfahrene Produktverantwortliche kann dieses Risiko deutlich minimiert oder gar vermieden werden.

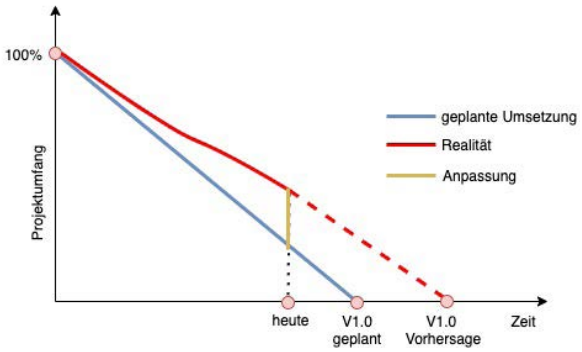



Abbildung 8: Anpassung des Projektumfangs

### Welche Option ist die Beste?

Wie so oft kann hier keine pauschale Aussage getroffen werden. Eventuell macht eine der Optionen für sich alleine Sinn oder eine entsprechende Kombination der Optionen 1, 2a und 3 führt zu einer entsprechenden Korrektur des Kurses. Lediglich Option 2b sollte vermieden werden, da genau diese letztendlich ein System in Richtung „End of Life" entwickelt.

Maßnahmen, welche letztendlich ein Legacy-System verhindern sind

- eine starke und präsente Führung durch eine*n erfahrene*n und ermächtigte*n Produktverantwortliche*n,
- der kontinuierliche Fokus auf initiative und proaktive Verbesserung durch einen Coach und
- bewusste technische Qualitätssicherungsmaßnahmen durch die Entwicklerinnen und Entwickler.

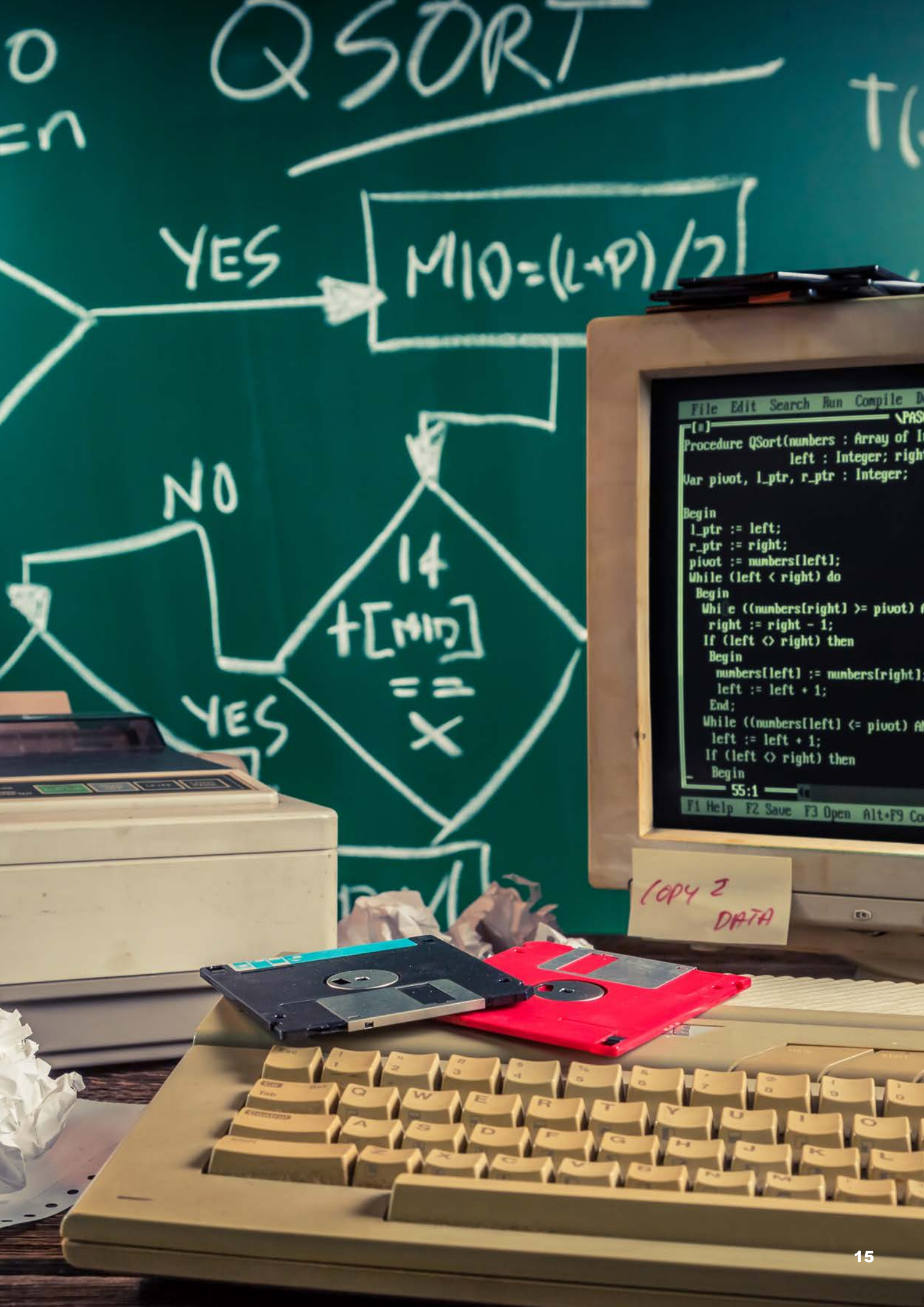
### Warum bindet man sich vertraglich an technische Schuld?

Abschließend soll noch ein wichtiger Einflussfaktor genannt werden, welcher die Optionen 1 – 3 entsprechend einschränken kann: Der Vertrag. Es ist verständlich, dass Auftraggeber*innen vertragliche Sicherheit in Hinblick dahingehend haben wollen, welcher Umfang in welchem Zeitraum zu welchen Kosten geliefert wird. Aus diesem Grund werden oftmals Verträge mit den Auftragnehmer*innen geschlossen, wo alle drei Faktoren fixiert werden. Somit hat der bzw. die Produktverantwortliche aber keinerlei Handlungsspielraum:

- Das Releasedatum steht fest – es darf keine Verspätung geben (Option 1)
- Die Kosten stehen fest – es darf keine Mehrarbeit durchgeführt werden (Option 1, Option 2a)
- Der Umfang steht fest – es dürfen keine Features entfernt werden (Option 3)

Somit bleibt, vertraglich betrachtet, meist nur noch Option 2b, nämlich die Erhöhung des Drucks auf die Entwickler*innen und die damit verbundene Minimierung der Qualität des Produktes.

Empfohlen wird hier vielmehr die Fixierung von Releasedatum und Kosten (z.B. durch Stabilisierung des Teams) und die Schaffung von Flexibilität in den Anforderungen an ein Produkt. Durch die Führung eines bzw. einer Produktverantwortlichen können Auftraggeber*innen den benötigten Funktionsumfang eines Softwareprodukts bis zum Releasedatum steuern und erhalten dadurch eine transparente Kostenkontrolle. Zusätzlich kann das Produkt während der Umsetzung an die Marktbedürfnisse angepasst und so zielgruppenorientiert positioniert werden. ◆


QSORT
YES
NO
YES
14
+[MID]
==
X
File Edit Search Run Compile
Procedure QSort(numbers : Array of
left : Integer; righ
Var pivot, l_ptr, r_ptr : Integer;
Begin
l_ptr := left;
r_ptr := right;
pivot := numbers[left];
While (left < right) do
Begin
right := right - 1;
If (left <> right) then
Begin
numbers[left] := numbers[right];
left := left + 1;
End;
left := left + 1;
If (left <> right) then
Begin
55:1
F1 Help F2 Save F3 Open
COPY 2
DATA

# Zeitreihenanalyse – Aber Richtig!

– DI Dr. Alexander Maletzky
Researcher & Developer in der Unit Medical Informatics

## Wie die Wahl der Trainingsdaten die Praxistauglichkeit von Modellen beeinflusst.

Zeitreihendaten, beispielsweise Maschinendaten in der Industrie oder Vitalparameter in der Medizin, sind heutzutage eine wichtige Datenquelle zur Analyse komplexer Systeme. Moderne Analysesysteme fußen meist auf Methoden des maschinellen Lernens, also auf gelernten Vorhersagemodellen, und greifen auf diese Datenquellen zurück. Für die Entwicklung praxistauglicher Modelle ist die richtige Wahl der Trainingsdaten jedoch eine herausfordernde Aufgabe.

### Das Problem: Die richtige Länge der Sequenzen

Zeitreihendaten werden üblicherweise in regelmäßigen Abständen automatisiert von Sensoren aufgezeichnet, und können wie in Abbildung 1 als Liniendiagramm visualisiert werden. Wie im Fachbeitrag Explorative Datenanalyse mit Zeitreihen (S.18) erläutert, ist die visuelle Inspektion von Zeitreihendaten ein wichtiger Schritt im Datenanalyse-Workflow, der einige Schwierigkeiten mit sich bringt. Noch herausfordernder ist allerdings die automatische Zeitreihenanalyse, bei der ein KI-Modell eigenständig Zeitreihen klassifiziert, Anomalien entdeckt, oder den zukünftigen Verlauf einer Zeitreihe vorhersagt. Modelle dieser Art basieren heutzutage meist auf Methoden des maschinellen Lernens, d.h. sie „lernen" selbstständig anhand von Trainingsdaten, die richtigen Entscheidungen zu treffen. Eine der Hauptaufgaben der Entwickler*innen der Modelle ist dabei – neben der Wahl der geeigneten Modellklasse und -parameter – vor allem die Wahl der Trainingsdaten. Zeitreihen liegen nämlich üblicherweise als lange Sequenzen von Messwerten vor, die sich über längere Zeiträume erstrecken. Je nach Anwendungsgebiet sollen Modelle aber bereits anhand vergleichsweise kurzer Ausschnitte valide Entscheidungen treffen können, und müssen somit auch auf solchen Ausschnitten trainiert werden – und wie diese ausgewählt werden, hat einen starken Einfluss auf die Praxistauglichkeit der resultierenden Modelle. Die Wahl muss nämlich einerseits so erfolgen, dass keine sogenannte Stichprobenverzerrung entsteht, d.h. die Samples die verschiedenen Aspekte der Zeitreihe (Kurvenmorphologie, Periodizität, Trends, etc.) adäquat widerspiegeln. Andererseits sollen die trainierten Modelle aber gerade auf jenen Ereignissen, die für den Anwender besonders interessant sind, korrekt funktionieren. Wenn diese nur selten vorkommen, müssen sie bei der Modellerstellung entsprechend überproportional berücksichtigt werden, was wiederum zu einer Stichprobenverzerrung führen kann.

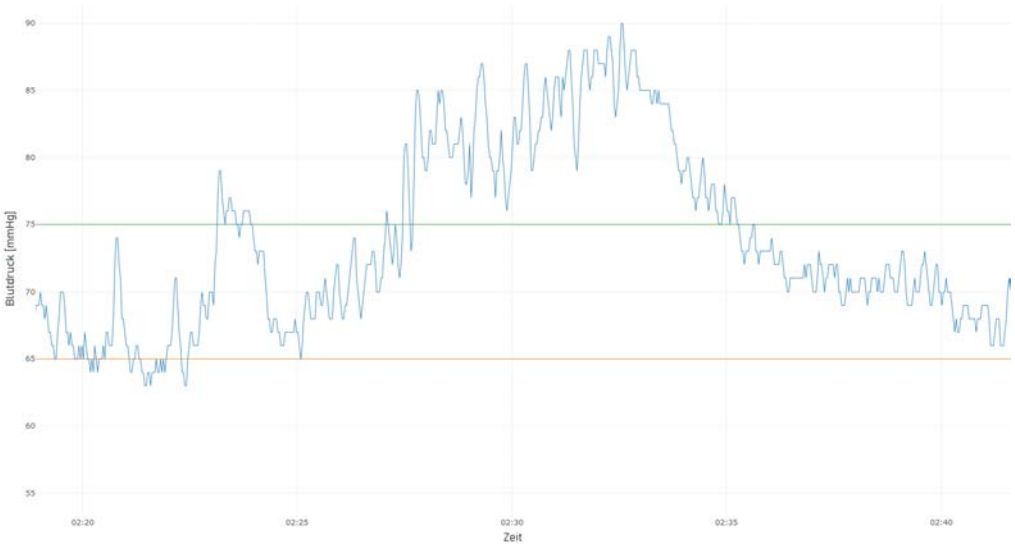



Abbildung 1: Mittlerer arterieller Blutdruck (MAP) eines Intensivpatienten für 30 Minuten, mit einem Messwert pro Sekunde. Die grüne Linie gibt den Wert an, über dem der Blutdruck als normal gilt, und die orange Linie jenen Wert, der einen kritischen Blutdruckabfall darstellt.

## Ein konkretes Beispiel aus der Intensivmedizin

In der Intensivmedizin wird der Zustand von Patienten kontinuierlich überwacht, um im Bedarfsfall ein schnelles Einschreiten des Pflegepersonals zu ermöglichen. Besonderes Augenmerk wird hierbei auf akute hypotensive Episoden (AHEs) gelegt, d.h., kritische Blutdruckabfälle, die zu irreparablen Schäden führen können. Die Vorhersage von zukünftigen AHEs in Form eines Frühwarnsystems, um bereits vor deren Eintritt Gegenmaßnahmen setzen zu können, ist ein aktuell viel beachtetes Forschungsthema im Bereich der Künstlichen Intelligenz [1, 2]. Auch Forscher*innen der Abteilung Medizin-Informatik der RISC Software GmbH beschäftigen sich aktuell im Projekt MC³ gemeinsam mit den Forschungspartner*innen vom MedCampus III des Kepler Universitätsklinikums und dem Institut für Machine Learning der JKU Linz mit dieser Fragestellung.

## Modellentwicklung: Sample-Auswahl für eine hohe Klassifikationsgenauigkeit

Eine mögliche Strategie zur Auswahl der Trainingssamples orientiert sich an der Zeitreihe des mittleren arteriellen Blutdrucks (MAP; siehe Abbildung 1): Immer kurz bevor der MAP unter den kritischen Wert von 65mmHg fällt, wird ein positiv gelabeltes Sample ausgewählt, also ein kurzes Beobachtungsfenster, anhand dessen das Modell später den bevorstehenden Abfall vorhersagen und Alarm auslösen können soll. Bleibt der MAP stattdessen für einen längeren Zeitraum konstant über 75mmHg, wird darin ein negativ gelabeltes Sample ausgewählt, d.h. hier soll das Modell keinen Alarm auslösen. In Abbildung 2 wird die Auswahl der Samples schematisch dargestellt. Als Eingangsdaten für das Modell dienen in beiden Fällen verschiedenste Zeitreihendaten des Patienten im Beobachtungsfenster, z.B. MAP, Herzfrequenz und Sauerstoffsättigung.

Klassifikationsmodelle, die auf diesen Trainingssamples trainiert werden, erreichen eine hohe Klassifikationsgenauigkeit auf dem unabhängigen Test-Set (das nach dem gleichen Schema generiert wird wie das Trainings-Set). Einem Einsatz in der Praxis steht somit nichts mehr im Weg.

## Einsatz in der Praxis: Wo liegt der Fehler?

Natürlich wurde das entwickelte Modell nicht sofort im Krankenhaus eingesetzt, sondern der Praxiseinsatz zunächst in einer Testumgebung simuliert. Dabei hat sich gezeigt, dass das Modell fast ununterbrochen Alarm auslöst, selbst wenn weit und breit keine AHE in Sicht ist. Obwohl die Klassifikationsgenauigkeit auf dem Test-Set sehr gut ist, funktioniert das Modell in der Praxis nicht.

## Fehleranalyse: Auswahl der Trainingssamples

Woran liegt die Praxisuntauglichkeit des Modells? Wie sich gezeigt hat, enthalten die Trainingssamples nur „Extrembeispiele", die sich zwar leicht klassifizieren lassen, aber nur einen kleinen Teil des Spektrums der in der Realität auftretenden Möglichkeiten abdecken. Der MAP ändert sich nämlich meistens nur langsam, d.h., ist am Ende des Beobachtungsfensters von positiv gelabelten Samples üblicherweise deutlich niedriger als bei negativ gelabelten Samples. Das Modell ignoriert sämtliche Zeitinformation und alle anderen Zeitreihen und achtet ausschließlich auf den letzten verfügbaren MAP-Wert: Ist dieser eher hoch, wird kein Alarm ausgelöst, andernfalls schon. Das funktioniert in der Praxis nicht, da sich der MAP dann vielfach in einer „Grauzone" bewegt, die in den Trainingssamples nicht vorkommt.

## Wie kann man es besser machen?

Eine Stichprobenverzerrung lässt sich dadurch vermeiden, dass die Trainingssamples entweder zufällig oder regelmäßig (z.B. alle 10 Minuten) ausgewählt werden, ungeachtet des MAP. So ein Ansatz bringt jedoch andere Probleme mit sich: Einerseits ist die Einteilung eines Samples in „positiv" (MAP wird unter kritischen Wert fallen) und „negativ" (MAP bleibt normal) nicht mehr so einfach, denn was tun, wenn der MAP zwar über 65mmHg bleibt, aber nur knapp? Es macht daher mehr Sinn, kein Klassifikations-, sondern ein Regressionsmodell zu trainieren, das beispielsweise den genauen MAP-Wert 15 Minuten später vorhersagen soll.

Ein anderes Problem ist das im Rahmen der Fehleranalyse entdeckte Phänomen, dass der MAP meistens nur langsam abfällt. Aus medizinischer Sicht sind nämlich genau jene (seltenen) Fälle interessant, wo der MAP rapide abfällt, weil ein Frühwarnsystem nur in solchen Fällen Sinn macht. Um das Modell für solche Situationen zu „sensibilisieren", kann ihnen beim Training eine höhere Wichtigkeit beigemessen werden.Forscher*innen des Projekts MC³ sind gerade dabei, Vorhersagemodelle für akute hypotensive Episoden basierend auf dem neuen Ansatz zu trainieren. Erweisen sie sich als praxistauglich, könnten sie schon in naher Zukunft das Pflegepersonal auf Intensivstationen unterstützen.

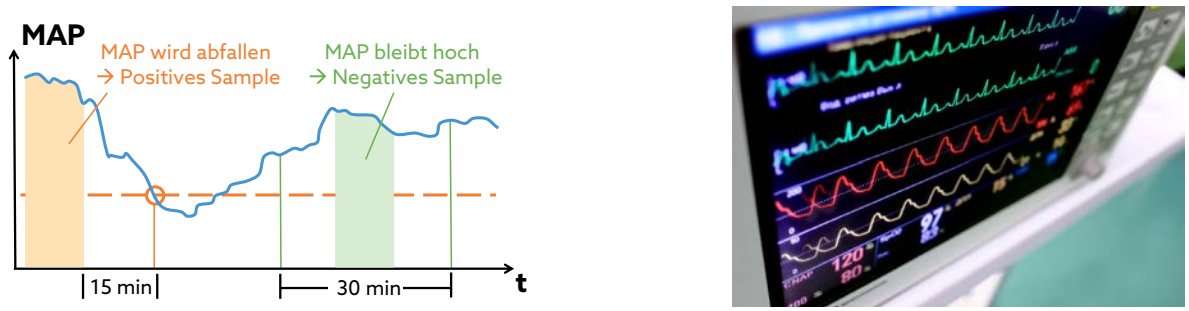



Abbildung 2: Schematische Darstellung der Sample-Auswahl in Abhängigkeit des mittleren arteriellen Blutdrucks (MAP).

## Fazit

Wie immer im maschinellen Lernen ist ein umfassendes Verständnis der Daten und des Anwendungsfalls für die Entwicklung gut funktionierender, praxistauglicher Modelle unumgänglich – nicht nur im medizinischen Umfeld. Oftmals sind sowohl Domänenwissen als auch explorative Datenanalysen (siehe Explorative Datenanalyse mit Zeitreihen) notwendig, um die nötigen Informationen zu extrahieren, mögliche Probleme frühzeitig zu erkennen, und die Modellentwicklung entsprechend anzupassen. Wie erläutert, schließt das insbesondere auch Trainingssamples mit ein, deren richtige Wahl gerade im Fall von Zeitreihendaten eine zentrale Rolle spielt. ◆

# Explorative Datenanalyse mit Zeitreihen

– Dominik Falkner, MSc
Data Scientist in der Unit Logistics Informatics

## Wissensgewinn durch Datensammlung und -auswertung im Zeitverlauf

Data Science ermöglicht es, aus Daten nützliches Wissen zu extrahieren. Dabei sind Daten so vielfältig wie Menschen. Sie sind nicht nur unterschiedlich in ihrer Form – wie nominal, ordinal oder kardinal – um aus ihnen Wissen abzuleiten, ist bei vielen Daten eine Erfassung über eine zeitliche Dauer essenziell. Daher werden Daten oft über einen Zeitverlauf gemessen und es entstehen sogenannten Zeitreihen. Eine Zeitreihe besteht aus einer Reihe von Datenpunkten, welche nach einem Zeitstempel (z.B. der Form 1.1.2021 11:00:01) sortiert sind. Zeitreihen findet man in den verschiedensten Branchen, egal ob in der Industrie (von Fertigungsprozessen abgeleitet), Medizin (EKGs) oder am Finanzmarkt (Börsenkurse). Oft sind Zeitreihen ein zentraler Punkt für Entscheidungen, bringen aber einige Herausforderungen für die Datenanalyse mit sich. Der folgende Artikel zeigt anhand von Beispielen, wie die explorative Datenanalyse mit Zeitreihen gestaltet werden kann.

### Data Science: Schöne neue Welt

Data Science ist zwar bereits eine jahrzehntelang etablierte Wissenschaft, für viele Unternehmen ist sie jedoch noch neu, da sie erst jetzt einen Nutzen für sich daraus erkennen. Der Hype der letzten Jahre um dieses Thema ließ viele Industrie- und Wirtschaftsunternehmen Data Science ohne etablierte Ansätze einführen. Dies führte dazu, dass viele Daten ohne Annotierungen vorliegen und das Wissen darüber auf ausgewählte Expert*innen gestreut ist. Die größte Herausforderung dabei ist zu wissen, wie Daten zustande kommen, wie Systeme reagieren und wie die Daten zu interpretieren sind. Dieses Wissen ist sowohl bei Data Scientists als auch bei Fachexpert*innen des Anwendungsgebietes verteilt und muss durch intensive Zusammenarbeit erst zusammengeführt werden. Der Erfolg von Data-Science-Projekten hängt somit stark von der Kooperation der verschiedenen Wissensträger*innen ab.

### Zeitreihen-Exploration: Was direkt ins Auge fällt

Der Mensch hat eine sehr gute und schnelle visuelle Wahrnehmung, dadurch kann er Zusammenhänge anhand von Bildern schneller verstehen als durch das Lesen der rohen Zahlen und Texte. Visualisierungen sind hervorragend dafür geeignet, dass sich Expert*innen einen umfassenden Überblick über komplexe Daten verschaffen. Diagramme dienen als kritisches Werkzeug zur Erklärung der Dateneigenschaften. Der Fallstrick dabei ist jedoch, dass gut gestaltete Visualisierungen zwar enorm hilfreich sind, naive Versuche aber oft in die Irre führen können und sich schnell als ineffektiv erweisen. Bevor aber mit der Umsetzung von Visualisierungen begonnen werden kann, ist ein anderer Aspekt bei Zeitreihen ausschlaggebend: die sogenannte Samplingrate. Die Samplingrate gibt an, wie oft das Analogsignal über den Messzeitraum abgetastet wird und bestimmt daher, wie genau der Vorgang beobachtet wird. Sie wird bereits beim Erheben von Werten festgelegt. Meist wird sie von den Fachexpert*innen vorgegeben, da die Samplingrate ohne Vorwissen nur schwer bestimmt werden kann. Die Samplingrate beeinflusst auch die Aufzeichnungsdauer und die Datengröße: eine sehr hohe Samplingrate kann zu einem immensen Speicherverbrauch führen, was sowohl die Verarbeitung als auch die Datenhaltung schwieriger gestaltet. Wenn Analysen bereits brauchbare Ergebnisse zeigen, kann es sinnvoll sein, die Samplingrate zu verringern und damit Speicherplatz zu sparen und Berechnungen zu beschleunigen. Für die Weiterverarbeitung durch Algorithmen wird weiters oft vorausgesetzt, dass die Abtastrate zwischen Zeitreihen gleich ist.

Wie in Abbildung 1 und 2 zu sehen ist, gibt es unterschiedliche Arten von Zeitreihen. Die erste Grafik zeigt eine Ableitung eines Elektrokardiogramms (EKG), welches als Entscheidungsgrundlage für Defibrillatoren dient, ob dieser einen elektrischen Impuls senden darf oder nicht. Die zweite Grafik zeigt physikalische Werte, die aus einem Fertigungsprozess extrahiert wurden. Beide Vorgänge treten wiederholt auf und können deshalb verglichen werden. Oft werden diese Vorgänge auch kontinuierlich aufgezeichnet, was zu einer langen kombinierten Zeitreihe führt. In diesem Fall muss diese zuerst in einzelne Sequenzen zerlegt werden, die dann übereinander gelegt werden. An diesem Punkt kann mithilfe der explorativen Datenanalyse ohne großen Aufwand nach Mustern und Gemeinsamkeiten in den Sequenzen gesucht werden. Meist ist es notwendig, die Zeitreihen zu visualisieren. Die einfachste Form dafür ist das überlagernde Liniendiagramm, ein Beispiel dafür zeigt Abbildung 3. Diese Darstellung erlaubt es, schnell mehrere Zeitreihen zu vergleichen und hilft, den Überblick zu bewahren. Je nachdem, wie viele Zeitreihen gleichzeitig angezeigt werden, kann es sinnvoll sein, interessante Datenpunkte (z.B.: die neueste Zeitreihe oder jene mit besonderen Eigenschaften) hervorzuheben. Eine weitere Möglichkeit ist, die Daten selbst zu filtern. Oft existieren Umgebungsparameter, welche

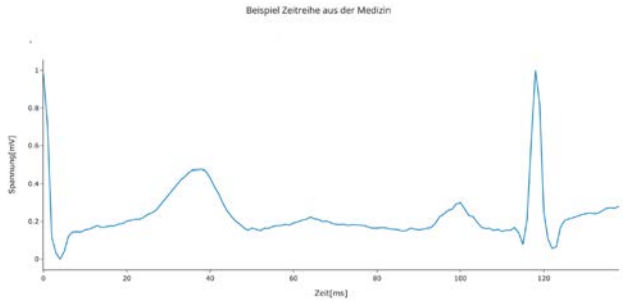



Abbildung 1: Beispiel-Zeitreihe aus dem medizinischen Bereich, die eine Ableitung des Elektrokardiogrammes zeigt.

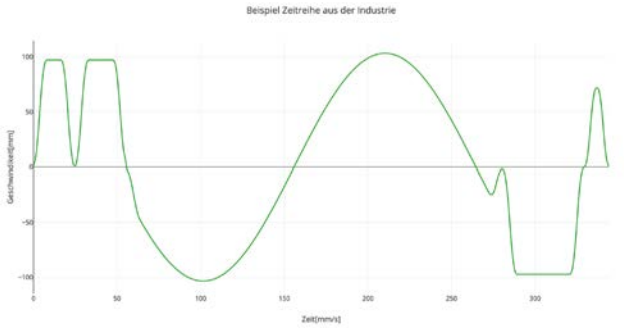



Abbildung 2: Beispiel-Zeitreihe aus der Industrie, die die Geschwindigkeit mit der sich eine Maschine bewegt, zeigt.

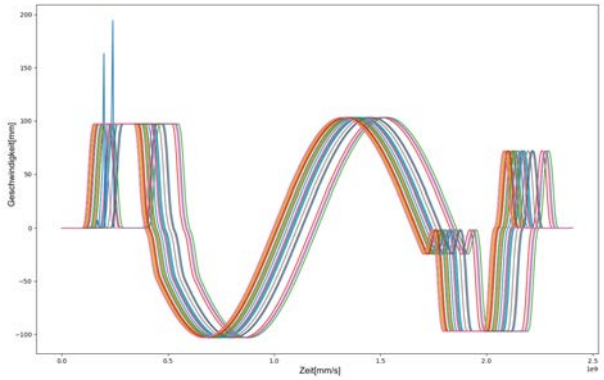
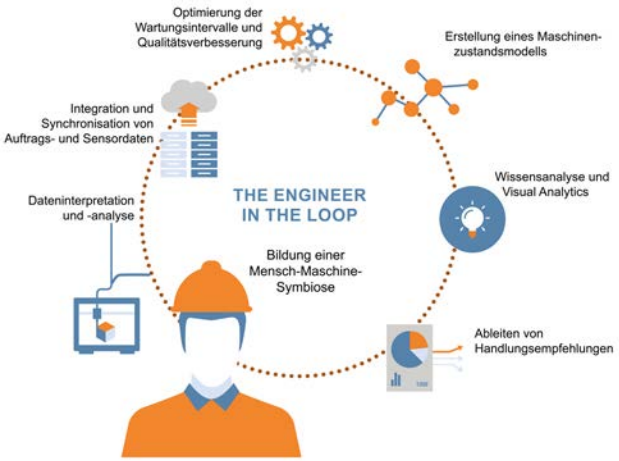



Abbildung 3: Überlagerndes Liniendiagramm mehrerer Zeitreihen, das oft als Ausgangspunkt für erste Analysen dient und einen guten Überblick bietet.




den Vorgang beeinflussen, wie beispielsweise die Außentemperatur. Das macht es leichter, Muster zu identifizieren. Diese Visualisierungen werden sowohl von Data Scientists als auch von Domänen-Expert*innen genutzt. Das Ziel ist dabei eine enge Zusammenarbeit zwischen Domänen-Expert*innen und Data Scientists, damit möglichst viele Erkenntnisse aus den Daten gezogen werden können.

## Expert-in-the-Loop: Zusammenarbeit ist das Wichtigste

Algorithmen allein können keine Erkenntnisse aus Daten gewinnen – noch schwieriger ist es, wenn die Daten nicht annotiert vorliegen. Auch Fachexpert*innen können Probleme oft nicht alleine lösen. Das fehlende Bindeglied dazwischen, um beide Welten miteinander zu verknüpfen, ist das Konzept des „Expert-in-the-Loop". Hierbei werden die vom Algorithmus gefundenen Informationen und das Wissen und die Erfahrung, die Expert*innen über viele Jahre gesammelt haben, kombiniert. Die Hauptidee ist, die Ergebnisse eines Algorithmus zu verbessern bzw. zu kontrollieren. Die Ergebnisse können dann verwendet werden, um die Analysen anzureichern. Die neuen Informationen helfen Expert*innen beim Suchen von Mustern und können Zusammenhänge aufzeigen, welche vorher nur schwer oder gar nicht zu erkennen waren. Zusätzlich können Expert*innen Feedback dazu geben, was der Algorithmus gut erkennt und was nicht. Auf dieser Basis kann der Algorithmus adaptiert werden.

Eine weitere Möglichkeit, Fachpersonal beim Explorieren von Daten zu unterstützen ist, die Bedienbarkeit der Oberfläche möglichst interaktiv zu gestalten. Ist die Visualisierung der Daten nur statisch und können die Fachexpert*innen damit nicht interagieren, sind die Möglichkeiten der visuellen Erfassung sehr begrenzt. Ist es hingegen möglich, Einfluss auf die Darstellungsoptionen zu nehmen, erweitern sich die Perspektiven und damit die Chance, Zusammenhänge zu finden. Die Integration des Domänenwissens funktioniert am besten, wenn die Daten in einer Weise visualisiert werden können, die für Expert*innen interpretierbar ist. Einerseits ist es notwendig, Visualisierungen zu erstellen, welche die verschiedenen Eigenschaften der Daten ausdrücken, und andererseits muss es eine Schnittstelle geben, die einfach zu verstehen ist und es ermöglicht, Domänenwissen einzubringen.

## Fazit

**Zeitreihenanalyse ist eine hervorragende Möglichkeit, um aus Daten im Zeitverlauf Erkenntnisse zu gewinnen. Besonders die Visualisierung unterstützt Menschen dabei, Zusammenhänge und Muster zu erkennen. Dies ist besonders wichtig, damit Fachexpert*innen die Daten mit ihrem Domänenwissen anreichern und so durch ihre aktive Mitarbeit im Data-Science-Prozess einen großen Mehrwert aus den Daten generieren.**

**Eine weitere Möglichkeit, Zeitreihen miteinander zu vergleichen, ist das Zeitreihen-Clustering. Dies stellt eine Methode dar, um Daten automatisiert nach Mustern zu durchsuchen. Lesen Sie mehr dazu in einem unserer nächsten Fachbeiträge. ◆**

# Wir brauchen eine Mobilitätsrevolution!

– Sabrina Wagner, BSc
Software Developer in der Unit Logistics Informatics

## Klimafreundliche Alternativen für die Mobilität der Zukunft

Die selbstbestimmte Mobilität ist ein integraler Bestandteil des alltäglichen Lebens und ein Komfort, auf den man heutzutage kaum mehr verzichten will. Doch der immer weiter zunehmende Verkehr ist verantwortlich für 30 % der österreichischen CO2-Emissionen [1]. Eine Umschichtung vom ungeteilten, motorisierten Individualverkehr zu klimafreundlichen Alternativen ist dringend notwendig. Mit welchen (Verkehrs-)Mitteln schaffen wir es, gemeinsam die Mobilitätsrevolution einzuleiten und die Mobilitätsgewohnheiten zu umweltfreundlichen Alternativen hin zu wandeln?

### Das Problem: $CO_2$ und Ineffizienz

Allein die über 5 Mio. PKW in Österreich – das entspricht 0.6 PKW pro Einwohner*in unabhängig von Alter, Fahrfähigkeit oder Führerscheinbesitz – stoßen 17,13 % der österreichischen $CO_2$-Emissionen aus [6] [2] [3]. Doch zusätzlich zu den immer mehr werdenden PKW in Österreich nimmt auch der Besetzungsgrad immer weiter ab: 2017 lag die durchschnittliche Anzahl von Insassen in einem Fahrzeug bei 1,15, das heißt, dass Fahrzeuge immer weniger Menschen befördern – meist sogar nur einen Menschen – und trotzdem Platz und Ressourcen wegnehmen [4].

### Öffentliche Verkehrsmittel

Die klassische Alternative zum eigenen PKW sind die öffentlichen Verkehrsmittel (ÖV) wie Bus oder Bahn. Speziell in den letzten Jahren wurde das öffentliche Verkehrsnetz stark ausgebaut, um die Attraktivität dieser nachhaltigen Fortbewegungsart zu steigern. Zusätzlich werden verschiedene Tarife für Pendler*innen bzw. Vielfahrer*innen angeboten, wie Jahreskarten und Semestertickets. Da der ÖV in Ballungsräumen bereits sehr flexibel nutzbar und finanziell sehr attraktiv ist, stellt er eine tatsächliche Alternative zum ungeteilten Individualverkehr dar. In ruralen Regionen ist vielerorts das Angebot noch nicht weit genug ausgebaut, um die Anforderungen eines flexiblen und komfortablen öffentlichen Nahverkehrs abzudecken. Dieses flexible Mobilitätsbedürfnis wird durch aktuelle Entwicklungen wie Arbeitszeitflexibilisierung und Home-Office noch verstärkt und oft durch den vor allem auf den Schulverkehr ausgelegten ÖV in ländlichen Gebieten nicht für alle Interessensgruppen abgedeckt.

### Mitfahrbörsen

Mitfahrbörsen sowie Mitfahrgelegenheiten im Allgemeinen sind eine Option, um kostengünstig und schnell die alltäglichen und wiederkehrende Strecken zurückzulegen. Um den Weg von und zur Arbeit mit den Kolleg*innen gemeinsam zurückzulegen, bieten unterschiedliche Anbieter Mitfahrgelegenheiten im Rahmen des Pendelverkehrs an. Diese regelmäßigen Mitfahrten decken nur einen Teil des Mobilitätsbedürfnisses ab, können aber zusätzlich zu den finanziellen und ökologischen Anreizen auch die Notwendigkeit eines Zweitfahrzeugs im Haushalt reduzieren. Auch für Unternehmen bieten betriebliche Mitfahrangebote soziale und finanzielle Vorteile, wie beispielsweise erhöhten Zusammenhalt und reduzierte Kosten für Parkmöglichkeiten.

Geprägt durch das digitale Zeitalter steht ein breites Angebot an Mitfahrbörsen und -Apps am Markt zur Verfügung, wie beispielsweise Carployee, Foahstmit, Ummadum oder Hey Way. Diese (lokale) Fragmentierung der unterschiedlichen Lösungen führt zu abgeschotteten Gruppen von Nutzer*innen und in der Konsequenz auch zu einer geringeren Anzahl von angebotenen Strecken. Weitere Anbieter wie BlaBlaCar mit ihrem Fokus auf Langstrecken, können bei nicht durch den öffentlichen Verkehr abgedeckten Strecken eine Ergänzung darstellen.

Zusätzlich zu den digitalen Mitfahrbörsen gibt es auch Angebote zum analogen und spontanen Mitfahren, wie das angebotene „Mitfahrbankerl" des Energiebezirkes Freistadt. Sie sollen das klassische Autostoppen ablösen und eine flexible Möglichkeit des Mitfahrens bieten.

### Car-Sharing

Fast ausschließlich werden bei der Frage der Umweltschädlichkeit im Bereich der Mobilität der Spritverbrauch und die dadurch verursachten Emissionen betrachtet. Jedoch gibt es neben der Problematik des niedrigen Besetzungsgrades und der dadurch verbundenen hohen Klimabelastung noch einen weiteren Faktor: Die Fahrzeugherstellung selbst. Ein Fahrzeug verursacht Treibhausgas-Emissionen im Ausmaß von mehreren zehntausend gefahrenen Kilometern bereits bei der Herstellung, ohne jemals einen Kilometer gefahren zu sein [7].

Car-Sharing bietet eine gute Alternative zum Kauf eines eigenen Fahrzeugs, indem ein PKW flexibel für die benötigte Dauer angemietet werden kann. Car-Sharing ist eine optimale Ergänzung für den öffentlichen Verkehr, da bei Notwendigkeit eines individuellen Fahrzeugs (z.B. zur Anreise zu nicht öffentlich zugänglichen Freizeitangeboten) zeitlich flexibel auch unterschiedliche Fahrzeugtypen angemietet werden können.



[1] Umweltbundesamt, 2018a
[2] Umweltbundesamt, 2018b
[3] Statistik Austria, 2018a
[4] Trafico, Umweltbundesamt, VCÖ 2018
[5] Greenpeace, Städtische Mobilität nach Corona: Auto-Kollaps oder Fahrrad-Boom, 2020

Auch bei den Anbietern von Car-Sharing kommt es zu einer starken Fragmentierung der Anbieter und durch die unterschiedlichen Zahlungsmodalitäten und Verfügbarkeiten auch zu einer unübersichtlichen Informationslage für Kund*innen. Es gibt sowohl private, wie beispielsweise auf der Plattform Getaround, als auch öffentliche Anbieter, wie beispielsweise ÖBB Rail&Drive, TIM oder MühlFerdl. Diese verfügen auch, unterschiedliche Zielsetzungen wie z.B. lokale Ergänzung zum ÖV und/oder Ersatz des Zweitwagens oder Tourismusfokussierte Angebote als Ergänzung zur öffentlichen Anreise. Grundsätzlich werden zwei Car-Sharing-Typen unterschieden: das stationsbasierte und das Free-Floating-Car-Sharing. Der Unterschied ist, dass beim stationsbasierten Car-Sharing mehrere fixe Stationen vom Anbieter definiert werden, an denen die Fahrzeuge sowohl abgeholt als auch abgestellt werden können, wobei beim Free-Floating-Car-Sharing das Fahrzeug an einem beliebigen Ort innerhalb einer Zone abgestellt werden kann. Am weitesten verbreitet ist der stationsbasierte Ansatz nur in manchen Ballungsräumen wird Free-Floating-Car-Sharing angeboten.

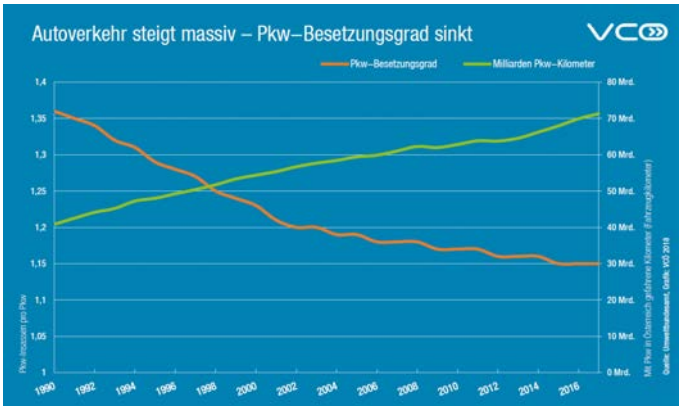
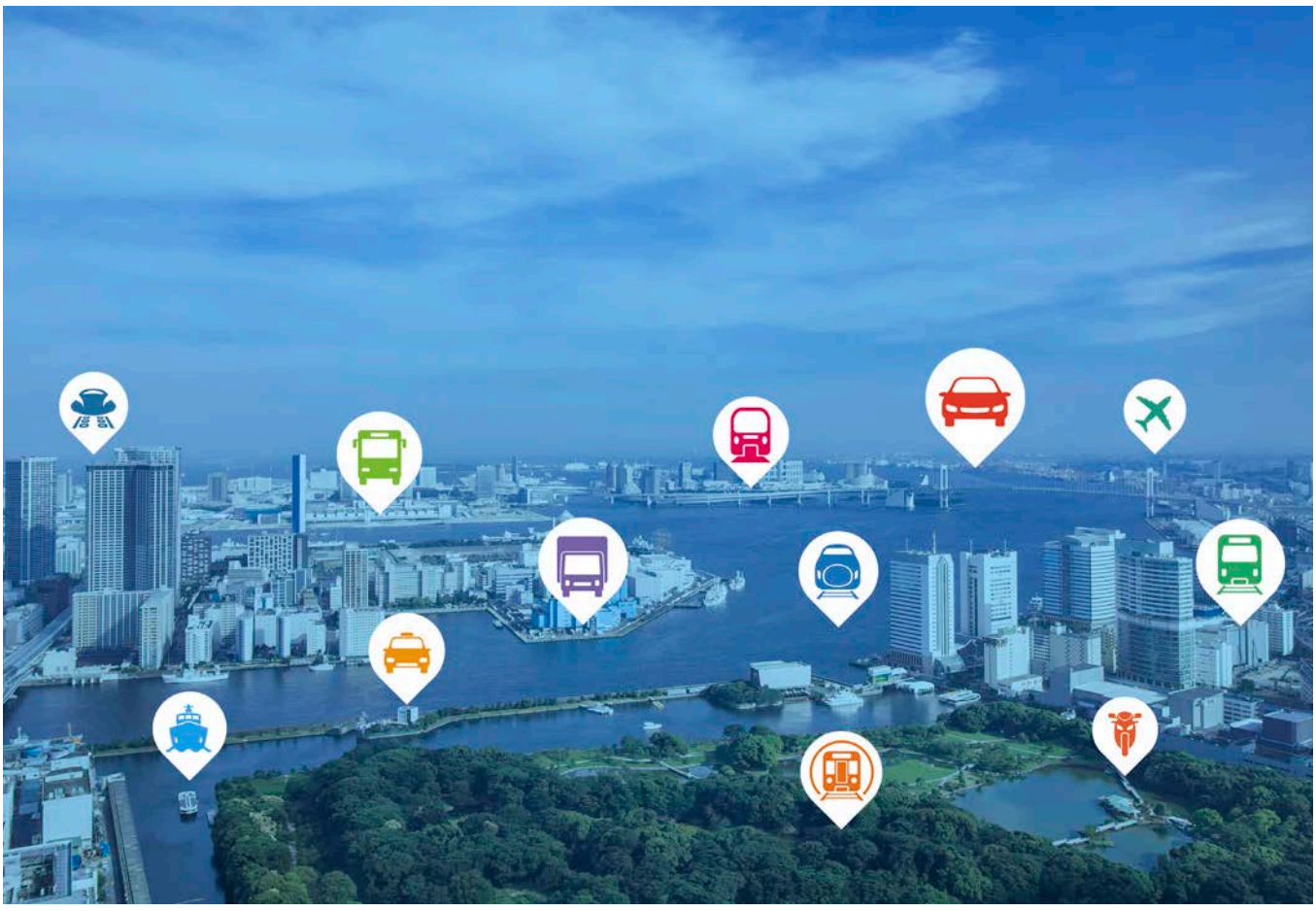



Abbildung (li) : Besetzungsgrad der Autos im Zeitverlauf und Anzahl der gefahrenen Kilometer



## Beispiele für Forschungsprojekte zu zukunftsfähiger Mobilität

Die RISC Software GmbH befasst sich stark mit einer zukunftsfähigen und nachhaltigen Mobilität und arbeitet in mehreren Projekten mit dem Ziel, nachhaltige, günstige und unkomplizierte Lösungen für die aktuellen Herausforderungen im Individualverkehr zu entwickeln.

### EVIS.AT

Das Forschungsprojekt EVIS.AT (Echtzeit Verkehrsinformation Straße Österreich) und die daraus gewonnenen Verkehrsdaten dienen als Basis für die weiteren genannten Projekte. In EVIS.AT wurden mehrere Flotten (LKW, PKW, eine Kombination aus LKW und PKW, Einsatzfahrzeug, Taxi) für eine Datensammlung akquiriert sowie Streckenabschnitte mit Sensorik (VDL/Zählschleifen, Bluetooth) für die Verkehrszählung ausgestattet. Ziel dieses Projektes ist es, Echtzeitverkehrslageinformationen für den Raum Oberösterreich zur Verfügung zu stellen. Diese Verkehrslage ist auch ein integraler Bestandteil der österreichweiten Verkehrslage der VAO.

### LisiGo

Die App LisiGo ist das Stau- und Nachrichtenportal der OÖNachrichten. Sie dient der Stauvermeidung auf dem Weg von und zur Arbeit, indem auf Basis der aktuellen Verkehrslage eine optimale Route für eine benutzerdefinierte Strecke berechnet wird. Zusätzlich kann auch eine Prognose für eine Fahrt für die nächsten 30 Minuten berechnet werden, um die Verkehrslage zu vergleichen und den optimalen Zeitpunkt für die Fahrt zu finden. LisiGo trägt somit zur Reduzierung von Staus bzw. Stop-and-Go-Verkehr und in weiterer Folge zur Reduzierung der CO2-Emissionen bei.

### DOMINO

Durch das Leitprojekt DOMINO (Drehscheibe für intermodale Mobilitätsservices und -technologien) soll die Gestaltung eines nachhaltigen, durchgängigen und öffentlich zugänglichen Mobilitätsmanagements „Mobility as a Service“ (MaaS) in Österreich ermöglicht werden. Dabei werden drei unterschiedliche Pilotregionen (OÖ; NÖ, Salzburg) aufgesetzt, um neue Mobilitätsservices zu entwickeln, zu erproben und schlussendlich zu integrieren. Die Laborumgebung in der Pilotregion OÖ dient als Basis für die Optimierung der täglichen Pendlerverkehre. Diese Optimierung wird durch die Erhöhung der Besetzungsgrade der Fahrzeuge (durch eine Mitfahrbörse), die Förderung auf den Umstieg auf öffentliche Verkehrsmittel, die Reduktion von Stausituationen (CO2-Ausstoß) im Großraum Linz und die Attraktivierung von Arbeitgeber*innen im Großraum Linz durch ein verbessertes Mobilitätsangebot erreicht. Die RISC Software GmbH entwickelt im Zuge des Projekts einen Ride-Sharing-Pool, der zum Ziel hat, die unterschiedlichen Mitfahrlösungen zu verbinden, um Mitfahren als integralen Bestandteil im Mobilitätsmix zu etablieren. ◆



*[6] KFZ Bestand – statistik.at*

*[7] VOL (Vorarlberg Online)*

# Data Engineering – Die solide Basis für eine effektive Datennutzung

– DI Paul Heinzlreiter
Senior Data Engineer in der Unit Logistics Informatics

## Der Weg der Daten von den Quellen zum integrierten Data Lake.

Data Engineering integriert Daten aus verschiedensten Quellen und macht sie effektiv nutzbar. Damit stellt es vor allem im Big Data Bereich eine Voraussetzung für effektive Datenanalyse, Machine Learning und Künstliche Intelligenz dar.

In den letzten Jahren hat das Thema der Informationsgewinnung aus großen Datenmengen für immer mehr Betriebe in den verschiedensten Wirtschaftssektoren stark an Bedeutung gewonnen. Beispiele hierfür sind historische Verkaufsdaten, die zur Optimierung des Produktangebots von Online-Shops verwendet werden können, und Sensordaten von einer Produktionslinie, die helfen können, die Qualität der Produkte zu steigern oder im Rahmen der vorbeugenden Wartung Maschinenteile rechtzeitig auszutauschen. Neben dem direkten Einsatz einer integrierten Datenbasis in der betrieblichen Praxis stellt gerade die Aktualität der Themen Künstliche Intelligenz (KI) und Machine Learning (ML) mit dem Versprechen, beispielsweise einen Produktionsprozess laufend optimieren zu können, eine starke Motivation dar.

Wenn der Prozess der Informationsgewinnung allerdings in seiner Gesamtheit betrachtet wird, wird schnell klar, dass KI und ML nur die sprichwörtliche Spitze des Eisbergs darstellen. Diese Methoden benötigen gerade für die Schritte des Modelltrainings und der Modellvalidierung große Mengen an konsistenten und vollständigen Datensätzen. Solche Datenmengen können beispielsweise von Sensornetzwerken oder von Sensoren in der Produktion erzeugt werden. Die Übernahme, Speicherung und Aufbereitung dieser Daten, um sie effektiv nutzbar zu machen, ist die zentrale Aufgabe von Data Engineering.

Dies ist unabhängig davon, ob das Ziel Firmeneinheitliches und effektives Reporting, Data Science zur Verbesserung des Produktionsprozesses oder KI heißt. Eine solide Datenbasis ist in allen Fällen notwendig. Die Integration von Daten in eine gemeinsame Datenbasis kann zusätzlich eine verlässliche Ground-Truth für verschiedenste Anwendungsfälle im Unternehmen bilden: Für ein effektives Tagesgeschäft, für strategische Planung basierend auf soliden Daten und Fakten oder für das Modelltraining im Bereich KI.

### Abgrenzung

Das übergreifende Ziel ist somit, die Qualität und Nutzbarkeit der zur Verfügung stehenden Daten zu erhöhen und folgt damit im Wesentlichen der Data-Science Hierarchy-of-Needs [1], welche die Stufen von den Rohdaten bis zur KI beschreibt. Analog zur Bedürfnishierarchie nach Maslow [2] stellen die unteren Ebenen der Pyramide eine notwendige Voraussetzung für die darauf aufbauenden Schritte dar.

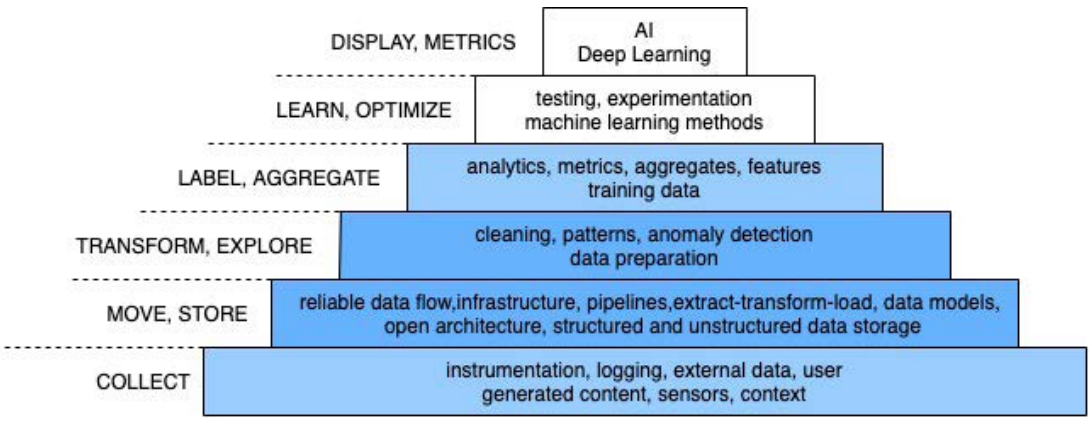



Abbildung 1: Data-Science Hierarchy-of-Needs [1]



*[1] Blasch, Erik & Sung, James & Nguyen, Tao & Daniel, Chandra & Mason, Alisa. (2019). Artificial Intelligence Strategies for National Security and Safety Standards.*
*[2] Abraham Maslow: A Theory of Human Motivation. In Psychological Review. 1943, Vol. 50 #4, Seite 370–396.*

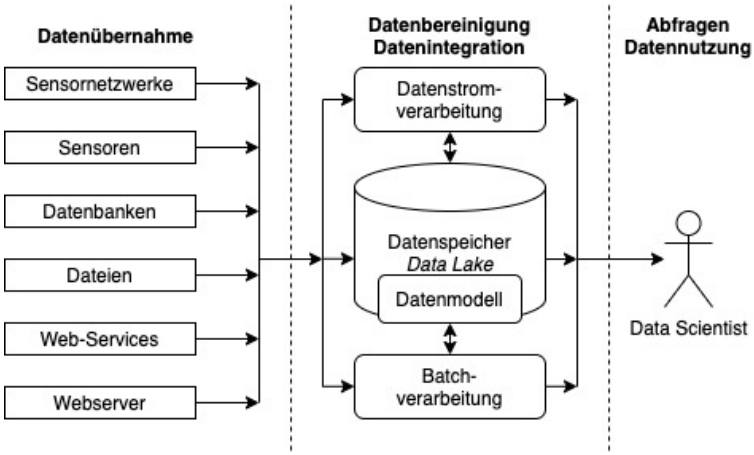



Abbildung 2: Datenbereinigung und -integration

An der Spitze der Pyramide stehen die Tätigkeiten der Data Science, die auf integrierte und bereinigte Datenbestände aufsetzen. Mit diesen können dann beispielsweise ML-Modelle trainiert werden. Die blau eingefärbten Ebenen stellen die Data-Engineering-Tätigkeiten dar, wobei der Schwerpunkt auf den Ebenen move, store sowie transform, explore liegt. Während die darüberliegenden Ebenen mit KI, Deep Learning, und ML die Domäne von Data Scientists sind, sind Tätigkeiten wie Data Labeling und Data Aggregation Grenzbereiche, die je nach genauer Aufgabe und Personalverfügbarkeit von Data Scientists oder Data Engineers durchgeführt werden können.

Die Tätigkeiten des Datensammelns an der Basis der Pyramide fallen insofern nur teilweise in den Bereich des Data Engineerings, als das dieses die Daten üblicherweise an einer definierten Schnittstelle – über Dateien, externe Datenbanken oder ein Netzwerkprotokoll – übernimmt. Dies liegt auch darin begründet, dass Data-Engineering einen Teilbereich der Informatik bzw. des Software-Engineerings darstellt, und damit üblicherweise nicht mit Themen wie Aufbau oder Betrieb von Datenerfassungshardware wie beispielsweise Sensoren befasst ist.

## Datenbereinigung und -integration

Im Rahmen des Data Engineering Prozesses werden die Rohdaten nach der Übernahme über mehrere Schritte aufbereitet und letztendlich konsistent und vollständig aufbereitet im Datenspeicher abgelegt:

1. Datenbereinigung
2. Datenintegration
3. Umformung der Daten

Diese Formen von Datentransformationen werden schrittweise und aufeinander aufbauend durchgeführt. Die technische Umsetzung kann durch Datenstromverarbeitung – die konsekutive Verarbeitung vieler kleiner Datenpakete – oder als Batchverarbeitung für den gesamten Datensatz gleichzeitig erfolgen. Ein entsprechend dimensionierter Datenspeicher – der Data Lake – ermöglicht es dabei, die Daten in verschiedenen Zuständen während ihrer Verarbeitung zu persistieren.

Die Bereinigung von Daten umfasst beispielsweise eine Überprüfung der eingelesenen Datenzeilen auf Vollständigkeit und syntaktische Korrektheit. Auch Datenfehler wie falsche Sensorwerte können durch vorgegebene Regeln in diesem Schritt festgestellt werden.

Wenn diese Kriterien verletzt werden, gibt es je nach Anwendungsfall folgende Möglichkeiten:

- Rohdatenqualität verbessern: Falls die Rohdaten in verbesserter Qualität nachgeliefert werden können, ersetzen diese die fehlerhaften Daten
- Daten verwerfen: Fehlerhafte Daten können beispielsweise verworfen werden, wenn der Datensatz für Trainingszwecke im ML eingesetzt werden soll, und ausreichend korrekte Daten zur Verfügung stehen.
- Fehler während des Imports automatisiert korrigieren: Wenn beispielsweise die Daten aus einer zusätzlichen Datenquelle bezogen werden können, können Fehler im Rahmen der Datenintegration behoben werden.

In der Praxis ist das Verwerfen der fehlerhaften Daten die am einfachsten umzusetzende Lösung. Wenn allerdings jeder einzelne Datenpunkt für die geplanten Auswertungen Relevanz haben kann, müssen fehlerhafte Daten nach Möglichkeit korrigiert werden. Dieser Fall kann zum Beispiel bei der Qualitätsbeurteilung in der Produktion auftreten, wenn die Produktionsdaten für ein fehlerhaftes Werkstück aufgrund eines Sensorfehlers nicht korrekt sind. Die Korrektur von Daten kann entweder manuell durch Domänenexpert*innen erfolgen, oder die korrekten Daten werden zu einem späteren Zeitpunkt nachgeliefert.

Die Einbindung von Domain-Expert*innen ist hier zentral, weil diese einerseits die Kriterien für die Korrektheit der Daten wie beispielsweise Sensorwerte kennen, und andererseits wissen, wie mit fehlerhaften oder unvollständigen Daten umgegangen werden soll.

Die Integration von Daten befasst sich mit der automatisierten Verknüpfung von Daten aus verschiedenen Datenquellen. Je nach Anwendungsdomänen und Art der Daten kann die Datenverknüpfung durch verschiedene Methoden erfolgen wie zum Beispiel:

- Eindeutige Bezeichner, analog zu Fremdschlüsseln im relationalen Modell
- Geographische oder zeitliche Nähe
- Domänenspezifische Zusammenhänge wie Abfolgen in Fertigungsprozessen oder in Fertigungsstraßen

Nach den Schritten der Datenbereinigung und der Datenintegration können Data Engineers einen für die Weiterverwendung durch Data Scientists geeigneten Datenbestand zur Verfügung stellen. Der oben genannte Schritt der Datenumformung bezieht sich auf laufende Anpassungen des Datenmodells, um die Performanz von Abfragen durch die Data Scientists zu verbessern.

## Datenspeicherung und Datenmodellierung für Big Data

Der bereinigte und integrierte Datenbestand kann in einer geeigneten Datenhaltungslösung abgelegt werden. In Anwendungsbereich von Industrie 4.0 werden beispielsweise durch Sensoren laufend Daten generiert, was oft innerhalb von Monaten zu Datenmengen im Terabyte-Bereich führt. Solche Datenmengen sind oft mit einer klassischen relationalen Datenbank nicht mehr handhabbar. Es gibt zwar am Markt verfügbare skalierbare Datenbanken, die das relationale Modell einsetzen, diese kommen aber für viele Umsetzungsprojekte – gerade im KMU-Bereich – aufgrund ihrer hohen Lizenzkosten nicht infrage.

Als Alternative dazu bieten sich horizontal skalierbare NoSQL-Systeme an, wobei der Begriff eine Abkürzung für „Not only SQL" darstellt. Unter diesem Begriff werden Datenspeicher zusammengefasst, die nicht-relationale Datenmodelle verwenden. Die Eigenschaft der horizontalen Skalierbarkeit bezeichnet die Möglichkeit, solche Systeme durch Integration zusätzlicher Hardware für prinzipiell unbeschränkte Datenmengen zu erweitern. Typische Vertreter von NoSQL-Systemen unterliegen zudem oft liberalen Lizenzmodellen wie beispielsweise der Apache-Lizenz und können so lizenzkostenfrei auch kommerziell eingesetzt werden. Zudem stellen diese Systeme keine speziellen Anforderungen an die verwendete Hardware, was die Anschaffungskosten solcher Systeme weiter reduziert. Somit stellen NoSQL-Systeme wie beispielsweise Apache Hadoop und verwandte Technologien eine kostengünstige Möglichkeit dar, Abfragen auf Datenmengen im Terabyte-Bereich auszuführen.

Gerade im Big Data-Bereich kommt der Auswahl einer geeigneten NoSQL-Datenbank sowie eines passenden Datenmodells zentrale Bedeutung zu, weil beide Aspekte zentral für die Performance des Gesamtsystems sind. Dies bezieht sich sowohl auf die Einbringung von Daten als auch auf Abfragen gegen das NoSQL-System.

Die Auswahl der einzusetzenden Technologie sowie das Datenmodelldesign ist ganz klar von den Anforderungen an das System getrieben:

- Welche Datenmengen und Datenraten müssen importiert werden?
- Welche Abfragen und Auswertungen sollen mit den Daten durchgeführt werden?
- Wie sind die Anforderungen an die Performanz der Abfragen? Handelt es sich um ein Echtzeitsystem?

Zentral ist beispielsweise die Frage, ob das System nur fix vorgegebene Abfragen unterstützen soll, oder – zum Beispiel unter Einsatz von SQL – flexible Abfragen ermöglichen soll.

Im Rahmen der Technologieauswahl kann man beispielsweise unterscheiden, ob der Zugriff auf die Daten immer über einen bekannten Schlüssel erfolgt, oder auch Abfragen auf die Werte anderer Attribute erfolgen. Im ersten Fall eignet sich ein System mit der Semantik einer verteilten Hash-Map, wie zum Beispiel Apache HBase, während sich im anderen Fall beispielsweise eine in-Memory Analyselösung wie Apache Spark anbietet. Falls die Nutzung der Daten primär auf die Verknüpfungen zwischen Daten abzielt, ist der Einsatz einer Graphdatenbank zu überlegen.

In einem Big-Data-System werden die Daten aus Performance-Gründen denormalisiert gespeichert, d.h. alle für ein Abfrageergebnis relevanten Daten sollten gemeinsam gespeichert werden. Der Grund hierfür ist, dass die Durchführung von Joins sehr ressourcenintensiv und zeitaufwendig ist. Für den Entwurf des Datenmodells sind die geplanten Abfragen daher zentral. Beispielsweise sollten die Attribute, die hauptsächlich als Parameter in den Abfragen auftreten, als Schlüsselattribute eingesetzt werden. Dies ist auch der Grund dafür, warum das Datenmodell oft beim Hinzukommen neuer Abfragen erweitert werden muss, um deren effektive Durchführung zu gewährleisten, und damit auch nach der Dateneinbringung laufend Data-Engineering-Tätigkeiten anfallen. ◆

**Use Case 1: Firmeninterne Datenintegration**

Daten aus verschiedenen Quellen können zusammengeführt und in einem integriertem Datenmodell effektiv verwendet werden

**Use Case 2: Datenaufbereitung für KI / ML**

Data-Engineering-Methoden können dazu dienen, eine große Menge an konsistenten und vollständigen Trainingsdaten für KI und ML zur Verfügung zu stellen

**Use Case 3: Transformation des Datenmodells zur Verbesserung des Datenverständnisses**

Data Engineering kann das Datenverständnis signifikant erhöhen, indem das Datenmodell dem Anwendungsfall besser angepasst wird. Ein Beispiel könnte die Einführung einer Graphdatenbank sein.

**Use Case 4: Verbesserte (schnellere) Datennutzung**

Data Engineering kann durch die Anpassung der Datenspeicherung und des Datenmodells zu einer erheblichen Beschleunigung von interaktiven Abfragen beitragen.

# Agile & Test-Driven: Der Kunde im Mittelpunkt

– Yvonne Marneth, BSc
Software Developer in der Unit Domain-specific Applications

## Wie Test-Driven-Development sowohl für Kund*innen als auch für das Entwicklerteam einen Vorteil bringt

Mit agilen Methoden werden Kund*innen aktiv in den Entwicklungsprozess einbezogen. In regelmäßigen Abständen bekommen sie die Möglichkeit, mit ihrem Feedback den Fortschritt und die Richtung ihres Produkts zu überprüfen. Auf diese Weise entsteht ein wertvolles Produkt, das schnell ausgeliefert werden kann. Mit dem Einsatz von Test-Driven-Development wird dieses Kernkonzept auch in die technische Entwicklung übertragen.

### Was ist Test-Driven-Development?

Test-Driven-Development (TDD) ist ein Vorgehensmodell in der Softwareentwicklung, das darauf aufbaut, Software über diverse Testmethoden automatisiert zu testen. Traditionell wird ein Feature zuerst implementiert und dann getestet. Test-Driven-Development kehrt diesen Prozess um: Hier wird für ein neues Feature zuerst ein Test geschrieben und dann die zugehörige Logik schrittweise implementiert, bis der Test erfolgreich abläuft. Anschließend wird der Code überarbeitet, um ihn zu verbessern und verständlicher zu machen. Zu guter Letzt wird er in die Codebasis integriert und dort mit allen vorhandenen Tests überprüft. Erst dann gilt das Feature als erfolgreich implementiert.

Diese Vorgehensweise kann zunächst sehr unintuitiv und zeitintensiv wirken. Tatsächlich bringt sie aber – vor allem in einen agilen Arbeitsmodus integriert – viele Vorteile mit sich und kann den Entwicklungsprozess sogar nachhaltig effektiver gestalten.

### Qualitätssteigerung

Die wahrscheinlich naheliegendste Folge des Test-Driven-Developments ist die höhere Code-Qualität, die aus den getroffenen Maßnahmen entsteht. Über diverse Qualitätsparameter wie die Anzahl von „Code Smells" (Unschönheiten im Code), Bugs, etc. wird diese Steigerung messbar. Diese Parameter werden auch unter dem Begriff „Technische Schuld" zusammengefasst. Auch diese Messwerte können automatisiert ermittelt werden und geben einen Überblick über den allgemeinen Zustand eines Softwareprodukts. Das Ziel von TDD ist hier, die technische Schuld im Blick zu behalten und möglichst gering zu halten. Das führt dazu, dass die Kosten für Veränderungen und Erweiterungen später wesentlich niedriger ausfallen.

Durch die eingezogene Feedbackschleife kann sichergestellt werden, dass sich der Code richtig verhält, es kann aber auch überprüft werden, dass ein unerwünschtes Verhalten nicht auftritt. Entwickler*innen beschäftigen sich also nicht nur mit dem Optimalfall, sondern auch mit möglichen Fallstricken, welche sonst erst in tatsächlichen Nutzertests auffallen würden. Die aufwendige und zeitintensive Testphase mit Benutzer*innen kann somit wesentlich verkürzt werden, Entwickelnde behandeln die Fälle direkt und versuchen nicht, zuerst ein gemeldetes Fehlerszenario nachzustellen und zu verstehen. Außerdem bearbeiten sie Fehler, solange der Code noch „frisch" im Kopf ist und müssen sich nicht erst wieder hineindenken.

Code Reviews sind dabei in der Überarbeitungsphase eine gute Möglichkeit, Kolleg*innen einen Einblick in den geschriebenen Code zu geben. Einerseits wird dadurch vermieden, dass sich sogenannte „Wissenssilos" bilden. Dabei besitzt nur ein einzelnes Teammitglied Wissen über eine bestimmte Codekomponente. Diese Abhängigkeit möchte man vermeiden. Andererseits steigt die Codequalität, da konstruktives Feedback eingebracht wird. Code Reviews können auch dazu dienen, neue Mitarbeiter*innen in ein Projekt einzuführen oder insgesamt Wissen im Team zu verteilen. Durch den Prozess kann so auch die Arbeitsteilung und Teamdynamik langfristig verbessert werden.

Falls Anwender*innen trotzdem noch weitere Fehler finden, unterstützen vorhandene Tests beim Lesen des Codes und somit bei einer schnellen Behebung des Fehlers. Durch neue Tests wird dann sofort sichergestellt, dass ein konkreter Fehler sich in der Zukunft nicht erneut einschleichen kann.

Kund*innen haben dadurch den Vorteil, dass die Lebenserwartung ihrer Software erhöht wird und bei Benutzer*innen weniger Fehler auftreten. Damit werden die Kosten für Support verringert. Die höhere Qualität ist allerdings nur ein Teilziel dieses Prozesses.

### Test-Driven-Development im agilen Kontext

In agilen Vorgehensmodellen wie Scrum oder Kanban wird im Gegensatz zu traditionellen Methoden ein iterativer Entwicklungszyklus abgebildet. Dieser ist darauf ausgelegt, Kund*innen möglichst schnell ein verwendbares Produkt zu liefern und möglichst

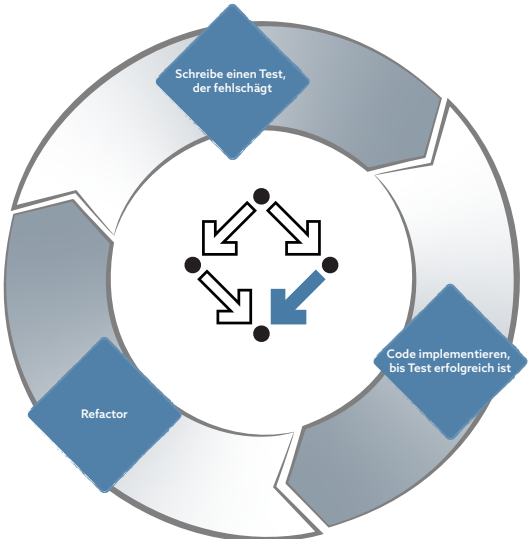



Abbildung 1: Test-Driven-Development beschreibt einen zyklischen Entwicklungsprozess, bei dem die Tests für Code erstellt werden, noch bevor dieser implementiert ist.

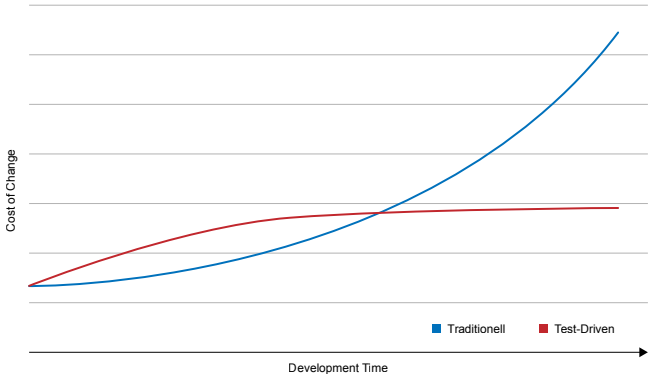



Abbildung 2: Das Ziel von Test-Driven-Development ist, die Cost of Change im Vergleich zu einem traditionellen Workflow nachhaltig zu verringern.

früh Feedback einzuholen. Auf diese Weise kann im weiteren Entwicklungsprozess direkt auf Probleme eingegangen und die Richtung korrigiert werden. Test-Driven-Development überträgt diesen zyklischen Arbeitsablauf auf die technische Entwicklungsebene. Der Zyklus besteht, wie in Abbildung 1 gezeigt aus dem kontinuierlichen Schreiben von Tests, implementieren von Funktionen und Überarbeiten des geschriebenen Codes. Das zyklische Arbeiten reduziert punktuell die Komplexität einer Aufgabe und hilft, den Fokus zu halten.

In agilen Prozessen können sich Anforderungen durch Kund*innen immer wieder verändern. Daher muss schon bei der Entwicklung auf Flexibilität und Änderungsfreundlichkeit in der Codebasis geachtet werden. Das ist wiederum nur möglich, wenn die Codebasis schlank und zielgerichtet ist, was durch ständiges Überarbeiten sichergestellt wird. Hohe technische Schuld führt dazu, dass Änderungen zu Regressionen führen können, welche teils zeitaufwendig und teuer zu beheben sind oder sogar erst auf dem Produktivsystem auffallen und somit zusätzliche Supportkosten verursachen. Das Ziel von TDD ist in Abb. 2 gezeigt.

Testabdeckung, erfolgreiche Testdurchläufe und Reviews können in einem agilen Umfeld helfen, eine aussagekräftige „Definition of Done" zu formulieren und deren Erfüllung objektiv zu verifizieren. Eine gute „Definition of Done" ist notwendig, um klare Erwartungen für die Fertigstellung einer Funktion zwischen Entwickler*innenteam und Kund*innen zu schaffen. Sie beschränkt sich für gewöhnlich nicht nur auf die reine Entwicklungszeit, die ein*e Entwickler*in braucht, bis ein Feature sichtbar ist, sondern bis es alle gemeinsam vereinbarten Anforderungen erfüllt. Diese können funktionaler, qualitativer und sicherheitstechnischer Natur sein. Wenn hier kein gemeinsames Verständnis zwischen Entwickler*innenteam und Kund*innen herrscht, müssen eigentlich abgeschlossene Funktionen oft nachgearbeitet werden. Das kann den Arbeitsfluss stören und führt letztendlich dazu, dass Funktionen sehr lange in der Umsetzung brauchen.

Test-Driven Development baut auf agilen Werten auf und hilft, das volle Potenzial dieser Arbeitsweise auszunutzen. Initial kann sich die Cycle-Time in einem Team erhöhen, während die Entwickler*innen lernen, die neuen Methoden einzusetzen. Mit der Zeit, wenn aus neuen Konzepten Routine wird und Erfahrung aufgebaut wurde, wird der Aufwand vernachlässigbar und zahlt sich mit einer niedrigeren „Cost of Change" aus. Die kontrollierte Erhöhung der Softwarequalität ist letztendlich die einzige zuverlässige Methode, um den Entwicklungsprozess langfristig zu beschleunigen.

## Fazit

Agile Workflows und Test-Driven-Development sind jeweils darauf ausgelegt, schnell Feedback einzuholen und darauf zu reagieren. Agile Vorgehensmodelle setzen hier vor allem auf der organisatorischen Seite an. Damit das technisch auch reibungslos funktionieren kann, führt Test-Driven-Development die agilen Werte auf der Entwicklungsebene ein. Wie bei der Implementierung des agilen Frameworks ist es hier essenziell, Test-Driven-Development an die Gegebenheiten, das Team, den Projektkontext, etc. anzupassen, damit es effektiv ist und sowohl für Kund*innen als auch für das Entwicklerteam einen Vorteil bringt. ◆

# Datenqualität: Vom Informationsfluss zum Informationsgehalt

– Sandra Wartner, MSc und Christina Hess, MSc
Data Scientists in der Unit Logistics Informatics

## Warum sich sauberes Daten(qualitäts)management auszahlt

Entscheidungen zu treffen ist nicht immer einfach – besonders dann nicht, wenn diese für die grundlegende Ausrichtung des Unternehmens relevant sind und damit Einfluss auf eine weitreichende Unternehmensstruktur nehmen können. Umso wichtiger ist es, im Entscheidungsfindungsprozess möglichst alle Einflussfaktoren zu kennen, Fakten zu quantifizieren und diese (anstatt Annahmen zu treffen) direkt miteinzubeziehen, um potenzielle Risiken zu minimieren und kontinuierlich Verbesserungen in der Unternehmensstrategie zu erzielen. Ein möglicher Quick-Win für Unternehmen lässt sich aus den Unternehmensdaten ableiten: zukunftsorientiertes Datenqualitätsmanagement – ein Prozess, dem leider häufig viel zu wenig Aufmerksamkeit geschenkt wird. Warum das Vorhandensein großer Datenströme meist nicht ausreicht, welche ausschlaggebende Rolle der Zustand der gespeicherten Daten in der Datenanalyse und im Decision Making spielt, wie man gute Datenqualität erkennt und warum das auch für Ihr Unternehmen wichtig werden kann, erklären wir Ihnen hier.

### Was kosten schlechte Daten?

Im Supermarkt achten wir beim Einkauf auf das Bio-Gütesiegel und Produkt-Regionalität, bei neuer Kleidung soll das Material aus erneuerbaren Rohstoffen bestehen und keinesfalls durch Kinderarbeit produziert werden, und der Stromanbieter wird nach Kriterien wie Sauberkeit und Transparenz ausgewählt – weil wir wissen, welchen Einfluss unsere Entscheidungen mit sich ziehen können. Warum dann nicht auch in der Datenhaltung- bzw. im Datenmanagement dem Prinzip Qualität vor Quantität treu bleiben?

In Zeiten von Big Data werden im Sekundentakt Informationsfluten generiert, die häufig als Grundlage für Business-Entscheidungen dienen. Treffen wir die falschen Entscheidungen, kann das einer Studie des MIT [1] zufolge sogar bis zu 25 % des Umsatzes kosten. Zusätzlich zum finanziellen Verlust ist unnötig hoher Ressourceneinsatz bzw. Mehraufwand zum Beheben der dadurch entstandenen Fehler und Richtigstellen der Daten notwendig, und der Anteil zufriedener Kund*innen sowie das Vertrauen in den Wert der Daten sinkt. Nicht nur Google hatte bislang mit drastischen Folgen von Fehlern im Datenbestand zu kämpfen, u.a. mit ihrem Produkt Google Maps [2]. Dabei führten Adressangaben am falschen Standort sogar soweit, dass ein Abrissunternehmen versehentlich das falsche Haus dem Erdboden gleichgemacht hat; fälschlicherweise geringere Kilometerangaben zum Navigationsziel über nicht existierende Straßen ließen Autofahrer*innen in der Wüste stranden oder Sehenswürdigkeiten schienen plötzlich an falschen Stellen auf. Auch die NASA musste am 23.09.1999 zusehen, wie beim Anflug auf den Mars die Mars Climate Orbiter und damit mehr als 120 Mio. $ verglühten – der Grund: ein Einheitenfehler [3]. Auch wenn die Auswirkungen schlechter Datenqualität nicht ganz so weitreichend sein können wie bei den großen Playern, betrifft das Thema Datenqualität dennoch jedes Unternehmen.

Dabei ist Datenqualität aus unternehmerischer Sicht kein IT-Problem, sondern ein Business-Problem. Dieses resultiert meist daraus, dass Business Professionals die Wichtigkeit der Datenqualität nicht bzw. zu wenig bewusst ist und sukzessive auch das Datenqualitätsmanagement schwach ausgeprägt ist oder überhaupt fehlt. Die Verknüpfung von Datenqualitätspraktiken mit Geschäftsanforderungen verhilft dabei, Ursachen für Qualitätseinbußen zu identifizieren und zu beheben, die Fehlerquote und Kosten dadurch zu senken und schlussendlich bessere Entscheidungen treffen zu können.

Trotz der oben genannten Kriterien ist es nicht einfach, gute oder schlechte Datenqualität anhand von konkreteren Merkmalen zu beschreiben, da Daten in unterschiedlichsten Strukturen existieren, die sich stark in ihren Eigenschaften unterscheiden. Der gesammelte Datenbestand im Unternehmen setzt sich abhängig vom Strukturierungsgrad aus unterschiedlichen Daten zusammen:

- Strukturierte Daten sind Informationen, die einem vorgegebenen Format bzw. einer definierten Struktur folgen (meist in Tabellenform aufbereitet) und gegebenenfalls sogar systematisch sortiert sind. Damit eignen sich diese besonders für Suchanfragen spezifischer Teile an Informationen wie bspw. einem bestimmten Datum, PLZ oder Namen.
- Unstrukturierte Daten hingegen liegen in einer nicht normalisierten, nicht identifizierbaren Datenstruktur vor und erschweren dadurch die Verarbeitung und Analyse. Darunter fallen beispielsweise Bilder, Audiodateien, Videos oder Text.
- Semi-strukturierte Daten folgen einer Grundstruktur, die sowohl strukturierte als auch unstrukturierte Daten beinhaltet. Ein klassisches Beispiel dafür sind E-Mails, für die u.a. Absender, Empfänger und Betreff im Nachrichtenkopf angegeben werden müssen, der Inhalt der Nachricht jedoch aus beliebigem, strukturlosem Text besteht.


Abbildung 1: Von der Datenqualität zu nachhaltigen Entscheidungen und Produkten

## Datenbasis und Datenqualität – Was steckt dahinter?

Es gibt bereits viele Definitionen zum Begriff Datenqualität, dennoch lässt sich eine allgemeine Aussage darüber nur bedingt treffen, da gute Datenqualität meist domänen-spezifisch definiert ist. Ein großes Datenset alleine (Quantität) ist noch kein Indiz dafür, dass die Daten wertvoll sind. Entscheidend für den tatsächlichen Nutzen der Daten im Unternehmen ist vor allem, ob diese die Realität korrekt widerspiegeln (Qualität) und ob die Daten für den vorgesehenen Anwendungsfall geeignet sind.

Es gibt verschiedene allgemeine Ansätze und Leitfäden zur Bewertung der Qualität von Daten. Oftmals wird gute Datenqualität sehr eng als die inhaltliche Korrektheit verstanden, wodurch andere wichtige Aspekte wie Vertrauenswürdigkeit, Verfügbarkeit oder Verwendbarkeit ignoriert werden. Cai und Zhu (2015) [4] beispielsweise definieren die in Abbildung 2 dargestellten Datenqualitätskriterien Availability, Relevance, Usability, Reliability und Presentation Quality. Im folgenden werden einige relevante Punkte für die Durchführung von datengetriebenen Projekten anhand dieser Kriterien diskutiert.

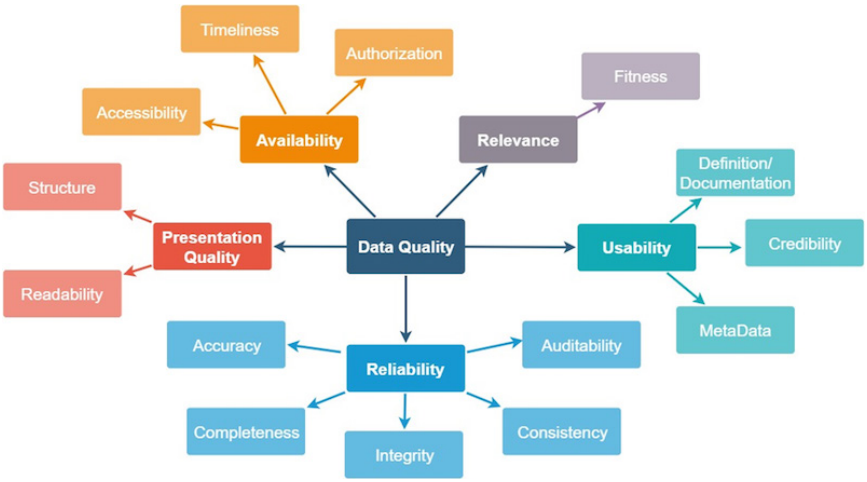



Abbildung 2: Datenqualitätskriterien nach Cai and Zhu (2015)

**Relevanz**

Bei Datenanalyseprojekten ist es zu Beginn besonders wichtig, sich bewusst zu überlegen, ob die Daten, die dem Vorhaben zugrunde liegen sollen, für den gewünschten Anwendungsfall geeignet sind. Bei KI-Projekten ist es daher immer wichtig, folgende Frage mit „Ja" beantworten zu können: Sind die Informationen, die zum Beantworten der Fragestellung benötigt werden, überhaupt zur Gänze in den Daten vorhanden? Wenn selbst Domänenexpert*innen diese Informationen in den Daten nicht ausmachen können, wie soll dann eine KI die Zusammenhänge lernen? Machine Learning Algorithmen arbeiten schließlich auch nur mit den ihnen zur Verfügung gestellten Daten und können keine Information generieren oder verwerten, die nicht in diesen enthalten ist.

**Verwendbarkeit**

Oftmals werden Metadaten gebraucht, um Daten richtig interpretieren und damit überhaupt nutzen zu können. Beispiele hierfür sind etwa die Kodierung, Herkunft oder auch Zeitstempel. Die Herkunft kann beispielsweise auch Auskunft über die Glaubwürdigkeit der Daten geben. Ist die Quelle der Daten wenig vertrauenswürdig oder stammen die Daten aus menschlichen Eingaben, lohnt es sich gegebenenfalls, diese genau zu prüfen. Je nach Inhalt der Daten ist auch eine gute Dokumentation wichtig – werden beispielsweise Codes verwendet, ist zur Interpretation vielleicht wichtig, wofür diese stehen, oder es werden zusätzliche Informationen benötigt, um Datumswerte, Zeitstempel oder ähnliches richtig lesen zu können.

**Zuverlässigkeit**

Der wahrscheinlich wichtigste Aspekt bei der Bewertung von Daten und ihrer Qualität ist, wie richtig und verlässlich diese sind. Dabei ist entscheidend, ob die Daten nachvollziehbar sind, ob sie vollständig sind und ob es Widersprüche darin gibt – schlicht, ob die enthaltenen Informationen stimmen. Wenn man schon das Gefühl hat, dass man den eigenen Daten und ihrer Richtigkeit nicht vertraut, wird man auch Analyseergebnissen oder darauf trainierten KI-Modellen nicht vertrauen. In diesem Zusammenhang ist oft auch Genauigkeit ein entscheidender Faktor: Für eine Fragestellung sind grob gerundete Werte vielleicht ausreichend, während in einem anderen Fall die Genauigkeit auf mehrere Nachkommastellen essenziell ist.

**Präsentationsqualität**
Daten müssen logischerweise nicht nur von Computern, sondern vor allem auch von Menschen „verarbeitet" werden können. Sie müssen also in vielen Fällen gut strukturiert sein bzw. aufbereitet werden können, damit sie auch für uns Menschen lesbar und verständlich sind.

**Verfügbarkeit**
Spätestens dann, wenn man bestimmte Datenmengen aktiv verwenden und z.B. in ein (KI-)System einbinden möchte, muss auch geklärt werden, wer wann auf diese Daten zugreifen darf und wie dieser Zugriff (technisch) ermöglicht wird. Vor allem für die Vertrauenswürdigkeit resultierender KI-Systeme ist dies ein wichtiger Faktor. Für Echtzeit-Systeme ist auch die Aktualität entscheidend. Denn will man Daten in Echtzeit verarbeiten (z.B. zur Überwachung von Produktionsanlagen), dann muss der Zugriff darauf schnell erfolgen können und zuverlässig sein.

## Wie erkennt man schlechte Datenqualität und wie können diese Datenmängel überhaupt erst entstehen?

Schlechte Datenqualität ist meist nicht so unscheinbar, wie man glaubt. Bei genauerem Hinsehen äußern sich je nach Strukturierungsgrad der Daten schnell die unterschiedlichsten Mängel. Jetzt mal Hand aufs Herz – sind auch Ihnen bereits einige Fälle aus Abbildung 3 bekannt? Falls nicht, nutzen Sie doch die Gelegenheit und machen Sie sich auf die Suche nach diesen Konflikterzeugern, denn Sie werden mit sehr hoher Wahrscheinlichkeit einige davon auffinden. Diese Daten können in der praktischen Anwendung viele Gestalten annehmen und sich in unterschiedlichen Problematiken wie u.a. Imageschäden oder rechtlichen Folgen manifestieren (Abbildung 4 zeigt nur ein paar wenige der negativen Auswirkungen). Auch die Kosten schlechter Datenqualität können weitreichend sein, das haben wir in „Was kosten schlechte Daten" bereits klargestellt. Doch wie können solche Probleme überhaupt erst entstehen? Die grundlegenden Ursachen liegen oftmals in den fehlenden Verantwortlichkeiten zur Datenhaltung bzw. überhaupt im fehlenden Datenqualitätsmanagement, aber auch technische Herausforderungen können Probleme verursachen. Oftmals schleichen sich diese Fehler auch über die Zeit ein. Besonders fehleranfällig sind unterschiedliche Datensammlungsprozesse und im Weiteren das Zusammenführen von Daten aus verschiedensten Systemen oder Datenbanken. Auch menschliche Eingaben produzieren Fehler (z.B. Tippfehler, Verwechslung von Eingabefeldern). Ein weiteres Problem liegt in der Datenalterung: Besonders dann, wenn Änderungen in der Datenerfassung bzw. -aufzeichnung stattfinden (z.B. fehlende Sensordaten bei Maschinenumstellung, mangelnde Genauigkeit, zu kleine/zu große Abtastrate, fehlendes Know-How, sich ändernde Anforderungen an die Datenbasis), kommt es zu Problemen. Weitere Risikofaktoren sind die oftmals fehlende Dokumentation und die fehlerhafte Versionierung der Daten.

Die perfekte Datenqualität ist in der Praxis normalerweise eine utopische Vorstellung, die auch durch viele nicht oder nur schwer steuerbare Einflussfaktoren geprägt wird. Das soll allerdings niemandem die Hoffnung nehmen, denn: Meist erzielen bereits kleine Maßnahmen eine große Wirkung.

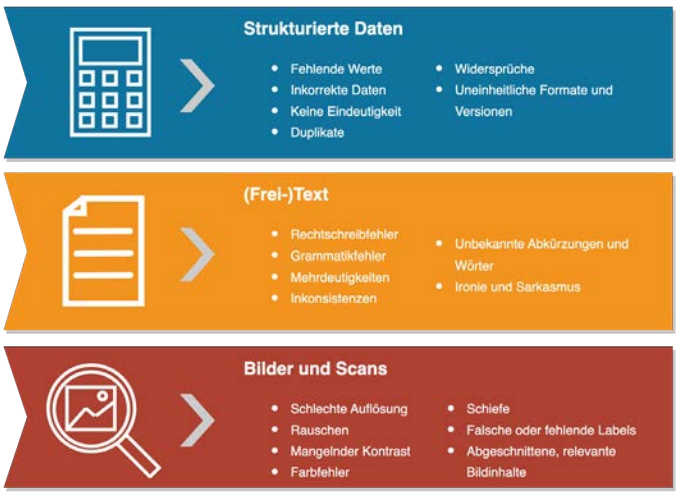



Abbildung 3: Beispiele zu schlechter Datenqualität nach Strukturierungsgrad der Daten


Abbildung 4: Negativbeispiele aus der Praxis

*[1] https://sloanreview.mit.edu/article/seizing-opportunity-in-data-quality/*
*[2] https://www.googlewatchblog.de/2019/07/google-maps-fehler-katastrophen/*
*[3] http://edition.cnn.com/TECH/space/9909/30/mars.metric/*
*[4] Cai and Zhu (2015): The Challenges of Data Quality and Data Quality Assessment in the Big Data Era*

## Daten als Treibstoff für Machine-Learning-Modelle

Nicht nur in der klassischen Datenanalyse sollte den Daten ein besonderes Augenmerk geschenkt werden, um die maximale Aussagekraft der Ergebnisse zu erhalten. Besonders im Bereich der Künstlichen Intelligenz (KI) spielt die verfügbare Datenbasis eine entscheidende Rolle und kann einem Projekt zum erfolgreichen Abschluss verhelfen oder dieses zum Scheitern verurteilen. Durch den Einsatz von KI – konkret von Machine Learning (ML) – können sich häufig wiederholende Prozesse intelligent automatisiert werden. Beispiele sind die Suche nach ähnlichen Daten, das Ableiten von Mustern oder das Erkennen von Ausreißern bzw. Anomalien. Essenziell dabei ist überdies, ein gutes Datenverständnis zu haben (Domänen-Expertise), um potenzielle Einflussfaktoren zu identifizieren und diese kontrollieren zu können. Bekannte Prinzipien wie "decisions are no better than the data on which they're based" und der klassische GIGO-Gedanke (garbage in, garbage out) unterstreichen dabei klar, wie wesentlich die notwendige Datenbasis für den Lernprozess der ML-Modelle ist. Denn nur, wenn die Datenbasis repräsentativ ist und die Realität so wahrheitsgetreu als möglich abbildet, kann auch das Modell lernen zu generalisieren und sukzessive die richtigen Entscheidungen zu treffen.

## Erfolgsfaktor Datenqualität

Für datenbasierte Arbeiten sollte Datenqualität definitiv an erster Stelle stehen. Weiters sollte das Bewusstsein zur Relevanz guter Datenqualität geschaffen bzw. nachgeschärft werden, um unternehmensweit positive Auswirkungen erzielen zu können, Kosten zu senken und freigewordene Ressourcen effizienter für die wirklich wichtigen Tätigkeiten einsetzen zu können. Unser Fazit: Ein gutes Datenqualitätsmanagement spart mehr als es kostet, ermöglicht den Einsatz neuer Methoden und Technologien und verhilft zu nachhaltigen Entscheidungen. ◆

# Arbeiten mit Fortran in 2020: Do's and Don'ts

– DI Dr. Christoph Hofer
Software Engineer in der Unit Industrial Software Applications

## Ein Knigge für Fortran-Developer

Fortran ist eine der ältesten Programmiersprachen. Der Name setzt sich zusammen aus „FORmel TRANslator". Für viele Softwareentwickler*innen ist Fortran der Archetyp für eine alte, behäbige, eingeschränkte und schwer zu verstehende Programmiersprache, mit der man am besten nichts zu tun haben möchte. Für die alten Versionen von Fortran mag dieses Vorurteil wirklich wahr sein. Fortran hat sich in seiner langen Geschichte aber viel verändert, sodass in seiner „modernen" Variante (wie etwa Fortran 2003) die Sprache einen viel schlechteren Ruf hat als ihr zusteht. Der typische Use-Case für Fortran als Programmiersprache sind rechenintensive, numerische Simulationen wie etwa Wettervorhersagen, Strömungssimulationen oder Stabilitätsberechnungen.

### Aus alt mach neu

Fortran gilt als die erste jemals realisierte höhere Programmiersprache und wurde in den Jahren 1954 – 1957 von IBM entwickelt (FORTRAN I). Der Umfang der Sprache war noch sehr eingeschränkt, zum Beispiel gab es nur Integer (Ganzzahlen) und Reals (Gleitkommazahlen) als Datentypen und noch keine Funktionen. In den folgenden Jahren wurden neue verbesserte und umfangreichere Fortran-Versionen entwickelt (FORTRAN II, FORTRAN III, FORTAN 66). Das nächste große Update bekam Fortran im Jahr 1977 (FORTRAN 77). Durch neue Features in der Sprache wurde diese Version sehr populär und damit schnell zu „dem" Fortran. Auch heute noch ist, wenn über Fortran-Code gesprochen wird, hauptsächlich FORTRAN 77 Code gemeint, was auch die vielen Vorurteile gegenüber der Sprache erklärt. Seitdem gab es noch einige Updates der Sprache, die sie an moderne Programmierkonzepte und Standards heranführen. Große Meilensteine in der Entwicklung waren die Updates zu Fortran 90 und Fortran 2003, welche neben der Namensänderung (FORTRAN → Fortran) gängige Konzepte wie unter anderem freie Sourcefile-Formate, Module, Operator Overloading, Derived Data Types, Pointers und objektorientiertes Programmieren zur Programmiersprache hinzufügten. Zusätzlich dazu gab es mit Fortran 95 und Fortran 2008 jeweils ein weiteres kleines Update der Sprache. Die aktuellste Version des Fortran Standards ist Fortran 2018, wobei noch kein Compilerhersteller alle Features unterstützt.

### Dos & Don'ts in Fortran

Durch die lange Entwicklungsgeschichte von Fortran und um Kompatibilität mit Legacy Code zu wahren, gibt es zahlreiche veraltete und teils obskure Sprachfeatures in aktuellen Fortran Compilern. Eine umfassende Sammlung von guten und schlechten Coding Practices würde den Umfang dieses Artikel sprengen. Dennoch möchten wir einige gängige Altlasten präsentieren, die sich in Legacy Code wiederfinden können, genauso wie ausgewählte Möglichkeiten, die neue Fortran-Standards bieten. Für eine umfangreiche Liste von Dos and Don'ts verweisen wir auf [1].

**Don't use common block and equivalent statement**

In Fortran 77 und früher war es üblich, dass verschiedene Variablen auf dieselbe Speicheradresse verweisen, nämlich mittels common block und equivalent statement. Diese Ausdrücke wurden genutzt, um Information zwischen Subroutinen zu teilen oder um (teuren) Speicherplatz für temporäre Variablen wiederzuverwenden. Mittlerweile sind diese Ausdrücke als veraltet deklariert und sollten nicht mehr verwendet werden. Um Daten zwischen Programmteilen auszutauschen, sollten Module genutzt werden und Speicherplatz bei Bedarf dynamisch allokiert und deallokiert werden.

**Avoid using GOTO**

Kein anderer Ausdruck ist so sehr mit Fortran verwurzelt wie das GOTO. In alten Programmen findet sich sehr oft eine exzessive Nutzung von GOTOs, teilweise auch geschuldet dem Mangel an alternativen Konstrukten. Über die Zeit haben sich verschiedene Varianten des GOTOs entwickelt, wie dem computed GOTO statement oder dem assigned GOTO statement. Auch gab es für die Handhabung von Schleifen oder IF-statements Varianten, die mit Sprüngen zu entsprechenden Labels arbeiteten. In modernem Fortran-Code sollten all diese Varianten der GOTOs, sofern möglich, vermieden werden und durch IF und SELECT Case (= switch) ersetzt werden. Eine nennenswerte Ausnahme für die Notwendigkeit von GOTOs in Fortran-Code ist das Fehlermanagement, da Exceptions in Fortran nicht existieren.

### Avoid SAVE attributes

Das SAVE-Attribut erlaubt es, dass Variablen bei wiederholten Funktionsaufrufen ihren Wert beibehalten. Gerade in Verbindung mit Parallelisierung kann dies zu schwer zu findenden Bugs und Dataraces führen. Das SAVE-Attribut kann gefahrlos bei Variablen verwendet werden, die immer denselben Wert bei jedem Funktionsaufruf besitzen, um so etwas Performance zu gewinnen. In allen anderen Fällen sollte es gemieden werden. Ein besonders hinterhältiges „Feature" von Fortran ist, dass alle Variablen, die bei ihrer Deklaration gleich initialisiert werden, automatisch ein implizites SAVE-Attribut erhalten.

### Use implicit none

Ein Konzept aus alten Fortran-Standards war, dass undeklarierte Variablen automatisch als REAL deklariert werden – außer jene, die mit dem Buchstaben i, j, k, l, m oder n beginnen und als INTEGER deklariert werden. Dieses Konzept ist sehr fehleranfällig, vor allem deshalb, weil der Compiler keine Fehlermeldung ausgibt, wenn nicht deklarierte Variablen verwendet werden, z.B. durch einen Tippfehler. Durch dieses Konzept hat sich folgender Witz über Fortran eingebürgert: Fortran ist die einzige Sprache wo „God is Real" gilt. Diese automatische Variablendeklaration kann mithilfe von IMPLICIT NONE für den aktuellen Bereich deaktiviert werden und es ist „good practice", diese im ganzen Programmcode umzusetzen.

### Make use of derived data types and classes

Mit Fortran 90 wurde begonnen, die Sprache Richtung objekt-orientierter Programmierung weiter zu entwickeln. Mit diesem Standard wurden User-Defined Datatypes eingeführt, die es erstmals erlaubten, wiederverwendbare Strukturen von logisch zusammengehörenden Daten zu verwenden. Ebenso wurde das Konzept von Generics hinzugefügt, sodass derselbe Funktionsname mit verschiedene Typen verwendet werden kann (es muss aber trotzdem die Funktion für jeden Typ programmiert werden). Mit dem Fortran 2003 Standard wurde nochmal das objektorientierte Programmieren forciert. Spätestens seit diesem Zeitpunkt sollte versucht werden, Daten und Logik in sinnvolle Klassen zu kapseln und Programmteile über wohldefinierte Interfaces interagieren zu lassen.

### Use the module system

Fortran 90 leitete auch eine neue Form einer Programmorganisation ein, nämlich das Modulsystem. Ein Modul besteht aus einer Menge von Deklarationen von Daten, Funktionen und Funktionsschnittstellen, die dann in anderen Programmteilen verwendet und sichtbar gemacht werden können. Zusätzlich bieten Module die Möglichkeit, den Zugriff der beinhalteten Funktionen/Daten mittels PRIVATE/PUBLIC einzuschränken. Seit dem Fortran 2008 Standard gibt es Submodules, die es nun dem Programmierer ermöglichen, den Programmcode in ein separates Submodule auszulagern. Die Notwendigkeit für diese ist einerseits, sehr große und unübersichtliche Module zu vermeiden, die Schnittstelle des Moduls klar ersichtlich zu haben, andererseits auch die Recompilationszeiten zu reduzieren.

### Don't rely on short-circuit evaluation

Sehr viele Programmiersprachen evaluieren bei einer logischen Kombination von zwei Ausdrücken nur den ersten, wenn durch diesen das Ergebnis bereits feststeht. Diese Verfahren wird als Short-Circuit-Evaluation bezeichnet. In Fortran ist es allerdings dem Compiler überlassen, ob Short-Circuit-Evaluation verwendet wird, d.h. im Fortran-Standard ist dies nicht verboten, aber auch nicht vorgeschrieben. Ein typischer Anwendungsfall für die Notwendigkeit von Short-Circuit-Evaluation wäre die Abfrage auf der rechten Seite.

```
IF (PRESENT(x) .AND. x > 0 ) THEN
      do something with x
END IF
```

In Fortran gibt es die Möglichkeit von optionalen Argumenten, d.h Parameter einer Funktion, die nicht zwingend übergeben werden müssen. Mit Hilfe der Funktion PRESENT(x) kann überprüft werden, ob der Parameter x übergeben wurde. In dem Beispiel rechts, wird nach der Überprüfung eine Abfrage getätigt, ob x größer als 0 ist. Falls x nicht übergeben wird, dann würde durch Short-Circuit-Evaluation die Abfrage x>0 nicht mehr getätigt werden, da bereits die erste Bedingung nicht erfüllt ist. Durch die möglicherweise nicht erfolgte Short-Circuit-Evaluation würde das Programm an dieser Stelle aber abstürzen. Die korrekte Schreibweise ist das Aufteilen auf zwei einzelne Bedingungen wie im Beispiel rechts gezeigt.

```
IF (PRESENT(x)) THEN
     IF(x > 0 ) THEN
          do something with x
     END IF
END IF
```

Andere typische Fälle sind Abfragen, ob ein Pointer einer Speicheradresse zugeordnet ist oder ob eine mathematische Operation mit den Eingabewerten erlaubt ist (Division, Wurzel). ◆

*[1] S. Chapman, Fortran for scientists & engineers, McGraw-Hill Education, 4. Edition, 2017*

# Entscheidungsunterstützung für Industrie und Wirtschaft: Optimierung will gelernt sein.

– DI Manuel Schlenkrich
Mathematical Optimization Engineer in der Unit Logistics Informatics

## Künstliche Intelligenz und Optimierung - das Beste zweier Welten

„Learning by doing", „Übung macht den Meister" oder „Aus Fehlern lernt man" – warum diese Phrasen nicht nur für uns Menschen motivierend sind, sondern auch mathematischen Algorithmen zum Erfolg verhelfen, erfahren Sie in diesem Beitrag. Für komplexe Problemstellungen in Wirtschaft und Industrie werden Optimierungsmodelle und Lösungsalgorithmen täglich verwendet, um schwierige Entscheidungen zu treffen. Dabei sind die meisten Entscheidungsträger*innen mit einem hochdynamischen Umfeld, unsicheren Prognosen und unzähligen Variablen konfrontiert. Um mit diesen Herausforderungen umgehen zu können und enormen Rechenaufwand einzusparen, werden Methoden entwickelt, die mit Hilfe von künstlicher Intelligenz funktionieren. Klassische Optimierungsverfahren werden mit maschinellem Lernen kombiniert, um den hohen Anforderungen gerecht zu werden – denn eines kann gesagt werden: Optimierung will gelernt sein.

### Smarte Algorithmen für komplexe Aufgaben

Erfolgreich zu sein, heißt gute Entscheidungen zu treffen. Doch wann ist eine Entscheidung optimal und vor allem, wie findet man diese? Wie viel Paar Skier soll ein Sportartikelhersteller produzieren, wenn deren Nachfrage nur geschätzt werden kann? Wie viele Aktienanteile soll eine Investorin für ihr Portfolio ankaufen, wenn der zukünftige Aktienkurs höchst unsicher ist? An welchen Orten sollen COVID-19 Testzentren errichtet werden, um möglichst vielen Menschen eine unkomplizierte Anfahrt zu gewährleisten? So unterschiedlich diese Anwendungen auch klingen mögen, sie alle haben eines gemeinsam: Optimale Entscheidungen werden gesucht. Bewährte Werkzeuge zur Entscheidungsunterstützung sind mathematische Optimierungsmodelle, die reale Probleme auf ihre wesentlichen Merkmale herunterbrechen. Für diese Modelle können dann, durch den Einsatz von Lösungsverfahren, bestmögliche Entscheidungen gefunden und auf die realen Problemstellungen angewandt werden.

Die Ansprüche der Wirtschaft und Industrie an diese mathematischen Modelle werden allerdings immer höher, die Problemstellungen immer komplexer und Aspekte wie schwankende Parameter und dynamische Umgebungen immer relevanter. Diese Entwicklung führt dazu, dass klassische exakte Lösungsverfahren Stunden, Tage oder sogar Wochen benötigen würden, um Entscheidungen unter realistischen Bedingungen zu berechnen. Gleichzeitig steigt die Verfügbarkeit einer großen Menge an Daten, vor allem angetrieben durch den technologischen Fortschritt und die voranschreitende Digitalisierung industrieller Prozesse. So beschäftigt sich ein neuer Trend mit der Verschmelzung von Methoden der klassischen Optimierung mit Ansätzen des maschinellen Lernens zu effizienten datengetriebenen Lösungsverfahren. Diese Verfahren sind in der Lage, durch Techniken der Künstlichen Intelligenz Muster in den Problemeigenschaften zu finden, komplexe Zusammenhänge zu vereinfachen und vielversprechendes Lösungsverhalten zu erlernen. Das Ziel ist es dabei, das Beste aus den Welten der Optimierung und des maschinellen Lernens zu kombinieren, um selbst für hochkomplexe Entscheidungsprobleme gute Lösungen in angemessener Zeit zu finden.

### Optimierung meets AI: das Beste zweier Welten kombinieren

Zur Entscheidungsfindung wird ein mathematisches Modell als Abbild der Realität erstellt, bei dem es eine Zielfunktion unter Einhaltung verschiedener Restriktionen zu optimieren gilt. Um eine optimale Lösung für ein Model zu finden, gibt es zwei verschiedene Ansätze. Einerseits können exakte Lösungsverfahren angewandt werden, die bei Auffinden einer Lösung auch deren globale Optimalität garantieren können. Diese Lösungsverfahren benötigen aber in der Regel enormen Rechenaufwand und liefern besonders für große und realistische Modelle oft keine Lösung in angemessener Zeit. Neben den exakten Verfahren existiert auch die Gruppe der heuristischen Lösungsverfahren. Dabei handelt es sich um Algorithmen, die auf konkrete Problemstellungen zugeschnitten wurden, um in kurzer Zeit gute Lösungen zu finden. Es kann dabei allerdings keine Aussage getroffen werden, ob es sich bei der Lösung um ein globales Optimum handelt. Den heuristischen Verfahren übergeordnet sind die sogenannten Metaheuristiken. Diese beschreiben eine allgemeine algorithmische Vorgangsweise, die auf eine Vielzahl an Problemstellungen anwendbar ist.

Ansätze des maschinellen Lernens können nun mit beiden Arten von Lösungsverfahren kombiniert werden. Bei exakten Methoden soll dies meist eine Verringerung des Rechenaufwandes bewir-

ken, indem die künstliche Intelligenz vielversprechende Bereiche des Lösungsraums auffindet, effiziente Ausführungsreihenfolgen innerhalb des Algorithmus bestimmt oder die Zielfunktionswerte schneller berechnet. Bei heuristischen Methoden geht es vorwiegend darum, durch den Einsatz von lernenden Algorithmen bessere Lösungsqualitäten zu erzeugen. Dabei sollen vor allem Aufgaben wie Parameter Tuning, Algorithmus-Auswahl oder Operator-Management durch die Künstliche Intelligenz übernommen werden.

## Wie wird gelernt?

Spricht man von Verfahren der Optimierung, die einen lernenden Aspekt beinhalten, so lassen sich grundsätzlich zwei Arten von Lernmethoden unterscheiden. Diese beiden Arten haben ihren Ursprung in zwei verschiedenen Motivationen zur Anwendung solcher Algorithmen. In manchen Bereichen besteht bereits umfassendes theoretisches oder empirisches Wissen von Expert*innen über das Entscheidungsumfeld, das durch eine exakte Optimierungsmethode abgebildet werden könnte. Die Anwender*innen möchten nun den lernenden Algorithmus dazu verwenden, dieses bekannte Wissen zu approximieren und somit erheblichen Rechenaufwand einzusparen. Es soll eine Verhaltensregel oder auch „policy" ($\pi_{ml}$) erlernt werden, die die Entscheidungen der Expert*innen ($\pi_{Experte}$) imitiert und somit ähnliche Resultate erzielt. Bei dieser Art des Lernens versucht der Algorithmus nicht, die Qualität der Resultate zu optimieren, sondern minimiert die Diskrepanz zwischen den getroffenen Entscheidungen und den Demonstrationen der Expert*innen. Es kommt allerdings auch vor, dass nicht genügend Informationen über das Entscheidungsumfeld vorhanden sind und neue Entscheidungsstrategien entwickelt werden sollen. Für diesen Fall wird der lernende Algorithmus in einem Trial-and-Error-Setting darauf trainiert, ein sogenanntes Reward Signal zu maximieren. Bei dieser Art des Lernens erforscht der Algorithmus selbständig den Entscheidungsraum und gewinnt mit jedem Erkundungsschritt neue Informationen über die Qualität von getroffenen Entscheidungen.

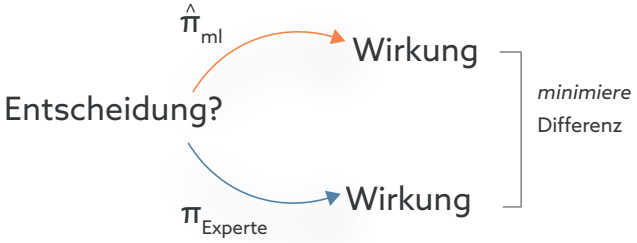



Abbildung 1: Lernen durch Demonstration

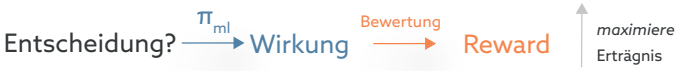



Abbildung 2: Lernen durch Erfahrung

## Wann wird gelernt?

Die vorgestellten Methoden lassen sich nicht nur hinsichtlich ihrer Art des Lernens gliedern, sondern auch danach, in welcher Struktur der Lernprozess in den Algorithmus eingebunden wird. Maschinelles Lernen kann im Vorhinein verwendet werden, um die Optimierungsmethode zu konfigurieren, beispielsweise durch Erlernen vielversprechender Parameter oder Ausführungsreihenfolgen innerhalb des Algorithmus. Eine andere Möglichkeit besteht darin, maschinelles Lernen und Optimierungsmethode abwechselnd iterativ auszuführen. Dabei gibt der Optimierungsalgorithmus laufend Informationen zur aktuellen Lösungsqualität an den lernenden Teil ab, während dieser wiederum neue vielversprechende Lösungsbereiche aus den Informationen ableitet und zurückgibt. Bei der dritten Variante ersetzt der lernende Teil das eigentliche Optimierungsverfahren und ermittelt zu einem gegebenen Problem bereits eine fertige Lösung. In diesem Fall spricht man auch von End-to-End-Learning.

**Konfiguration**

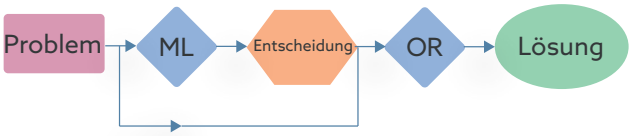



Abbildung 3: Algorithmuskonfiguration durch ML am Beginn

**End-to-end learning**

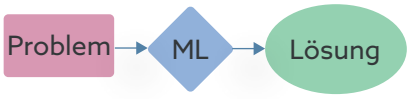



Abbildung 5: End-to-end learning

**Konfiguration**

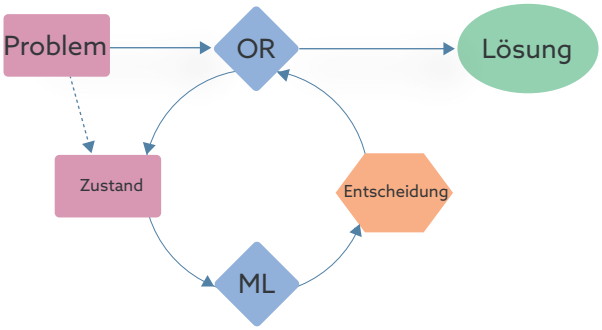



Abbildung 4: Iterative Algorithmuskonfiguration durch ML

## Was wird gelernt?

Nachdem die Frage geklärt ist, wie gelernt wird, bleibt noch die viel wichtigere Frage offen: Was wird nun eigentlich genau erlernt? Dafür gibt es verschiedene Ansätze, abhängig von den jeweiligen Eigenschaften der verwendeten Optimierungsmethode und problemspezifischen Faktoren.

### Parameter-Tuning durch ML

Metaheuristiken enthalten in der Regel etliche Parameter, die signifikanten Einfluss auf die Performance haben. Maschinelles Lernen kann eingesetzt werden, um diese Parameter für eine bestimmte Problemklasse oder auch für einzelne Instanzen eines Problems zu erlernen. Dabei kommen vor allem Techniken der linearen oder logistischen Regression, neuronale Netze oder Response-Surface-Methoden zum Einsatz.

### Zielfunktionsauswertung durch ML

Für komplexe Probleme ist die Auswertung der Zielfunktion mit hohem Rechenaufwand verbunden. Maschinelles Lernen kann verwendet werden, um eine Approximation der Zielfunktion zu erstellen und somit die Auswertung zu beschleunigen. Polynomielle Regression, neuronale Netzwerke oder Markov Fitness Models sind populäre ML Methoden, die dafür infrage kommen.

### Populations- und Operatormanagement durch ML

Viele Metaheuristiken (etwa lokale Suchverfahren) verwenden Operatoren, um ausgehend von bereits erzeugten Lösungen neue vielversprechende Lösungen zu generieren. Bei genetischen Algorithmen spricht man auch von Populationen, die durch Mutations- und Kreuzungsoperatoren verändert werden. Oft wird der Einsatz dieser Operatoren schon im Vorhinein durch fixe Regeln basierend auf den Lösungseigenschaften vorgeschrieben. Diese Regeln können allerdings auch durch maschinelles Lernen ständig angepasst und verbessert werden. So eignen sich beispielsweise inverse neuronale Netze oder Klassifikationsalgorithmen aus dem Bereich der Symobolic-Learning-Methoden, um Regeln zu erlernen, die bereits erfolgte Fehlversuche nicht wiederholen und erklären können, warum manche Operatoren an dieser Stelle besser geeignet sind als andere.

### Algorithmus-Auswahl durch ML

Es kann vorkommen, dass ein ganzes Portfolio an verschiedenen Lösungsmethoden für dieselbe Problemklasse zur Verfügung steht und man daran interessiert ist, welche davon die beste Performance liefert. Das Algorithm-Selection Problem beschreibt genau diesen Sachverhalt, bei dem eine Lösungsmethode ***a*** aus der Menge der verfügbaren Methoden A so gewählt werden soll, dass die Performance von a, angewandt auf ein Problem x, unter allen Methoden in A bestmöglich ist. Dieses Auswahlproblem soll unter Abhängigkeit der Problemeigenschaften des Problems x gelöst werden. Klassifizierungsalgorithmen und neuronale Netze eignen sich, um das verfügbare Portfolio A basierend auf den Problemeigenschaften in mehr oder weniger vielversprechende Methoden zu unterteilen.

### Bestimmung der Ausführungsreihenfolge durch ML

Das Branch-and-Bound-Framework ist ein weit verbreitetes exaktes Lösungsverfahren. Dabei wird die Problemstellung Stück für Stück in kleinere Teilprobleme zerlegt („Branching") und lässt sich in einer Baumstruktur („Branch and Bound Tree" / „Suchbaum") darstellen, in der jeder Knoten eine unvollständige Lösung des Gesamtproblems repräsentiert. Für diese Knoten können dann durch Lockerung der Restriktionen untere Schranken berechnet werden. Gleichzeitig können durch heuristisches Lösen der Teilprobleme obere Schranken gefunden werden und sobald für einen Knoten im Suchbaum die untere Schranke über der oberen Schranke liegt, kann der gesamte „Ast" verworfen werden, was wiederum den Suchraum einschränkt („Bounding"). Je schneller gute obere Schranken gefunden werden, desto schneller können ganze Bereiche des Suchraums verworfen werden, was zu einer erheblichen Performancesteigerung des Branch-and-Bound-Algorithmus führt. Um solche oberen Schranken zu erhalten, werden verschiedene Heuristiken eingesetzt, die in jedem Knoten versuchen, eine zulässige gute Lösung zu generieren. Es ist allerdings stark vom jeweiligen Problem und von der jeweiligen Instanz abhängig, welche vorhandene Heuristik die besten Ergebnisse liefert. Es wäre wünschenswert, die jeweils beste Heuristik früh einzusetzen und nicht davor mit schlechteren Heuristiken Zeit zu verschwenden. Üblicherweise wird die Ausführungsreihenfolge der Heuristiken im Vorhinein definiert, unabhängig von der jeweiligen Probleminstanz und unfähig, auf dynamische Änderungen während des Suchlaufs zu reagieren. Ein neuer Ansatz verbessert die Ausführungsreihenfolge der Heuristiken datenbasiert und passt sie während des Suchlaufs ständig an.

## Zusammenfassung

In einer Zeit, in der große Mengen an Daten bei nahezu jedem Prozess gesammelt werden, ist es naheliegend, diese in der Entscheidungsfindung zu berücksichtigen und die dazu verwendeten Optimierungsalgorithmen mit lernenden Komponenten zu unterstützen. Der Einsatz von maschinellem Lernen hat das Potenzial, den Rechenaufwand exakter Lösungsverfahren zu verringern und die Lösungsqualität heuristischer Verfahren zu verbessern. Die Verschmelzung der beiden Welten „klassische Optimierung" und „Künstliche Intelligenz" liegt im Trend und man darf gespannt sein, welche Resultate die Forschung auf diesem interdisziplinären Gebiet noch erzielen kann. Eines steht jedenfalls fest: Es ist spannend, wie vielfältig die beiden Ansätze kombiniert werden können und wie sich Ergebnisse dadurch verbessern lassen – gelernt ist eben einfach gelernt! ◆

RISC Software GmbH
www.risc-software.at

linkedin.com/company/risc-software-gmbh

twitter.com/RISC_Software

facebook.com/RISC.Software

xing.com/pages/riscsoftwaregmbh

**Impressum**
Herausgeber und
Medieninhaber:
RISC Software GmbH,
Softwarepark 32a, 4232 Hagenberg,
+43 7236 93028, office@risc-software.at
Für den Inhalt verantwortlich: DI Wolfgang Freiseisen
Chefredaktion: Mag. Cornelia Staub
Design und Grafische Gestaltung: Melanie Laßlberger, MSc

Version: 1.0 | 11.01.2023
Bildnachweis: RISC Software GmbH, iStock.com, Adobe Stock
wenn nicht anders angegeben