RISC Software GmbH
www.risc-software.at

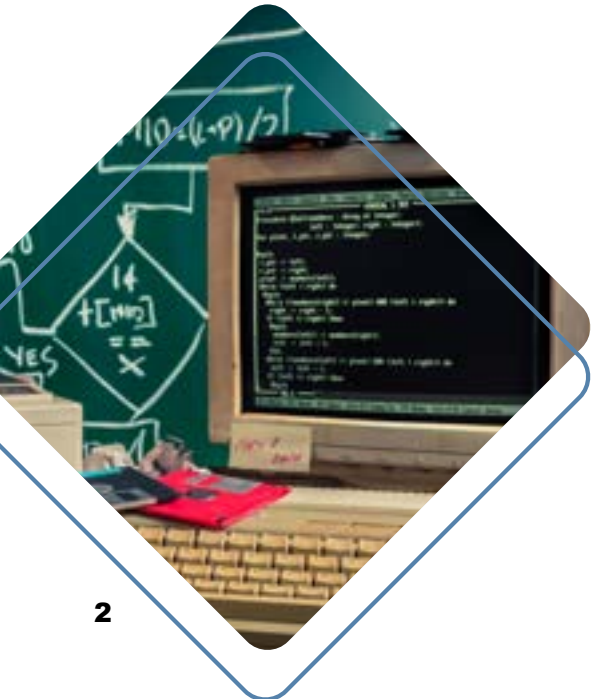
INSIGHTS[1]

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

Agile Softwareentwicklung

Software-Reengineering

Data Science and Prescriptive Analytics



INHALT

Liebe Leserin, lieber Leser,

was Sie hier in Händen halten, ist die erste Ausgabe unserer gesammelten Fachbeiträge. Unsere Expert*innen bieten Ihnen sowohl allgemeine Themeneinstiege als auch tiefgreifende Informationen zu den Gebieten Optimierung, Simulation, Data Sciene und Prescripte Analytics, sowie Software-Reengineering und agile Softwareentwicklung. Vieles, das Sie hier lesen, ist langjährig aufgebautes und in Forschungsprojekten erarbeitetes Know-how gepaart mit den modernsten wissenschaftlichen Methoden, die erfolgreich in Kundenprojekten umgesetzt wurden. Diese Erfolge sind nicht zuletzt dem persönlichen Einsatz unserer Mitarbeiter*innen geschuldet. 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.

Die digitale Transformation betrifft alle Abteilungen und Organisationsbereiche und fordert einen hohen Grad an Veränderungen, oftmals auch einen kulturellen Wandel. Wir verstehen uns als Enabler und bringen dafür die umfassenden multidisziplinären Kompetenzen mit. Wir stellen die Expert*innen, mit deren Erfahrungen und Domain-Wissen dieser Prozess vorangetrieben wird. Dies ist für unsere Kund*innen ein enormer Vorteil, da die technologischen Herausforderungen aufgrund der zunehmenden Dynamik und Komplexität immer vielfältiger werden.

Die RISC Software GmbH ist nicht nur eine erfahrene Partnerin für Forschungsprojekte und langjährige Kooperationen, sondern auch Wissensvermittlerin. Wir bieten im Rahmen unser AI Academy Workshops zu Themen wie Agiles Projektmanagement, Artificial Intelligence und vieles mehr, auch nach speziellen Kundenanforderungen. Wir unterstützen Sie in Ihrem Aufbau von spezifischem Know-how in Zusammenhang mit Digitalisierung und neuen Technologien.

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 ihrem vielfältigen Fachwissen!

Viel Spaß beim Lesen,

Wolfgang Freiseisen
CEO Software GmbH

Agile vs. klassische Softwareentwicklung

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

Das passende Vorgehensmodell für Ihr Projekt

Klassische Ansätze wie beispielsweise Wasserfall, V-Modell oder Spiralmodell oder agile Ansätze wie zum Beispiel Scrum oder Kanban? Zu Projektbeginn stellt sich die Frage, welches der existierenden Vorgehensmodelle wohl am Besten für das aktuelle Vorhaben geeignet ist. Die RISC Software GmbH greift auf langjährige Erfahrung mit verschiedenen Vorgehensmodellen zurück und berät ihre Kund*innen zu Beginn eines Projektes, welches Vorgehensmodell zu empfehlen ist.

Veränderungen

Klassische Vorgehensmodelle verfolgen einen linearen Ansatz, wo von Anfang an auf einen definierten Endzustand hingearbeitet wird. Hierbei passiert das Projekt verschiedene Phasen in sequenzieller Abfolge. Die Phasen werden üblicherweise durch Meilensteine abgeschlossen. Aufgrund der starren Übergänge und Abfolge der Phasen wird möglichst vermieden, neue Anforderungen oder eine Veränderung von Anforderungen in ein Projekt einzubringen. Dies ist meist mit den budgetierten und zeitlichen Zielen nicht vereinbar.

Agile Vorgehensweisen setzen anstelle dieser sequenziellen Abfolge auf ein iterativ-inkrementelles Modell. Iterativ bedeutet, dass die Produktentwicklung in Zyklen geschieht, während inkrementell bedeutet, dass bei jedem Zyklus ein potentiell nutzbares Produktinkrement entsteht. Dies hat zur Folge, dass notwendige Planabweichungen frühzeitig erkannt werden können.

Feedback

Das Erkennen der Notwendigkeit von konkreten Veränderungen stellt jedes Projektteam vor eine Herausforderung. In klassischen Vorgehensmodellen geschieht dies bestenfalls bei den Meilensteinen, also zwischen zwei Phasen – schlimmstenfalls erst zu Projektende. Bei agilen Vorgehensweisen beinhaltet jede Iteration das zeitnahe Feedback der Kund*innen, wodurch vermieden wird, dass das Endprodukt nicht den Marktanforderungen genügt.

Abbildung 1: Wasserfallmodell versus iterativ inkrementellem Modell

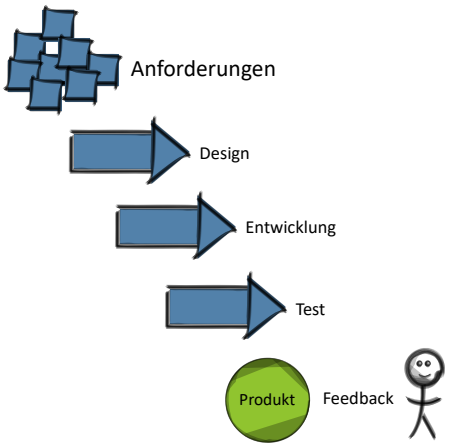


Abbildung 2: Von der Anforderung bis zum Feedback

Komplexität

Projekte scheitern aufgrund der Fehleinschätzung ihrer Komplexität. Die Definition von Komplexität gem. Cynefin-Framework ergibt sich daraus, dass der Zusammenhang zwischen Ursache und Wirkung nicht im Voraus vorhersagbar ist. Komplexe Systeme benötigen ermergente Praktiken um ein Produktziel zu erreichen. Im Gegensatz dazu stehen komplizierte Systeme, bei denen – mit der entsprechenden Wissensbasis – der Zusammenhang zwischen Ursache und Wirkung vorhersagbar ist. Somit können komplizierte Systeme bzw. Produkte durchaus im Voraus geplant werden. Agile Vorgehensweisen beruhen in ihren Prinzipien auf emergenter Produktentwicklung und fördern durch ihre iterativ-inkrementellen Praktiken die Verminderung der Komplexität.

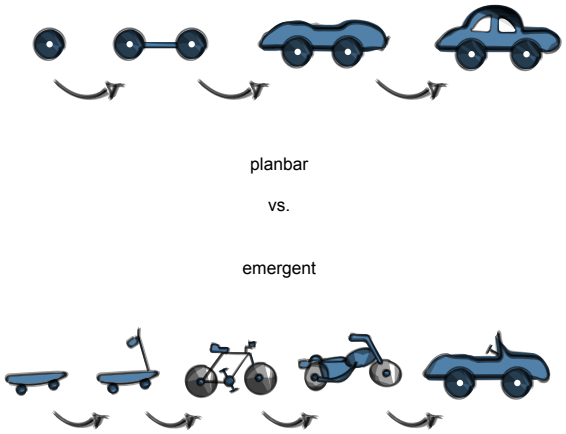


Abbildung 3: Planbare versus emergente Produktentwicklung

Fazit

Beide Ansätze – die klassischen und die agilen – haben ihre jeweiligen Vorteile. Letztendlich beruht die Einschätzung, welches Vorgehensmodell das geeignete ist, auf der Erfahrung beider Partner – den Auftraggeber*innen und den Umsetzungspartner*innen. Dennoch können folgende Grundregeln für eine Entscheidungsfindung herangezogen werden:

Projekte, die in einer kurzen Zeit mit kleinen Teams realisiert werden können und sich in einem klar definierten Kontext bewegen, sind ggf. geeignet für klassische Vorgehensmodelle. Auch bei einer klassischen Vorgehensweise arbeitet die RISC Software GmbH nach den agilen Werten und ergänzt üblicherweise die Projektprozesse durch agile Praktiken. Vor allem rege und kontinuierliche Kommunikation mit den Kund*innen ist ein Grundsatz, der nie vernachlässigt wird.

Für alle anderen Projekte empfehlen sich agile Vorgehensweisen. Die Lenkung der Produktentwicklung und die Verminderung der Komplexität sind elementare Prinzipien der agilen Werte und auch langfristige Projekte mit mehreren Teams können durch Skalierungstechniken koordiniert werden. Ein Spezialist*innenteam der RISC Software GmbH berät ihre Kund*innen bereits bei der Projektanbahnung individuell bzgl. der möglichen Vorgehensmodelle und verfolgt auch noch laufend während der Projektumsetzung die Ergebnisse um rechtzeitig ggf. notwendige Anpassungen am Vorgehensmodell mit den Kund*innen vornehmen zu können. ◆

Vertrauen in die Künstliche Intelligenz

– Christina Hess, MSc
Data Scientist in der Abteilung Logistics Informatics

Wie wir vertrauenswürdige KI-Systeme schaffen und nutzen

Künstliche Intelligenz (KI) unterstützt uns in unserem Alltag, bewusst oder unbewusst, bereits in vielen Bereichen. Wir agieren mit automatisierten Assistenten wie beispielsweise Sprachassistenten oder der Einparkhilfe im Auto, verwenden Gesichtserkennung beim Entsperren unserer Smartphones und lassen uns Musik und Filme vorschlagen. KI ist in unserem alltäglichen Leben schon sehr präsent – vielleicht sogar mehr, als uns oft bewusst ist. Die Anzahl der KI-Anwendungen wird auch in Zukunft noch weiter steigen. Wenn wir KI-Systeme erschaffen und verwenden, müssen wir uns in Zukunft nicht nur über die Chancen, sondern auch über die Risiken und die Herausforderungen bei deren Einsatz bewusst sein. Guidelines können uns dabei helfen, KI-Systeme zu erschaffen, die nicht nur effizient, sondern auch ethisch korrekt, sicher und vertrauenswürdig sind.

KI auf dem Vormarsch

Die Vorteile und Einsatzgebiete von KI sind vielfältig. Anhand einiger einfacher Beispiele ist dies leicht nachzuvollziehen: Recommendation-Systeme versuchen, aus unserer bisherigen Auswahl unseren Geschmack zu lernen und uns dann Inhalte vorzuschlagen, die uns gefallen. Streaming-Anbieter wie Spotify oder Netflix setzen seit Jahren auf diese Art von KI. Methoden des Natural Language Processing (NLP) helfen uns, Texte zu verstehen oder zu generieren. Dadurch wird zum Beispiel eine automatisierte Übersetzung von Texten in so hoher Qualität möglich, dass diese Texte oftmals nicht mehr von von Menschen geschriebenen Texten zu unterscheiden sind.

In der Industrie können Maschinen durch die Anwendung von KI frühzeitig gewartet und dadurch Ausfälle reduziert oder gänzlich verhindert werden. Bei solchen Anwendungen spricht man von Predictive Maintenance. Generell können durch Predictive Analytics Ausfälle von Maschinen, die Qualität von hergestellten Produkten oder auch Bestellungen bzw. Bestellmengen vorhergesagt werden. Bei Prescriptive Analytics – als Erweiterung zu Predictive Analytics ein aktuelles Forschungsgebiet – versucht man, auf Basis von Vorhersagen, kausalen Zusammenhängen und Domänenwissen optimale Entscheidungen zu treffen.

In der Medizin können KI-Systeme ärztliches Fachpersonal sowohl in der Diagnose als auch in der Behandlung unterstützen. Beispielsweise können auf Basis von Bilddaten Tumore oder Krankheiten automatisiert erkannt, auf Basis von Patientendaten und Symptomen Diagnosen vorgeschlagen und Krankheitsverläufe analysiert und prognostiziert werden.

KI-Systeme im Auto ermöglichen eine automatische Verkehrszeichenerkennung, helfen die Spur zu halten und ermöglichen sogar autonomes Fahren.

Aktuelle Herausforderungen und Probleme

Die Verwendung von KI-Systemen bringt allerdings auch einige Nachteile und Risiken mit sich. Systeme, die nicht ausgereift sind oder zu wenig getestet wurden, können fehleranfällig sein. Ist die vom Streaminganbieter zum User Profiling eingesetzte KI nicht gut, bekommt man als Anwender Inhalte vorgeschlagen, die einen nicht interessieren bzw. die man nicht sehen will. Werden Qualitäten oder Ausfälle falsch prognostiziert, kann das zu Ausfällen, Umsatzeinbußen und erhöhten Kosten führen. Bei der Anwendung von KI in sensibleren Bereichen kann es zu noch tragischeren negativen Auswirkungen kommen. Man denke beispielsweise an ein autonom fahrendes Fahrzeug, dessen KI schlecht darin ist, Personen und Objekte zu unterscheiden. Dann wird im schlimmsten Fall nicht der Person ausgewichen und die Mülltonne angefahren, sondern umgekehrt. Neben einer möglichen Fehleranfälligkeit von KI-Systemen sind auch andere Aspekte zu betrachten. Aktuell werden diesbezüglich vor allem ethische Aspekte, Datenschutz und die Transparenz von KI-Systemen thematisiert und diskutiert.

Neben einer möglichen Fehleranfälligkeit von KI-Systemen sind auch andere Aspekte zu betrachten. Aktuell werden diesbezüglich vor allem ethische Aspekte, Datenschutz und die Transparenz von KI-Systemen thematisiert und diskutiert. Fehler und Ungenauigkeiten in den verwendeten Daten, der Implementierung oder der Interpretation der Ergebnisse können dazu führen, dass Personen oder Gruppen diskriminiert werden, sie in ihrer Entscheidungsfähigkeit eingeschränkt werden oder das System auf andere Weise Schaden anrichtet. Ein bekanntes Beispiel ist das folgende: Bias (Verzerrungen) in den Daten führt dazu, dass auch auf den Daten trainierte Modelle diesen Bias abbilden. Ein wirklich gutes Modell lernt schließlich, so zu entscheiden, wie es der Mensch würde. Sind nun die dem Modell zugrundeliegenden Daten rassistisch, d.h. bilden diese Daten von Menschen getroffene, rassistische Entscheidungen ab, wird auch das Modell rassistisch entscheiden. Ein wirklich gutes Modell wie wir es uns als Gesellschaft wünschen würden – eines, das sicher und vertrauenswürdig ist – würde den in den

Daten vorhandenen Bias nicht mitlernen bzw. uns eine Möglichkeit bieten, dies zu verhindern.

Neben der Datenqualität stellt oft auch die Interpretierbarkeit von KI-Modellen eine große Herausforderung dar. Bei Black Box-Modellen kann die Entscheidung des KI-Systems nicht oder nur sehr ungenau nachvollzogen werden. Dies wäre aber hinsichtlich Transparenz und Vertrauenswürdigkeit des Systems äußerst wichtig. Stellen Sie sich vor, eine KI entscheidet darüber, ob sie einen Kredit bekommen oder nicht. Im Falle einer Ablehnung möchten Sie nun wissen, warum Sie so beurteilt wurden, wie Sie beurteilt wurden. Sie haben sogar das Recht zu erfahren, warum eine KI welche Entscheidung über Sie getroffen hat. Hier wird ein großes aktuelles Problem augenscheinlich: Wenn man die Entscheidungen einer KI nachvollziehen können möchte, kann man keine Modelle verwenden, deren Ergebnisse nicht restlos erklärbar sind – auch wenn diese Modelle besser in der Beurteilung sind als andere.

Davon unabhängig, ob eine KI gute Entscheidungen über einen trifft und ob diese interpretierbar sind: Vielleicht möchte man als Mensch einfach nicht von einer KI beurteilt werden. Ein in diesem Rahmen oft diskutiertes Thema ist das sogenannte Social Profiling. Um das Beispiel der Kreditvergabe erneut zur Illustration heranzuziehen: Teilt eine KI die Anwerber*innen automatisch in Kohorten ein, kann es sein, dass Ihnen aufgrund Ihres Alters oder Ihrer Herkunft ein Kredit verwehrt bleibt, der vergleichbaren KandidatInnen aber gegeben wird. Mit solchen ethischen Herausforderungen beschäftigen sich diverse Institutionen, Forschungseinrichtungen und NGOs. Beispielsweise legt Amnesty International hier anhand von Beispielen dar, inwiefern KI-Systeme eine Gefahr für unsere Grundrechte darstellen können.

Guidelines zur Schaffung von vertrauenswürdigen KI-Systemen

Die vorher genannten Fragestellungen werden von verschiedenen Organisationen/Institutionen und aus verschiedenen Richtungen bearbeitet. Um KI-Systeme so entwickeln zu können, dass sie robust und transparent sind und keine Gefahr für uns darstellen, sondern uns unterstützen und uns überall da helfen, wo wir als Menschen an unsere Grenzen stoßen, wurden bereits verschiedene Guidelines entwickelt. Das Ziel ist dabei, dass wir als Gesellschaft der KI und ihren Entscheidungen vertrauen können. Ein bekanntes Beispiel sind die im April 2019 von der EU veröffentlichten Ethik-Leitlinien für eine vertrauenswürdige KI. Nach diesen sollten KI-Systeme unter der Einhaltung verschiedener ethischer Grundsätze entwickelt und eingesetzt werden. Neben Fairness und Schadensverhütung (KI-Systeme sollen keinen Schaden verursachen oder sich anderweitig negativ auswirken) werden im Kontext von KI-Systemen die ethischen Grundsätze der Achtung der menschlichen Autonomie und der Erklärbarkeit beschrieben. Die Erklärbarkeit umfasst in diesem Zusammenhang nicht nur die Nachvollziehbarkeit der generierten Ergebnisse (Stichwort Black Box-Modelle), sondern auch Transparenz in Bezug auf alle Prozesse, die Fähigkeiten und den Zweck des KI-Systems. Um die menschliche Autonomie zu wahren, muss die menschliche Aufsicht und Kontrolle über das KI-System und alle Prozesse sichergestellt sein. KI-Systeme sollen demnach Menschen nicht unterordnen, nötigen, konditionieren oder ähnliches, sondern die Fähigkeiten des Menschen unterstützen und ergänzen.

In den Guidelines werden verschiedene Anforderungen an KI-Systeme gestellt, um die ethischen Grundsätze zu wahren und Vertrauen zu schaffen, und verschiedene technische und nicht-tech-

nische Methoden zur Umsetzung vorgeschlagen. Im Folgenden sind alle Anforderungen an vertrauenswürdige KI aufgelistet.

- Vorrang menschlichen Handelns und menschlicher Aufsicht: KI-Systeme sollen die menschliche Autonomie und Entscheidungsfindung unterstützen und durch den Menschen beaufsichtigt werden können. Je weniger Aufsicht ein Mensch über ein KI-System ausüben kann, desto ausführlicher muss es zuvor getestet werden und desto strenger muss die Lenkung und Kontrolle sein.
- Technische Robustheit und Sicherheit: Damit ein System als robust gilt, muss es einwandfrei funktionieren und die Ergebnisse präzise und reproduzierbar sein. Die Sicherheit schließt die Risikoprävention, das Erstellen von Auffangplänen und den Schutz vor Angriffen und Zweckentfremdung ein.
- Schutz der Privatsphäre und Datenqualitätsmanagement: Benutzerdaten müssen geschützt werden und dürfen nicht dazu verwendet werden, Benutzer*innen zu diskriminieren. Das Datenqualitätsmanagement ist zur Sicherstellung der Qualität der Daten, der Dokumentation und zur Kontrolle über die Zugriffe nötig.
- Transparenz: Nur wenn ein System und dessen Ergebnisse rückverfolgbar, d.h. nachprüfbar und erklärbar, sind und die Fähigkeiten und Einschränkungen des Systems klar kommuniziert werden, ist das System voll transparent.
- Vielfalt, Nichtdiskriminierung und Fairness: Alle Interessenträger müssen während des gesamten Lebenszyklus des KI-Systems eingebunden und gleichbehandelt werden. Wichtig ist dabei, dass Verzerrungen (Stichwort Bias) vermieden werden und das System möglichst barrierefrei und benutzerorientiert gestaltet ist.
- Gesellschaftliches und ökologisches Wohlergehen: Eine KI soll zum Wohle aller Menschen eingesetzt werden und während ihres gesamten Lebenszyklus sollen die breitere Gesellschaft, andere fühlende Wesen und die Umwelt als Akteure berücksichtigt werden. Dies schließt vor allem Nachhaltigkeit, Umweltfreundlichkeit, soziales Bewusstsein und die Wahrung von Demokratie ein.
- Rechenschaftspflicht: Zur Gewährleistung der Verantwortlichkeit und Rechenschaftspflicht für KI-Systeme und deren Ergebnisse müssen Vorkehrungen getroffen werden. Schutzbedürftige Personen müssen besonders berücksichtigt werden und im Falle, dass es zu nachteiligen Auswirkungen kommt, muss ein angemessener Rechtsschutz gegeben sein.

Fazit

Für die Entwicklung von guten und sicheren KI-Systemen wird es in Zukunft noch wichtiger, dass es keine negativen Auswirkungen gibt und dass alle Entscheidungen auch für uns Menschen nachvollziehbar sind. Andernfalls schenken die Menschen der KI zu wenig Vertrauen. Und fehlt das Vertrauen, fehlt auch die Akzeptanz solcher intelligenten Systeme in unserer Gesellschaft. Es ist besonders wichtig, dass sowohl Entwickler*innen als auch Anwender*innen die Chancen von KI-Systemen, aber auch deren Risiken und Grenzen kennen und verstehen. Dadurch und durch die Einhaltung von Guidelines, Achtsamkeit in der Entwicklung und durch die Verwendung von menschenzentrierten Entwicklungs- und Gestaltungsgrundsätzen können wir zukunftsfitte KI-Systeme schaffen. ◆

Software-Reengineering: Wann wird das Alt-System zum Problem?

– DI (FH) Alexander Leutgeb
Head of Unit Industrial Software Applications

If ain't broke, don't fix it

Unternehmenskritische Software ist nicht vor einem gewissen Alterungsprozess gefeit. Ein gleichwertiger Ersatz ist aber oft nicht so einfach verfügbar. Wann aber ist es an der Zeit um das Legacy-System mithilfe von Software-Reengineering abzulösen? Eine weit verbreitete Weisheit lautet: „Never change a running system". Dieser Spruch, bei dem es sich vermutlich um eine abgewandelte Form der Aussage „Never change a winning team" des britischen Fußballspielers und -trainers Sir Alf Ramsey handelt, ist allerdings im englischen Sprachraum kaum verbreitet. Dort wird stattdessen die Aussage „If it ain't broke, don't fix it." verwendet. Die Aussagen sind ähnlich, jedoch wirkt die zweite Formulierung nicht ganz so streng und dogmatisch. Man soll also nur versuchen etwas zu reparieren, wenn es tatsächlich defekt oder beschädigt ist.

Aus funktionaler Sicht würde das bedeuten, ein System ist nicht mehr in der Lage die Aufgabe zu erfüllen, für das es konzeptioniert wurde. Für Software kann man hier auch noch einen Schritt weiter gehen. Ein Softwaresystem ist „reparaturbedürftig" wenn es in einen Zustand kommt, der die Wartung schwierig bis unmöglich macht. Wir haben für Sie sechs Anzeichen gesammelt, die darauf hindeuten, dass Handlungsbedarf für Ihr Softwaresystem besteht:

Fehlende oder überholte Dokumentation

Ist die Dokumentation eines Systems in vielen Teilen veraltet oder stimmt diese in weiten Teilen nicht mehr mit dem tatsächlichen System überein, ist das ein deutliches Zeichen eines Legacy Systems an dem bereits viele Änderungen durchgeführt wurden. Noch schlimmer ist das Fehlen einer umfangreichen Dokumentation. Solch ein Umstand erschwert die Wartung und Weiterentwicklung eines Systems und macht ein zügiges Einschulen neuer Mitarbeiter*innen fast gänzlich unmöglich.

Ursprüngliche Entwickler haben das Unternehmen verlassen

Üblicherweise ist sehr viel Wissen über ein komplexes System in den Köpfen weniger Mitarbeiter*innen gespeichert. Dies führt dazu, dass der Fortbestand der Software unmittelbar von den ursprünglichen Entwickler*innen abhängt, insbesondere wenn die Dokumentation nicht adäquat ist. Der kontinuierliche Abgang dieser Mitarbeiter*innen führt unweigerlich zu einer Situation, die eine Weiterentwicklung des Systems unmöglich macht.

Die Wissensbasis für das System fehlt

Ein deutliches Alarmsignal ist, wenn kaum mehr Verständnis für die grundlegende Funktionsweise der Software bei den Mitarbeiter*innen vorhanden ist. Im schlimmsten Fall kennt niemand im Unternehmen mehr die ursprünglichen Zusammenhänge und Konzepte. Damit wird es extrem schwierig im Fall eines auftretenden Problems nachhaltige Lösungen zu finden. Das Ergebnis sind oft schnelle „Hacks" um das Problem auf irgendeine Weise zu lösen, diese beschleunigen den Teufelskreis jedoch noch mehr.

Selbst die Umsetzung von kleinen Änderungen ist sehr aufwändig

Ein von Lehman und Belady formuliertes „Gesetz" zur Evolution von Software [1] besagt, dass Systeme immer komplexer werden, bis aktiv daran gearbeitet wird, um die Komplexität zu reduzieren. Erfordert selbst die Umsetzung von kleinen Änderungen oder Erweiterungen großen Aufwand, ist dies ein eindeutiges Zeichen dafür, dass das System bereits zu komplex geworden ist. Wenn bereits kleine Änderungen einen großen Zeitaufwand erfordern, werden schwierigere Änderungen kaum umsetzbar.

Ständige Notwendigkeit Fehler zu beheben

Kein umfangreiches Softwareprojekt wird frei von Fehlern sein, wodurch regelmäßige „Bugfixes" zur Notwendigkeit werden. Führt jedoch immer wieder das Beheben eines Fehlers zum Auftreten eines neuen Fehlers, ist dies ein Anzeichen dafür, dass die Architektur des Systems möglicherweise nicht mehr den aktuellen Anforderungen entspricht. Dies birgt die Gefahr, dass kleine Änderungen im Programm unvorhergesehene Auswirkungen zeigen.

Code Smells

Als „code smells" (übelriechender Code) werden Teile des Sourcecodes bezeichnet, die zwar nicht fehlerhaft sind, aber schlecht strukturiert und implementiert sind. So hat etwa die Duplizierung von Sourcecode, also die Verwendung derselben Codezeilen an verschiedenen Stellen des Programms, einen sehr schlechten Einfluss auf die spätere Wartbarkeit des Systems.

[1] M. M. Lehman, "Programs, life cycles, and laws of software evolution," in Proceedings of the IEEE, vol. 68, no. 9, pp. 1060-1076, Sept. 1980 vgl. https://ieeexplore.ieee.org/document/1456074

Die Lösung: Software Reengineering

Werden die „Symptome“ rechtzeitig erkannt, so kann mittels geeigneter Methoden dem Verfall der Software entgegengewirkt werden, um die Qualität des Systems auf hohem Niveau zu halten und mögliche Ausfälle zu vermeiden. Durch Reengineering wird ein bestehendes System neu strukturiert aber vor allem auch für zukünftige Entwicklung und Erweiterung vorbereitet.

Während es bei der Wartung darum geht Software, die sich in einer Produktivumgebung befindet an geänderte Umgebungsbedingungen anzupassen und erkannte Fehler auszubessern, werden durch Reengineering Systeme oder deren Teile von Grund auf neu konzeptioniert und umgesetzt.

Die RISC Software GmbH unterstützt Sie gerne bei der Erstanalyse sowie bei der Umsetzung eines Reengineerings Ihres Softwaresystems. ◆

Arbeiten mit Fortran in 2020: Einsatzgebiete

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

Fortran – die bewährte Programmiersprache für technische Anwendungen

Der Name Fortran setzt sich zusammen aus "FORmel TRANslator" und ist eine der ältesten Programmiersprachen. Für viele Softwareentwickler*innen ist sie 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, Stabilitätsberechnungen, uvm.

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, FORTRAN 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 kleines Update der Sprache. Die aktuellste Version des Fortran Standards ist Fortran 2018, wobei noch kein Compilerhersteller alle Features unterstützt.

Compiler: die Qual der Wahl

Neben dem Umfang der Programmiersprache an sich ist für die Programmierer*innen auch ein einfacher und angenehmer Workflow während der Entwicklung wichtig, heißt in welcher Entwicklungsumgebung kann man mit welchem Compiler den geschriebenen Code gut testen und debuggen. Die eingeschränkte Anzahl an verfügbaren Entwicklungsumgebungen verstärkt den Eindruck, dass Fortran heutzutage nicht mehr zu den gängigsten Sprachen gehört. Um modernes Fortran zu programmieren (Fortran 2003/2008) hat man mehrere Compiler zur Auswahl, sowohl Open Source als auch kommerzielle, unter anderem

- Intel Fortran (ifort, kommerziell)
- NAG Fortran (kommerziell)
- GNU Fortran (gfortran,Open Source)
- Flang (ehemals „f18", in Entwicklung, Open Source)

Hinsichtlich Entwicklungsumgebungen gibt es speziell für Fortran entwickelte IDEs, wie etwa den NAG Fortran Builder, oder auch Integrationen für bestehende IDEs, wie die Intel Integration für Visual Studio, oder Erweiterungen für Editoren, wie etwa Visual Studio Code. Neben den mitgelieferten Debuggern (gdb, idb) sind auch externe kommerzielle Debugger, wie Total View für Fortran erhältlich. (https://totalview.io/)

Warum wird Fortran heutzutage noch eingesetzt? Legacy Code

Viele Fortran Systeme die heute noch in Betrieb sind wurden in den 1980er entwickelt. Zu dieser Zeit war Fortran die Sprache für wissenschaftliches Rechnen und numerische Simulationen. In solchen Programmen steckt sehr viel Wissen und Erfahrung, oftmals von mehreren Generationen von Ingenieur*innen und Wissenschaftler*innen. Ein Übertragen in andere „moderne" Programmiersprachen ist oft viel zu aufwendig oder zu teuer und der Mehrwert für die Unternehmen wäre kaum gegeben. Ein Reengineering des Programmcodes in einem modernen Fortran Standard und das aktive Weiterentwickeln in diesem ist oftmals die bessere Alternative.

Fortran ist für technische Anwendungen konzipiert

Wie bereits zu Beginn des Artikels erwähnt sind numerische Simulationen der typische Use-Case für Fortran als Programmiersprache. Diese Bereiche zeichnen sich dadurch aus, dass einerseits effiziente Programmausführung wichtig ist, aber auch die Berechnungen durch mathematische Formeln beschrieben sind. Fortran gibt den Programmierer*innen die Möglichkeit diese Formeln in lesbarer Weise kompakt im Programmcode zu repräsentieren. Weiters ist die Sprache ausgelegt für effiziente Berechnungen, wodurch hoch performanter Code eher die Regel als die Ausnahme ist.

Fortran ist eine einfache Sprache

Verglichen mit C++ oder Skriptsprachen wie Python hat Fortran ein eher eingeschränktes Feature Set, weshalb für viele heutige Geschäftsanwendungen die Entwicklung sehr mühsam wäre, für technisch/naturwissenschaftliche Anwendungen ist sie aber sehr wohl passend. Das kleinere Feature Set erlaubt es auch weniger erfahrenen Programmierer*innen gute Software zu entwickeln. Gerade durch die vielen Möglichkeiten von C++20 aber auch wie von Python sind weniger erfahrene Programmierer*innen mit den Möglichkeiten oft überfordert. Falsch angewendet kreieren komplexere Sprachen oftmals mehr Verwirrung als sie zur Lösung des Problems beitragen. In diesem Sinne wurde Fortran für Wissenschaftler*innen und Ingenieur*innen, nicht für Informatiker*innen und Softwareentwickler*innen konzipiert.

Integration Fortran/C/Python

Mit Fortran 2003 wurde mit dem Modul iso_c_binding neue Möglichkeit zur einfacheren Interaktion zwischen Fortran und C Code geschaffen. Dieses Modul inkludiert:

- Arten von primitiven Datentypen (Integer, Reals), die den entsprechenden C-Datentypen entsprechen (int, double, ..)
- Funktionen zum Arbeiten mit C-Pointern, wie die Konvertierung von C zu Fortran Pointer oder Auslesen der Speicheradresse einer Fortran Variable.
- Konstanten für nicht anzeigbare C-Zeichen, wie das Zeilenumbruchzeichen (\n) oder das horizontale Tab (\t).

Nicht alle Datentypen aus Fortran kann man mittels dem iso_c_binding in den C-Code übertragen und vice - versa. Beispielsweise existiert in Fortran kein Äquivalent einer C-Union oder in C gibt es kein direktes Äquivalent des Fortran ALLOCATABLE Attribut für Arrays, das eine dynamische Allokation des Speichers erlaubt, der automatisch wieder freigegeben wird, falls der Array out-of-scope geht. Weiters hat C/C++ keine Unterstützung für Array-slices, also eine strukturierte Teilmenge eines Arrays, daher müssen bei der Übergabe dieser Kopien angelegt werden.

In einigen Projekten der RISC Software GmbH war die Vorgabe Python Interfaces für Fortran Code zu schreiben. Dazu hat sich das Programm SWIG (http://www.swig.org/) bewährt, das Code für Python/C Interfaces generiert. Um ein Python/Fortran Interface zu erzeugen, wird zuerst die Fortran/C Schnittstelle geschrieben, um mittels SWIG dann die Python/C Schnittstelle zu erzeugen. SWIG ist in seiner Anwendung sehr flexibel und lässt sogar numpy-Arrays als Argumente der Python Funktionen zu, was zu einerseits effizienten und anderseits gut lesbaren Python führt. Durch die Einbindung in das CMake Buildsystem funktioniert dieser Zugang plattformunabhängig und bietet einen einfachen und effizienten Workflow in der Entwicklung. ◆

Schätzen in agilen Projekten mit Story Points

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

Aufwandsabschätzung von agilen Softwareprojekten

Um den Umfang eines Projektes für Angebot und auch Umsetzung vorab einzuschätzen, werden herkömmlicherweise Personenstunden oder -tage verwendet. Dies birgt einige Probleme und Gefahren mit sich. Komplexe Aufgabenstellungen sind vor Projektbeginn oftmals nicht abschätzbar, zu großzügige Puffer oder kurze Zeitfenster lassen somit Angebote oft ungenau werden.

Eine der größten Herausforderungen in der Entwicklung von Individualsoftware liegt in einer doch einigermaßen einfach klingenden Aufgabe: der Aufwandsabschätzung. Der Begriff beinhaltet eine „Schätzung", was grundsätzlich den empirischen Charakter dieser Aufgabe aufzeigt. Dennoch sind insbesondere in der Anbahnungsphase möglichst „genaue Schätzungen" gefragt und teilweise werden diese sogar als Vertragsbasis herangezogen.

Alles ist relativ

Anders ist die Schätzung mit Story Points: sie basiert nicht auf absoluten Werten sondern auf relativen. Story Points können auf unterschiedliche Werte, welche die Komplexität einer Aufgabenstellung festlegen, angesetzt werden. Am gängigsten ist jedoch eine allgemeine Bewertung des Umfangs oder Aufwands. Eine direkte Konvertierung in eine konkrete Zeitaussage ist vorerst nicht möglich.

Mit Story Points werden Aufgaben von den Mitgliedern des Entwicklungsteams in Relation zueinander geschätzt. Exemplarisch wird von einem Team eine bekannte Aufgabe definiert, welche mit der dazugehörigen Schätzung (z. B. 3 Story Points) einen Ausgangswert für weitere Schätzungen bietet. Jede künftige Aufgabe wird nun in Relation zu dieser Messgröße vom gesamten Entwicklungsteam geschätzt. Menschen fällt es meist sehr leicht einen Aufwand in „größer als" oder „kleiner als" einzuschätzen, weshalb an dieser Stelle ein erster Vorteil der Story Points besteht. Weiters werden Story Points in einer angepassten Fibonacci-Folge verwendet, welche die Werte 0, 1, 2, 3, 5, 8, 13, 20, 40, 100 und ∞ beinhaltet. Diese Skala erlaubt eine Schätzung in größer werdenden Bereichen, welche hier vor allem noch nicht genau spezifizierte oder zu große Aufgaben erkennbar macht. Es ist nicht notwendig diese Skala in ihrer Gesamtheit auszunutzen. Durchgeführt wird eine solche Schätzung vom Entwicklungsteam z. B. mit Karten im „Planning Poker".

Unterschiedliche Geschwindigkeiten miteinberechnen

Um Story Points für agile Softwareentwicklung nutzen zu können, wird ein weiterer Wert benötigt: die Teamgeschwindigkeit (Velocity). Die Teamgeschwindigkeit gibt an, wie viele Story Points das Team durchschnittlich je Iteration leisten kann. Dieser Wert ist variabel und ändert sich üblicherweise über die Dauer eines Projekts, weshalb kontinuierliche Beobachtungen und Anpassungen notwendig sind. Vor allem die Stabilität der Teamzusammensetzung ist hierfür eine essentielle Voraussetzung. Die Begründung liegt darin, dass die Schätzung nicht nur relativ im Bezug auf die Aufgaben, sondern auch unabhängig vom Leistungspotential der einzelnen Teammitglieder geschieht. Eine Aufgabe mit 3 Story Points kann für ein Teammitglied einen tatsächlichen Zeitaufwand von 5 Stunden bedeuten, für ein anderes Teammitglied nur drei Stunden. Da Story Points als Team vergeben und die Aufgaben auch als Team mit bekannter Teamgeschwindigkeit bearbeitet werden, ergibt sich eine gute Vorhersagbarkeit über den möglichen Umsetzungsumfang einer Iteration.

Werden für die Teamgeschwindigkeit drei Werte herangezogen, nämlich die durchschnittliche, die schnellste und die langsamste der letzten sechs Iterationen, so kann für die Vorhersagbarkeit der Umsetzungsdauer ein Korridor angegeben werden, welche in Hinsicht auf eine Planung weitere Vorteile bietet. So können damit Aufgaben in einem Backlog einfach identifiziert werden, welche gemäß der aktuellen Planung zeitlich nicht möglich sind oder welche garantiert werden können.

| Win-win-Situation:

Das Schätzen mit Story Points bietet somit den Kund*innen eine bestmögliche Transparenz für die Planung, Kontrolle und Anpassung einer Projektroadmap. Somit können trotz agiler Vorgehensweise zuverlässige Aussagen über Machbarkeit, Fertigstellung, etc. gemacht werden. Auch für das Entwicklungsteam ist eine Planbarkeit bei gleichzeitiger Agilität gewährleistet. Die RISC Software GmbH setzt Story Points seit längerem in den agilen Projekten erfolgreich ein. ◆

Kann Data Science Industriebetriebe aus der Krise führen?

– Mag.ª Stefanie Kritzinger, PhD
Head of Unit Logistics Informatics

Wie es möglich ist, Kosten zu minimieren, auf Nachfrageschwankungen flexibel zu reagieren und Produktionsstillstände durch Störung zu vermeiden.

Gerade in wirtschaftlich schwierigen Zeiten spielt Digitalisierung und die damit einhergehende Automatisierung eine entscheidende Rolle. Produzierende Unternehmen stehen vor einer noch nie dagewesenen Herausforderung: Prozesse sollen bereits digitalisiert und teilautomatisiert steuerbar sein, um die Produktion aus der Ferne zu überwachen und zu leiten. Absatzmärkte und Personalplanung unterliegen äußeren, unkontrollierbaren Einflüssen und Produktion in kleinen Losgrößen ist attraktiver als je zuvor. Je nach Branche und Digitalisierungsgrad können einige Unternehmen damit problemlos umgehen, andere nicht.
Die enorme Bedeutung der digitalen und virtuellen Vernetzung wird uns aktuell bei der Bekämpfung der Pandemie vor Augen geführt. Dank der Digitalisierungsbestrebungen der letzten Jahre wurden Prozess- und Produktionsdaten zunehmend als wesentlicher Teil der Wertschöpfung angesehen. Wer seine Hausaufgaben bereits gemacht hat, sammelt seit einiger Zeit ein umfangreich und automatisiert Daten der eigenen Unternehmensprozesse. Zum einen, um damit Echtzeitinformationen für die Reaktion auf kurzfristige Änderungen in der Produktion zu analysieren und zu verarbeiten. Zum anderen, um aus den gesammelten Datenpools zukünftige Ereignisse ableiten und möglichst genau prognostizieren zu können.

Prozesse: Qualität steigern und Kosten minimieren

Um die Qualität zu steigern und die Kosten zu minimieren, liegt ein wichtiger Erfolgsfaktor darin, aus den gesammelten Produktionsdaten wertvolle Informationen zu gewinnen. Aber dies ist keine triviale Aufgabe. Durch intensives Data Engineering werden Prozess- und Produktionsdaten über die qualitätsrelevanten Prozessschritte nutzbar. Die notwendigen Parameter werden identifiziert, um automatisierte Datenanalysen mittels Data und Visual Analytics oder auch moderner Methoden der Künstlichen Intelligenz durchzuführen. Damit können Auffälligkeiten erkannt, richtig bewertet und die Auswirkungen auf die finale Produktqualität prognostizierbar werden. Ergänzt wird die Verbesserung des Qualitätsmanagements durch die Rückverfolgbarkeit der Produktionsparameter und Qualitätsmerkmale des gesamten Prozesses. So ist ein besseres Verständnis des Produktionsprozesses durch das Erkennen von Ursache-Wirkung-Zusammenhängen aufgrund von Anomalien und Mustern möglich. So können auch Wartungsintervalle und -zyklen optimiert und in weiterer Folge Produktionsabläufe verbessert werden.

Damit gelingt es beispielsweise, den Ausschuss zu minimieren, indem die Maschinen mit der richtigen Betriebstemperatur produzieren, ungeplante Stillstandzeiten minimiert oder die Wartungsintervalle optimiert werden.

Bottlenecks frühzeitig erkennen: Prescriptive Analytics

Gerade in Zeiten der Krise sind kurzfristige Nachfrageschwankungen das tägliche Geschäft. Meist sind Produktionssysteme aufgrund ihrer individuellen Struktur und Organisation zudem hoch komplex. Knappe Ressourcen, gesonderte Wünsche von Kund*innen, die damit einhergehende Produktvielfalt sowie Termindruck überlasten vorhandene Kapazitäten und führen zu kostenintensiven Engpässen beispielsweise durch Personalaufstockung oder verspätete Lieferungen. Eine gute Vorbereitung zur frühzeitigen Erkennung von Bottlenecks basiert auf einer intelligenten prognosegestützten Planung.

Auf Basis von historischen Produktionszahlen und weiteren Einflussparametern sowie mithilfe moderner Methoden aus dem Bereich Statistik und der Künstlichen Intelligenz können Produktionszahlen vorhergesagt werden. Aufgrund dieser mit hoher Wahrscheinlichkeit zutreffenden Prognosen können adäquate Maßnahmen abgeleitet und die zu erwartende Entwicklung positiv beeinflusst werden – dies wird unter dem Begriff Prescriptive Analytics subsummiert. Vorausschauende Analysefunktionen schaffen eine höhere Transparenz in der bevorstehenden Produktion. Gezielte Berechnungen und Visualisierungen verdeutlichen, wo Engpässe entstehen können und wo es anhand der Planung zu Verzögerungen kommen wird. Somit werden echte Erkenntnisse geliefert, die eine Intervention ermöglichen, bevor das Problem bei Kund*innen angelangt.

Stillstände vermeiden: Störungsmanagement

Fehlendes Material, mangelndes Personal und die sich in Folge einer Krise verändernden Anforderungen sind meist bekannte Faktoren für Produktionsstillstände. Störungen führen häufig zu Umsatzeinbußen, zu großen finanziellen Verlusten und wirken sich negativ auf das Betriebsergebnis aus. Wenig Aufmerksamkeit wird dabei der Bewältigung einer Störung von ihrer Entdeckung (oft als Discovery bezeichnet) bis zu ihrer vollständigen Wiederherstellung (oft als Recovery bezeichnet) gewidmet. Speziell bei unvorhergesehenen Ereignissen ist ein effizientes Störungsmanagement in der Lage, Störungen reaktionsschnell aufzufangen. Neuplanungen des Produktionsprozesses sind notwendig, um mit den eingeschränkten Ressourcen durch Zulieferengpässe oder Kurzarbeit die Liefertreue so weit als möglich einzuhalten und gleichzeitig die empfohlenen Maßnahmen einzuhalten.

Im aktuell sehr turbulenten und dynamischen Umfeld ist es notwendiger denn je, den Digitalisierungsgrad weiter zu erhöhen, um an den potentiellen Zielgrößen Steigerung der Qualität, Minimierung der Kosten, Erkennung von Bottlenecks und effizientes Störungsmanagement festhalten zu können. Zwei wesentliche Stellhebel sind dabei die Prozesstransparenz und die Reaktionsfähigkeit.

| Know-how

Die Data Engineers und Data Scientists der RISC Software GmbH verfügen über umfangreiche Kompetenzen und langjährige Erfahrung in unterschiedlichsten Bereichen des Data Managements und Data Analytics. Durch den Einsatz von modernen Methoden aus den Bereichen Data Analytics and Visual Analytics sowie Machine Learning zur smarten Datenanalyse und Prognose kann die Herausforderung von Big Data als wichtige Chance zur Prozess- und Umsatzoptimierung wahrgenommen werden. ◆

Modernisierung von Software

– Michael Hava MSc, DI (FH) Josef Jank MSc, und DI (FH) Alexander Leutgeb
Software Architects & Project Manager der Unit Industrial Software Applications

Inkrementelles Re-Engineering für eine nachhaltige Software-Entwicklung

Bei langlebigen Software-Systemen übersteigen die Wartungskosten die initialen Entwicklungskosten bei weitem. Entkommen Sie der Kostenfalle durch rechtzeitige pro-aktive Modernisierungsmaßnahmen. Ein evolutionäres Vorgehen garantiert planbare Kosten, kontinuierliche Releases und unmittelbaren Kundennutzen bei überschaubarem Risiko.

Raus aus den (technischen) Schulden

Untersuchungen zeigen, dass der Erfolg von Firmen zunehmend von Software bestimmt wird. Einer der erfolgreichster Händler (Amazon) ist daher nicht zufällig eine der erfolgreichsten Softwarefirmen, sondern hat damit die Grundlage für den Erfolg geschaffen. Insbesondere im industriellen Umfeld entstand im Zuge der Digitalisierung im Laufe der letzten Jahrzehnte zunehmender Bedarf im Bereich der Softwareentwicklung. Viele Firmen haben daher eigene Software-Entwicklungsteams etabliert. Diese bestanden anfangs allerdings oftmals aus reinen Fachdomänenexpert*innen ohne tiefergreifende Software-Entwicklungsexpertise. Da sich Software-Entwicklung immer mehr als Kernthema herauskristallisierte, wurde die Notwendigkeit der Software-Entwicklung als eigenständige Disziplin erkannt und die Teams um Software Ingenieure erweitert. Die lange Entwicklungshistorie, die heterogenen Teams und eine entsprechende Entwickler-Fluktuation führen zu einer Heterogenität hinsichtlich der Entwicklungsmethodik, der verwendeten Technologien und der Code-Qualität. Steuert man dieser Heterogenität nicht durch fortwährende Konsolidierung entgegen, steigen die Kosten für die Wartung der Software über die Jahre enorm an. Darüber hinaus sind Änderungen immer schwieriger und die Integration neuer Funktionalität ist nur mehr mit hohem Aufwand und Risiko möglich. Im schlimmsten Fall können so die technischen Schulden in einem „technischen Bankrott" führen (siehe Abbildung 4).

Durch eine gezielte Modernisierung kann man Kosten und Risiko für die Wartung stark reduzieren und auf zukünftige Anforderungen schneller reagieren. Durch eine inkrementelle Vorgehensweise können die Anpassungen kontinuierlich in den Produktivbetrieb übernommen werden und es wird sichergestellt, dass das System stets den Anforderungen genügt. Dabei lässt sich der Aufwand für die Modernisierung im Verhältnis zu einer Neuentwicklung relativ leicht und genau abschätzen. Ein weiterer Aspekt technischer Anwendungen ist oftmals, dass die Anforderungen hinsichtlich Modellgrößen über die Jahre stark gewachsen sind, sodass die Laufzeiten für Berechnungen und der Speicherbedarf mit der aktuellen Umsetzung der Software nicht mehr praktikabel sind. Im Zuge einer Modernisierung können solche Engpässe identifiziert und durch eine adäquate Softwareumsetzung unter Ausnutzung des Paralleliserungspotentials moderner Hardware-Architekturen behoben werden.

Die ideale Vorgehensweise ist abhängig von der Problemstellung

Die Grundlage für eine technisch und wirtschaftlich erfolgreiche Modernisierung von Legacy-Software ist eine gut überlegte Gesamtstrategie. Dabei ist es wichtig möglichst unvoreingenommen an das Thema heranzugehen und neben der Modernisierung und Sanierung auch radikale Ansätze wie eine vollständige Neuentwicklung oder den Einsatz von Standardsoftware zu berücksichtigen und zu bewerten. Bei der Frage, ob man existierende Software einem Reengineering unterzieht oder stattdessen eine Neuentwicklung projektiert, ist die erstgenannte Alternative häufig vorzuziehen, weil das Reverse-Engineering bis hin zu den Requirements enorm aufwändig sein kann (siehe Abbildung 5). Oftmals ist fast keine Dokumentation verfügbar und das Know-how nur mehr im Quellcode manifestiert. Aufgrund der substanziell geringeren Kosten, Risiken und Durchlaufzeit wird man sich daher in vielen Fällen für eine spezifische Modernisierung entscheiden, wo die notwendigen Re-Engineering Maßnahmen (Re-Code, Re-Design, Re-Specify, Re-Think) gezielt für die unterschiedlichen Teile des Systems identifiziert werden. Bei der Wahl der jeweiligen Maßnahme ist abzuwägen, ob der daraus resultierende Nutzen den Aufwand rechtfertigt.

Bei der Modernisierung sollten Teile mit hohem und niedrigem Änderungsbedarf identifiziert und strategisch unterschiedlich behandelt werden. Bei den stabilen Teilen mit geringem Änderungsbedarf sollte man immer kritisch hinterfragen, ob Aufwand und Risiko den potenziell geringen Nutzen rechtfertigen. Anti-Corruption Layer bieten die Möglichkeit einer schrittweisen Migration vom legacy zum neuen System, wobei zu jedem Zeitpunkt ein stabiles Produktivsystem zur Verfügung steht und die weitere Modernisierung nach Bedarf entschieden werden kann. Ein Anti-Corruption Layer isoliert Teile von einem Gesamtsystem und stellt die Kompatibilität zwischen unterschiedlich evolutionär entwickelten Teilen sicher (Legacy Anwendung mit neuen Code-Teilen, neue Anwendung mit Legacy Code-Teilen).

Eine wichtige Strategie im Zuge einer Modernisierung ist kontinuierliche Reduktion von Abhängigkeiten und die Verbesserung durch Modularisierung. Das Ziel sind lose gekoppelte Komponenten mit klar definierten Schnittstellen und nach Möglichkeit standardisierte Datenformate und Protokolle. Im Falle von hocheffizienten und

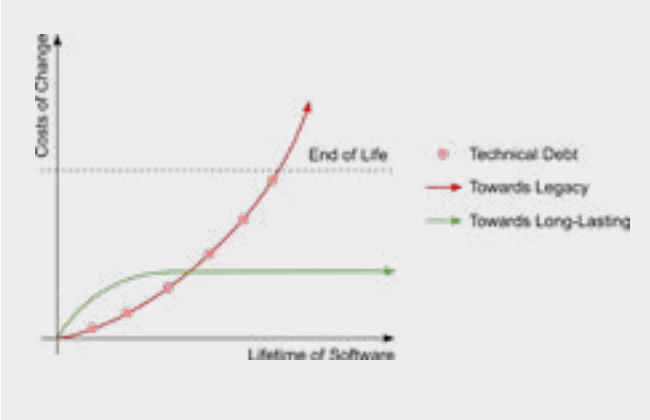


Abbildung 4: Vorzeitiges Lebensende von Software aufgrund technischer Schulden

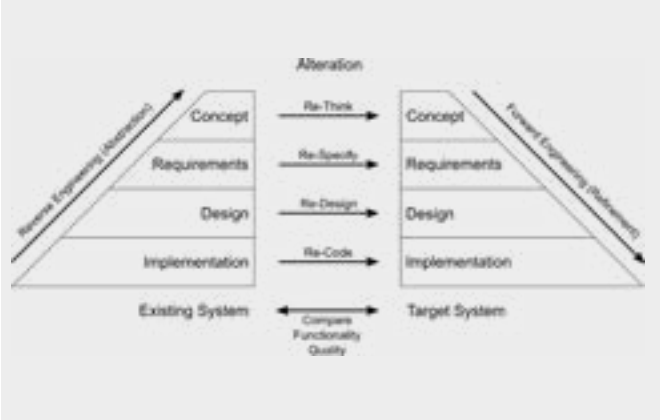


Abbildung 5: Kategorisierung von Re-Engineering Maßnahmen

spezialisierten Implementierungen in Fortran/C/C++ ermöglicht dies eine einfache Integration in Plattformen wie beispielsweise Python, .NET und Java in unterschiedlichen Anwendungsszenarien. Ein wesentliches Verbesserungspotential im Zuge der inkrementellen Modernisierung bietet der kontinuierliche Ausbau der Testautomatisierung. Damit wird einerseits das Verhalten des aktuellen Systems dokumentiert und andererseits dienen die Tests als „Sicherheitsnetz", um unbeabsichtigte Seiteneffekte und Fehler im Zuge von Änderungen automatisch erkennen zu können.

Im Zuge der konkreten Implementierung von Änderungen und Erweiterungen bieten sich zahlreiche Chancen, um von der Weiterentwicklung moderner Programmiersprachen und Bibliotheken zu profitieren und damit die Software wartungsfreundlicher, robuster und performanter zu gestalten. Gerade im Hinblick auf die Performance gab es in den letzten Jahren viele Verbesserungen (Parallelisierung etc.) von denen man häufig unmittelbar profitiert. Darüber hinaus können oft auch Teile der Eigenentwicklung ersetzt werden, weil diese entsprechende Funktionalität in der Zwischenzeit direkt unterstützt wird oder entsprechende Open Source Alternativen verfügbar sind. Technisch anspruchsvolle Aufgabenstellungen mit komplexen Berechnungen können damit schneller gelöst werden (Strong Scaling) bzw. man kann größere Problemstellungen lösen (Weak Scaling).

Zusätzlich zu der angeführten Modernisierung auf Produktebenen sollten natürlich auch gesamte Softwareentwicklungsprozess analysiert und gegebenenfalls verbessert / modernisiert werden. In den letzten Jahre wurde im Rahmen von Studien deutlich gezeigt, dass insbesondere die rasche Änderbarkeit (inkl. Rollout) von Software ein guter Indikator ist – nicht nur für die Leistungsfähigkeit der Softwareentwicklung, sondern aufgrund der zunehmenden Bedeutung von Software meist sogar für den gesamten wirtschaftlichen Erfolg eines Unternehmens. Die RISC Software GmbH ist langjährige Entwicklungspartnerin in großen industriellen Softwaresystemen, unterstützt und forciert daher auch die Modernisierung des Softwareentwicklungsprozesses bei ihren Partnerunternehmen.

Profitieren Sie von den Potenzialen von modernem C++

Die Entwicklung vieler großer Softwaresysteme startete im Laufe der 90er Jahre. Zu diesem Zeitpunkt gab es im Vergleich zu heute wesentlich weniger Programmiersprachen, OpenSource war in der Breite noch kaum ein Thema und die Anzahl der verfügbaren Bibliotheken war überschaubar. C++ war die Programmiersprache der Wahl für anspruchsvolle moderne (industrielle) Anwendungen. Zu diesem Zeitpunkt war der erste C++ Standard noch in Arbeit und existierende Implementierungen waren fragmentiert. Standardfunktionalität wie beispielsweise Container wurden individuell nach unterschiedlichen Designphilosophien entwickelt, weil moderne Alternativen wie die STL noch nicht bekannt bzw. verbreitet waren.

Während sich in den 2000er Jahren zahlreiche neue, moderne Programmiersprachen etablierten, schien die Entwicklung von C++ zu stagnieren. Nach der Veröffentlichung von C++ 98 begannen zwar die Arbeiten an der Nachfolgeversion – die Fertigstellung verzögerte sich allerdings bis 2011. Um weitere lange Phase der Stagnation zu verhindern, wechselte das C++ Komitee seinen Veröffentlichungsprozess und liefert alle 3 Jahre einen neuen Standard. Die Resonanz auf neue C++ Versionen ist überaus positiv – immer mehr Unternehmen beteiligen sich aktiv an der Weiterentwicklung dieser leistungsfähigen Sprache. Die RISC Software GmbH ist mit Michael Hava über das ASI (https://www.austrian-standards.at/) im Komitee vertreten.

Dem C++ Komitee ist es gelungen, die Lesbarkeit, Robustheit und Performance in weiten Bereichen wesentlich zu verbessern, ohne bestehenden Code zu brechen. Anwendungen profitieren daher ohne langwierigen/teuren Rewrite unmittelbar von den Verbesserungen und können inkrementell auf Modern C++ „portiert" werden. Während vorhandener Code nach wie vor funktioniert, bietet sich die Möglichkeit Quelltexte schrittweise lokal zu verbessern. Nachfolgend einige Beispiele um zu zeigen, was im Rahmen von lokalen Verbesserungen in Hinblick auf Kompaktheit, Ausdrucksstärke, Robustheit und Effizienz möglich ist.

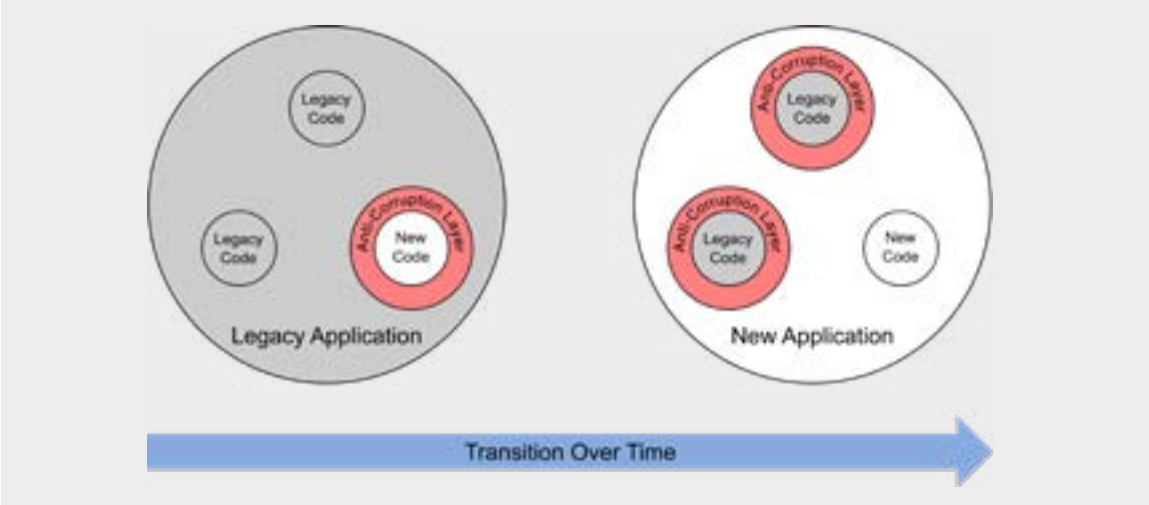


Abbildung 6: Schrittweise Migration von der Legacy zur neuen Anwendung mit Hilfe von Anti-Corruption Layer

Für unsere Kunden haben wir schon viel erreicht

Die RISC Software GmbH besitzt langjährige Erfahrung in der Entwicklung nativer Software-Systeme in technischen Anwendungen. Diese sind häufig sehr komplex und haben hohe Anforderungen hinsichtlich Robustheit, Zuverlässigkeit und Determinismus der Ergebnisse bzw. Zeit. Zu ihren Kunden zählen Airbus, WFL, DS-Automotion und zahlreiche andere, mit denen oftmals langjährige Entwicklungspartnerschaften bestehen. Das Dienstleistungsspektrum erstreckt sich über die Entwicklung neuer Systeme, das Re-Engineering bestehender Systeme bis hin zum Consulting und zu Schulungen.

Entwicklung der Software Bibliothek VML (Virtual Modeling Library)

Die VML (https://virtual-modeling.at) ist eine Software Bibliothek der RISC Software GmbH, die neue Algorithmen zur exakten geometrischen Modellierung von Festkörpern implementiert. Sie unterstützt an Constructive Solid Geometry (CSG) angelehnte Operationen und die Hüllvolumenberechnung. Die VML bietet eine gute Skalierbarkeit bezüglich der Anzahl der bei der Modellierung durchgeführten Operationen. Die Bibliothek ist bestens für industrielle Anwendungen geeignet, welche kombinierte Anforderungen hinsichtlich geometrischer Genauigkeit, Geschwindigkeit und Skalierbarkeit aufweisen. Um die hohe Effizienz sicherzustellen, ist die VML in C++ implementiert und verwendet parallele Algorithmen, welche das Potential moderner Hardware-Architekturen wie Multi-Core Central Processing Units (CPUs) und Graphic Processing Units (GPUs) ausnutzen. Die VML kommt z. B. bei dem Produkt CrashGuard Studio der Firma WLF Millturn Technologies (https://www.wfl.at) zum Einsatz. CrashGuard Studio ist eine 3D-Simulationssoftware für multifunktionale CNC-Dreh-Bohr-Fräszentren, die es ermöglicht, Maschinen mit ihrer komplexen Kinematik und den umfangreichen Bearbeitungs- und Erweiterungsmöglichkeiten sehr realitätsgetreu nachzubilden.

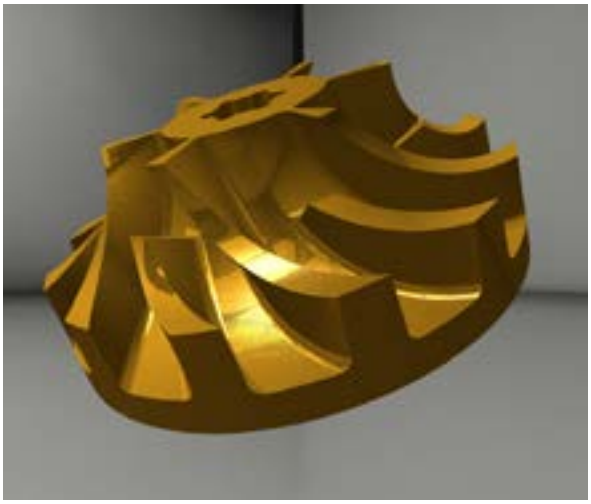


Abbildung 7: Software Bibliothek VML (Virtual Modeling Library)

Re-Engineering des Strukturoptimierungssystems Lagrange

Airbus Defence and Space setzt das multidisziplinäre Strukturoptimierungssystem Lagrange im Bereich des Engineering von Flugzeugstrukturen ein und startet dessen Entwicklung Anfang der 80er Jahre in der Programmiersprache Fortran77. Die RISC Software GmbH startete das Re-Engineering dieses Software-Systems im Jahr 2005. Im ersten Schritt ging es um das Ersetzen einer kritischen Berechnungskomponente, um deren Limitierung hinsichtlich der maximalen Problemgrößen aufzuheben. Dabei erfolgte die Neuentwicklung in C++ und wurde in das Alt-System integriert. In einem nächsten Schritt wurde das Gesamtsystem vom alten Fortran 77 Sprachstandard auf den neueren Fortran 2003/2008 portiert. Da das Gesamtsystem an vielen Stellen Limitierungen hinsichtlich der maximalen Problemgrößen aufwies und die Erweiterbarkeit aufgrund des Designs nur eingeschränkt gegeben war, wurde ein inkrementelles Re-Engineering des Gesamtsystems in Fortran 2003/2008 durchgeführt. Dabei sollte das neue System auch jederzeit im Produktivbetrieb verwendet werden können und die gleichen Ergebnisse wie das Alt-System liefern.

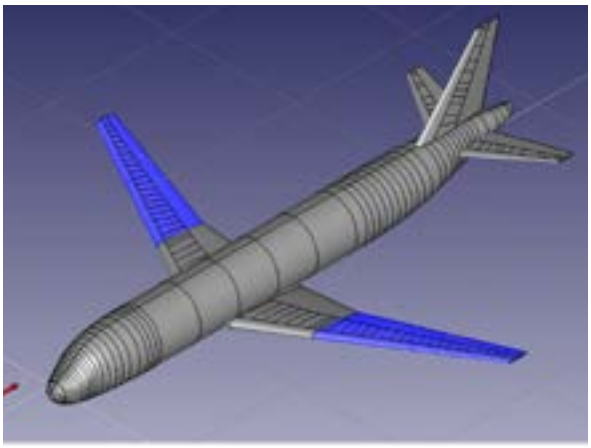


Abbildung 8: Strukturoptimierungssystem Lagrange im Bereich Airbus Defence and Space

Consulting für Optimierungsmaßnahmen einer Schmierfilmberechnungsroutine

Die FH Wels entwickelte im Rahmen eines Simulationssystems eine Routine für die Berechnung des elastohydrodynamischen Drucks, welcher im Ölfilm zwischen Kolben und Zylinder im einem Verbrennungsmotor auftritt. Die RISC Software GmbH führte ein Consulting für Optimierungsmaßnahmen dieser in C++ implementierten Berechnungsroutine durch. Dabei sollte das Potential moderner Hardware-Architekturen bestmöglich ausgenutzt werden. Durch eine Analyse des Systems konnten die kritischen Punkte identifiziert und geeignete Optimierungsmaßnahmen erarbeitet werden. Abschließend wurde im Rahmen eines Workshops die Vorgehensweise und die Ergebnisse vermittelt. ◆

Abbildung 9: Verbrennungsmotorsimulation für die Berechnung des elastohydrodynamischen Drucks

Welcome Change

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

Der neue Scrum Guide

Welcome Change – ein essentieller Grundsatz in der agilen Entwicklung, der im Scrum-Framework durch die drei Säulen Transparenz, Inspektion und Adaption in seiner Wichtigkeit unterstützt wird. Und wie auch die Veränderung während der Produktentwicklung uns täglich begegnet, genauso ist es auch notwendig, Veränderung in der Entwicklung der agilen Praktiken und Frameworks zuzulassen und zu leben. Am 18. November 2020 war es soweit: der neue Scrum Guide wurde veröffentlicht. Finden Sie hier die auf den ersten Blick grundlegenden Änderungen.
Vorweg, was hat sich nicht geändert? Scrum ist immer noch Scrum – ein leichtgewichtiges Framework, welches Teams bei der Lösung komplexer Probleme hilft und auf die Lieferung von Wert für Kund*innen ausgelegt ist. Kundenzentriert und dennoch einen konzentrierten Blick auf das Team und dessen Mitglieder.

Mehr Freiheiten führen zu mehr Individualität

Das Scrum Framework schrieb nie sonderlich viele Praktiken vor, um ein Team optimal einsetzen und wachsen zu lassen. Der neue Scrum Guide geht hier noch einen Schritt weiter und wird in vielen Teilen weniger „vorschreibend" und reduziert sich auf das Wesentliche. Zum Beispiel wurden beim Daily Scrum die bisher definierten Fragen gestrichen.

Ein stärkeres Team

Das Scrum Team ist künftig nur noch ein Team. Das mag auf den ersten Blick eigenartig klingen, dennoch bestand mit der bisher im Scrum Guide geführten Bezeichnung des „Development Teams" die Gefahr, dass sich die Entwickler*innen als „Subteam" formierten und ggf. die positiven Synergien im gesamten Scrum Team ausblieben. Im neuen Scrum Guide wird der Begriff „Development Team" ersetzt durch „Developer". Es gibt jetzt nur noch ein Team – das Scrum Team!

Keine Hüte mehr – nur noch Verantwortlichkeiten

Bisher wurden Scrum Master, Product Owner und Development Team anhand von Rollen (roles) beschrieben. Der neue Scrum Guide verzichtet auf den Begriff der Rollen komplett und führt nun einheitlich die Kommunikation von Verantwortlichkeiten (accountabilities) ein. Auch dies mag dazu führen, dass Scrum Teams künftig näher zusammenwachsen und sich besser gemeinsam bewegen können.

Der Scrum Master wird zum „echten" Leader

Beim Scrum Master wurde eine auf den ersten Blick kleine Änderung durchgeführt: vom „Servant Leader" wurde dieser zum „True Leader", welcher aber das Team und die Organisation weiter unterstützt. Diese Änderung muss uns erst überzeugen, da das Konzept des „Servant Leader" bisher ein klares Bild und Abgrenzung zum klassischen Management darstellte, welches dadurch etwas geschwächt werden könnte.

Selbstorganisation vs. Selbstmanagement

Das „self-organizing" Scrum Team wird zum „self-managing" Scrum Team. Hierdurch wird der hohe Stellenwert der Autonomie des gesamten Teams hervorgehoben. Während im Scrum Guide von 2017 noch das Development Team selbstorganisiert war, so ist es nun das Scrum Team, welches gemeinsam entscheidet.

Der Weg ist das Ziel

Als Basis für diese gemeinsamen Entscheidungen soll nun auch neu das Produktziel dienen. Das Produktziel soll ein gemeinsames Bild vom Produkt erzeugen und die möglichen Wege klarer darstellen.

Drei Verpflichtungen

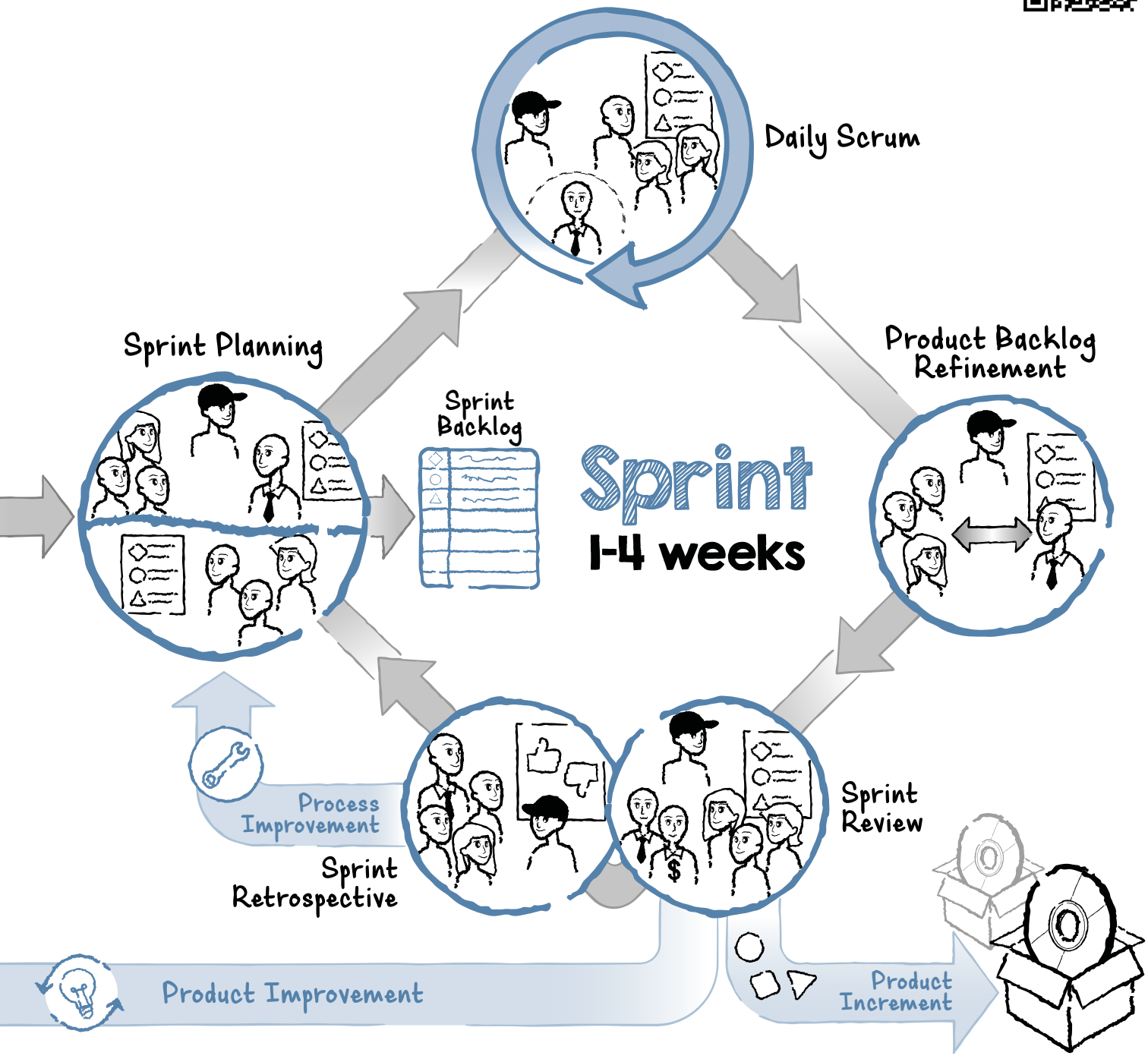
Im Scrum Guide von 2017 wurden bereits das Sprintziel und die Definition of Done erwähnt, dennoch waren diese nicht sonderlich stark verankert. Gleichzeitig mit der Einführung der Produktzieles wurden diese drei nun als Verpflichtungen (commitments) den Artefakten zugewiesen:

- Product Backlog erhält das Produktziel
- Sprint Backlog erhält das Sprintziel
- Inkrement erhält die Definition of Done

Das Sprintziel wurde bisher im Rahmen des Sprint Plannings kommuniziert. Neu ist hier nun die gemeinsame Festlegung des Sprintzieles durch das Scrum Team. Durch die Institutionalisierung einer vorangestellten, zusätzlichen Frage im Sprint Planning ergeben sich dort jetzt folgende Themen:
- Warum ist dieser Sprint wertvoll?
- Was kann im Sprint umgesetzt werden?
- Wie wird die gewählte Arbeit umgesetzt?

Scrum

Mit folgenden QR Code kann das Poster runtergeladen werden:



| Fazit

Die Änderungen im Scrum Guide 2020 können wir einfach mit einem Wort beschreiben: spannend. Wir sind neugierig die neuen Themen in unseren Projektalltag aufzunehmen und mit ihnen zu experimentieren. Wir sind zuversichtlich, dass viele der hier genannten Änderungen die Entwicklung unserer Arbeitsweise positiv beeinflussen. Gerne laden wir unsere Kund*innen ein, mit uns gemeinsam diesen Schritt der Weiterentwicklung zu gehen.

| Über Scrum

Scrum ist eine agile Arbeitsweise im Projektmanagement, die von von Ken Schwaber und Jeff Sutherland entwickelt wurde und häufig in der Softwareentwicklung zum Einsatz kommt. Die Vorgehensweise ist im sog. Scrum-Guide festgelegt und wird firmen- und herstellerunabhängig entwickelt. Am 18. November 2020 wurde eine aktualisierte Version des Scrum-Guides veröffentlicht. ◆

„OK Google: Was ist Natural Language Processing?"

– Sandra Wartner, MSc
Data Scientist in der Abteilung Logistics Informatics

Wie Maschinen die menschliche Sprache lesen, entschlüsseln und verstehen

Sprache ist nicht gleich Sprache – während Menschen über Tausende von Jahren eigene Kommunikationswege geschaffen haben, dienen Millionen an Nullen und Einsen als Maschinencode bzw. Maschinensprache dazu, dass Computer Befehle verstehen und ausführen können. Die Verarbeitung von natürlicher Sprache durch Maschinen (Natural Language Processing, kurz NLP) ermöglicht es, die menschliche Sprache maschinell zu lesen, zu entschlüsseln und zu verstehen. Sprachassistenten, Rechtschreibkorrekturen, E-Mail-Spamfilter – NLP als Technologie ist omnipräsent und verbirgt sich bereits hinter vielen, tief in unserem Alltag verankerten Abläufen und Softwareapplikationen. Das oftmals verborgene Potenzial in vielen Datenbergen ist dabei noch lange nicht erschöpft.

Die von uns Menschen generierte Datenflut wächst von Tag zu Tag. Allein für das Jahr 2020 zeigten Wachstumsstatistiken, dass jede Sekunde pro Person 1,7MB an Daten generiert werden. Wir verschicken Fotos, legen Dokumente in der Cloud ab, streamen Musik oder Videos, kommunizieren über Videokonferenztools und nutzen noch viele weitere Annehmlichkeiten, die uns das Internet bietet. Allein in den letzten beiden Jahren wurden ca. 90 % der weltweiten Datenmenge generiert – und die Zahlen steigen weiter an. Auch die COVID-Pandemie trägt u.a. durch den erhöhten Bedarf an Online-Kommunikation und Home-Office zu einer stark ansteigenden Wachstumsrate bei.

Ein beträchtlich großer Teil der existierenden Datenberge besteht aus Textdaten. Diese generieren wir vor allem selbst, indem wir z. B. E-Mails, Produktrezensionen, Tweets oder Textnachrichten verfassen. Gleichzeitig können wir das Potenzial der kontinuierlich wachsenden Datenberge nutzen, um die uns im Alltag immer häufiger unterstützenden Anwendungen überhaupt erst zu schaffen. Wir verwenden Übersetzungsfunktionen von einer Sprache in eine andere (z. B. DeepL), beim Verfassen von Texten und Nachrichten machen uns Programme auf Tippfehler aufmerksam, digitale Sprachassistenten wie Alexa, Cortana, Siri und co. unterstützen uns bei einer Vielzahl an Tätigkeiten und Suchmaschinen bieten Suchvervollständigung an – all diese Dienste und Funktionen bauen auf einer wesentlichen Technologie auf: Natural Language Processing (NLP).

Künstliche Intelligenz als Schnittstelle zwischen Mensch und Maschine

Die maschinelle Verarbeitung natürlicher Sprache stellt kein neues Forschungsfeld dar, allerdings haben die letzten Jahre aufgrund der Verfügbarkeit von höherer Rechenleistung, enormen Datenmengen (Big Data) sowie modernen Algorithmen eine Vielzahl an revolutionären Errungenschaften im NLP-Umfeld mit sich gebracht: Computer sind in der Lage zu lesen, zu verstehen und zu sprechen. Als interdisziplinäres Feld der Linguistik, Computerwissenschaft und Künstlichen Intelligenz (KI) ermöglicht NLP die Kommunikation zwischen Mensch und Maschine in unterschiedlichen Formen (geschrieben und gesprochen) und in einer Vielzahl an Sprachen.

Wollen wir den Google-Assistenten auf unserem Smartphone befragen, damit uns eine synthetisierte Stimme NLP erklärt, reicht ein einfaches „OK Google" und die nachgestellte Frage. Im Optimalfall erhalten wir eine Antwort, die uns zufriedenstellt und genau jene Information liefert, nach der wir gesucht haben. Diese Aufgabe klingt für die Ausführung durch einen Menschen zwar relativ einfach, für eine Maschine bedeutet dies jedoch, Sprache in ihre elementaren Bestandteile aufzulösen, die Frage und den Kontext zu verstehen und sequenziell unterschiedliche Problemstellungen lösen zu müssen.

Natural Language Understanding (NLU) konzentriert sich auf die Extraktion von Informationen aus Text und damit auf das Erwerben von Textverständnis hinsichtlich eines bestimmten Teilaspekts. Dabei spielen v.a. Syntax (grammatikalische Struktur) und Semantik (Bedeutung von Wörtern) eine wesentliche Rolle. Beispiele hierfür sind:

- grammatikalische Analysen (z. B. Part-of-Speech (POS) Tagging),
- Erkennen von Personen, Orten oder anderen Schlüsselwörtern in Texten (z. B. Named Entity Recognition (NER)),
- Stimmungs- und Meinungsanalyse (Sentimentanalyse) und
- Klassifizierung von Text in vordefinierte Kategorien.

Natural Language Generation (NLG) fokussiert sich auf die Erzeugung von natürlicher Sprache und kommt u.a. für die automatisierte Erstellung, Zusammenfassung oder Übersetzung von Texten zum Einsatz.

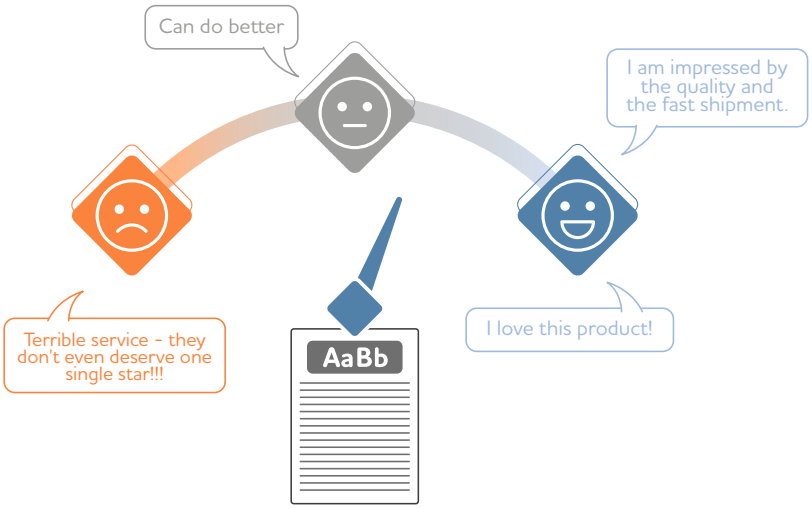


Abbildung 10: Sentiment-Analyse

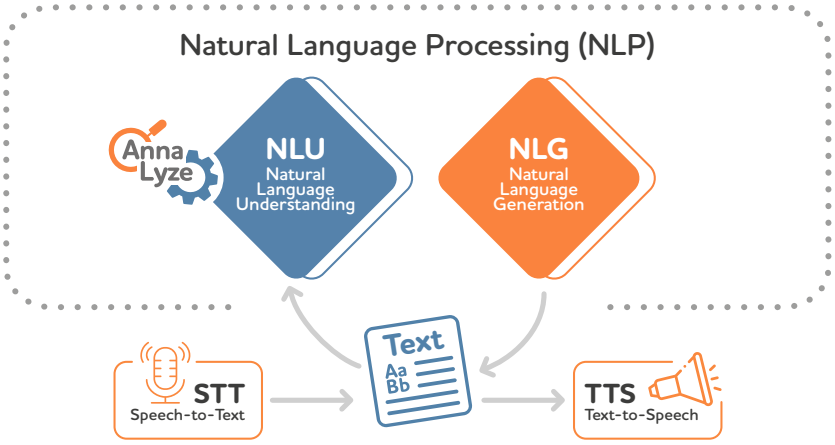


Abbildung 11: NLP-relevante Komponenten

Da NLU und NLG ausschließlich mit geschriebener Sprache arbeiten, wird häufig eine Komponente für Spracherkennung (Speech-to-Text, kurz STT) und Sprachsynthese (Text-to-Speech, kurz TTS) notwendig, die dann als Schnittstelle zwischen dem NLP-System und der realen Welt fungieren.
Für das „OK Google"-Beispiel bedeutet dies, dass die Anfrage mittels STT von der gesprochenen Sprache in die geschriebene Sprache konvertiert wird. Auf die Anfrage, welche durch NLU erkannt wurde, kann reagiert werden, indem beispielsweise relevante Suchergebnisse gesammelt und ausgewertet werden. Das dabei generierte Wissen kann zumeist (abhängig von der Art des Resultats) mit Hilfe von NLG und TTS akustisch wiedergegeben oder die besten Treffer am Endgerät angezeigt werden.

NLP wird in der Informatik als eine der kompliziertesten Problemstellungen betrachtet. Natürliche Sprache an sich verfügt über keine identifizierbare Struktur (häufig auch als unstrukturierte Daten bezeichnet) und ist ein komplexes System aus aneinandergereihten, teilweise voneinander abhängigen Zeichen und daher von Grund auf nicht einfach zu verstehen. Deutsch, Englisch, Russisch, Japanisch, Arabisch – jede Sprache hat ihre eigene komplexe Syntax und Eigenheiten. Hinzu kommen weitere Erschwernisse, da Sprache oft nicht linear funktioniert, sondern sich unterschiedlicher Stilmittel, Redewendungen und Informationen zwischen den Zeilen bedient. Das Erkennen von Sarkasmus ist selbst für einen Menschen nicht immer möglich. Mehrdeutigkeiten einzelner Wörter müssen über eine Kontextanalyse aufgelöst werden, um bspw. das Wort „Bank" mit einer Sitzgelegenheit oder einem Geldinstitut eindeutig assoziieren zu können. Nuscheln, Stottern, das Sprechen im Dialekt und Hintergrundgeräusche erschweren dem Sprachassistenten die Auswertung und können im Weiteren zu einer fehlerhaften Antwort führen. Algorithmen müssen sich diesen und noch einigen weiteren Herausforderungen stellen, um ihren Anforderungen gerecht zu werden.

Ältere Systeme griffen auf regel- bzw. rein statistisch-basierte Ansätze zurück, wohingegen der Durchbruch erst mit Machine Learning (insbesondere Deep Learning) und der Verfügbarkeit großer Datenmengen errungen werden konnte. Machine Learning-Modelle versuchen, aus einer Menge an Beispielen allgemeine Muster abzuleiten (Wie verwenden Menschen Sprache? Welche Grammatikregeln kommen zur Anwendung?) und diese für die Entscheidung eines Individualfalles anzuwenden – ähnlich einem Kind, das die menschliche Sprache erlernt. Je mehr Beispiele dem System zur Verfügung gestellt werden und je besser diese die Realität bzw. das zukünftige Anwendungsszenario widerspiegeln, desto höher liegt die Trefferquote bei neuen, unbekannten Aufgaben, die das System lösen soll. Die aktuell vielversprechendsten Modelle bzw. State-of-the-Art Ergebnisse für Aufgaben aus dem NLP-Bereich werden mit Deep Learning Algorithmen erzielt, die eine komplexere Modellierung erlauben als herkömmliche Machine Learning Modelle. Deep Learning wurde von der Funktionsweise des menschlichen Gehirns inspiriert und setzt vielschichtige Neuronale Netze ein. Durch die hochgradig verknüpften Strukturen wird "tiefgehendes Lernen" ermöglicht, welches gerade für das komplexe Konstrukt der Sprache essentiell ist.

Text Analytics und Use-Cases im Unternehmen

Um das oftmals ungenutzte Potenzial in den Unternehmensdaten auszuschöpfen und Businessprobleme lösen zu können, müssen bestehende (Roh-)Daten untersucht und Wissen aus diesen abgeleitet sowie quantifiziert und visualisiert werden. Mit Text Analytics kann dieser Prozess abgebildet werden, um große Mengen an unstrukturierten Textdaten zu verarbeiten und Einblicke zu gewinnen. Nur wenn für die Ergebnisse ein einheitliches Verständnis aller Stakeholder geschaffen und der Schritt der nahtlosen Integration von Lösungen in bestehende Workflows und Systeme bewältigt werden kann, können daraus weitere Handlungsentscheidungen abgeleitet und damit der Erfolgsfaktor für das Unternehmen langfristig gesteigert werden.

Immer mehr Unternehmen aus unterschiedlichen Branchen setzen auf NLP-Lösungen, um die angehäuften, unterschiedlichen Textformen in einer Vielzahl an Bereichen besser managen und nutzen zu können. Besonders wenn wiederkehrende Aufgaben zu erledigen sind, kann eine Automatisierung dieser Tasks sinnvoll sein.
Im Folgenden sind beispielhafte Use-Cases aufgelistet, um die breite Anwendbarkeit von NLP-Lösungen darzustellen.

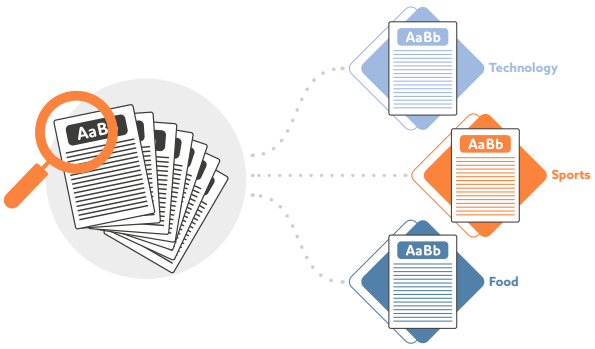


Abbildung 12: Dokumentenklassifikation

Use Case: Automatisierte Dokumentklassifikation

Sie arbeiten im Controlling und möchten zukünftig nur noch all jene Dokumente (bzw. Dokumenttypen) bekommen, für die Sie zuständig sind.

Use Case: Automatisierte Extraktion von Informationen aus Dokumenten wie bspw. Rechnungen oder Lieferscheinen

Sie sind Leiter*in der Warenannahme und wollen zukünftig die Details der Lieferungen nur noch kontrollieren anstatt sie händisch zu erfassen.

Use Case: Customer Support

Sie sind Onlineversandhändler*in und möchten die Reaktionszeiten des Kundenservices durch die automatisierte Verarbeitung und Beantwortung von Kundenanfragen verkürzen.

Use Case: Automatisierte Bewertung von Kundenfeedback

Sie sind Marketingbeauftragte*r und möchten auf Social Media einen Überblick über die Stimmung und die Reaktionen in Bezug auf Ihre neue Werbekampagne erhalten.

Use Case: Social-Media-Analyse

Sie sind Angestellte*r im Verfassungsschutz und wollen extremistische, radikale und gewaltverherrlichende Social Media-Profile und Posts entdecken und überwachen.

Use Case: Unterstützung in der klinischen Dokumentation und Organisation

Sie arbeiten als Facharzt und möchten essentielle Informationen aus mehreren, umfangreichen Anamnesen einzelner Patient*innen zusammenfassen, um einen ganzheitlichen Blick über die Krankheitsgeschichte zu erhalten.

Fazit

Der Fortschritt im NLP-Bereich ist nicht aufzuhalten und stellt kontinuierlich neue und bessere Lösungen für ein breites Spektrum an Problemen bereit. Die Präzision der entwickelten Modelle sowie deren Verfügbarkeit für die breite Masse steigen weiterhin und immer mehr bahnbrechende Entwicklungen schaffen den Sprung aus dem Forschungsbereich in die Produktion. Spannend bleibt auf alle Fälle, welche weiteren Durchbrüche die kommenden Jahre mit sich bringen werden – klar ist, sie werden kommen. Die RISC Software GmbH unterstützt Sie gerne bei der Einreichung und Durchführung von (Forschungs-) Projekten im Bereich Natural Language Processing. ◆

C++20 Concepts

– Michael Hava MSc
Senior Software Architect in der Abteilung Industrial Software Applications
und Mitglied des C++ Standard Committee

Ein kurzer geschichtlicher Überblick über die Entwicklung der Programmiersprache und die Umsetzung von „Templates" und „Concepts".

Der berühmte Bjarne Stroustrup, der Erfinder der auch heute noch weit verbreitenden Programmiersprache C++, begann 1979 in den Bell Labs ursprünglich mit der Entwicklung von C with Classes. Sein Ziel war eine Programmiersprache, welche die Geschwindigkeit von C mit den für große Projekte notwendigen Organisationsfähigkeiten (=Objektorientierung) von Simula verbinden soll.

Aufgrund des großen Erfolgs beginnt Stroustrup kaum 10 Jahre später (1987) mit der Arbeit an einer Erweiterung der inzwischen in C++ umbenannten Sprache: Templates – dieses generische Konzept soll die Implementierung von generischen Klassen und Funktionen ermöglichen, und so nebenbei die aus der C übernommenen Präprozessor-Makros weitgehend ersetzen. Die Entwicklung dieser Templates beruhte im Wesentlichen auf drei Designzielen:

- full generality: Templates sollen nicht auf einzelne Anwendungsfälle ausgerichtet werden, sondern für die breite Verwendung ausgelegt sein.
- zero overhead: Mittels Templates erzeugter Code soll nach dem Kompilieren von handgeschriebenem Code (hinsichtlich Effizienz, Performance, etc.) nicht unterscheidbar sein.
- good interfaces („constraints"): Ähnlich wie C++ die Typsicherheit im Vergleich zu C verbessert hatte, sollen auch generische Schnittstellen eine Verbesserung darstellen.

Ergebnis der Arbeit: Es scheint zum gegenwärtigen Forschungsstand unmöglich, alle drei Ziele gleichzeitig zu erreichen. Da die ersten beiden Ziele essenziell für die Akzeptanz von Templates sind, während Constraints „nur" die Usability verbessern, werden Templates ohne ihnen zu C++ hinzugefügt. Constraints sollen so bald als möglich nachgeliefert werden.

Algorithmen werden auf algebraischen Strukturen definiert

1994 wird C++ (inzwischen auf dem Weg ein ISO-Standard zu werden) um die Standard Template Library (STL), einer Bibliothek für generische Algorithmen und Behälterklassen, erweitert. Sie ist das Ergebnis von Alexander Stepanovs fast 20-jähriger Forschung zum Thema generische Programmierung und unterscheidet sich fundamental von zeitgenössischen objektorientierten Ansätzen.

Am Anfang dieser Forschungstätigkeit stand 1976 seine Erkenntnis: algorithms are defined on algebraic structures. Basierend auf dem mathematischen Modell der algebraischen Strukturen entwickelte er Concepts [1], den zentralen Grundpfeiler von generischer Programmierung. Analog zu algebraischen Strukturen beschreiben Concepts die benötigten Operationen und unterliegenden mathematischen Axiome, welche ein Algorithmus an die zu verarbeitenden Daten stellt. Ziel der generischen Programmierung ist es, Algorithmen nur basierend auf den minimal notwendigen Concepts zu definieren und sie so für eine Vielzahl an konkreten Typen verwendbar zu machen.

Nach Experimenten in mehreren Programmiersprachen (Scheme, Ada, ...) ist C++ die erste, welche in seinen Augen ausdrucksstark genug für generische Programmierung ist – auch wenn die fehlende Unterstützung für Concepts aufwändige Emulationen notwendig macht. Mit der Integration der STL in die Standardbibliothek wird auch die Idee von Concepts in C++ übernommen. Aus programmiersprachlicher Sicht stellen Concepts eine formale Repräsentierung von Constraints dar. Der Fokus für die Fertigstellung von Templates verschiebt sich in der Folge zu Concepts. Nichtsdestotrotz erscheint der erste C++ ISO-Standard 1998 (C++98) ohne Erweiterungen des Template-Systems, Concepts müssen weiterhin emuliert werden.

Keine Fertigstellung von Templates in Sicht

In den ersten Jahren nach C++98 fokussierte das Standardkomitee sich auf Fehlerbehebung und Stabilisierung. Daher nahm die Entwicklung von Concepts erst ab 2003 im Rahmen der Arbeiten am nächsten C++ Standard (C++0x) wieder Fahrt auf. Bjarne Stroustrup und Gabriel Dos Reis veröffentlichten eine Reihe von Papers über ein mögliches Concepts Design. Eine Forschungsgruppe der Universität Indiana veröffentlichte 2005 ebenfalls ihre Ergebnisse. Aus den beiden konträren Ansätzen entstand 2006 das gemeinsame Design der C++0x Concepts, welches 2008 in den Arbeitsentwurf des C++ Standards integriert wurde.

In den folgenden Monaten wurden allerdings einige Probleme am Design ersichtlich. Da die Fertigstellung des neuen Standards schon mehrfach verschoben wurde und davon auszugehen ist, dass die Fehlerbehebung zu weiteren Verschiebungen führen wird, entfernt das Standardkomitee Concepts 2009 wieder aus dem Arbeitsentwurf. C++11 erschien somit 13 Jahre nach dem ersten Standard immer noch ohne Fertigstellung von Templates.

Erste Concepts-basierte Bibliothek

Während den Arbeiten an C++14 entwerfen Bjarne Stroustrup, Gabriel Dos Reis und Andrew Sutton basierend auf den Erkenntnissen von C++0x Concepts bereits 2013 einen neuen Entwurf – Concepts Lite. Der zentrale Unterschied zum alten Ansatz: diesmal fokussiert man sich auf ein „minimales Feature" [2], welches später bei Bedarf erweitert werden kann. Das Ergebnis ihrer Arbeit wird 2015 als Technical Specification (TS) [3] veröffentlicht.

Im Sommer 2017, vier Monate nach der Fertigstellung von C++17, werden Concepts basierend auf dem TS in den Arbeitsentwurf für C++20 integriert. Zusammen mit der Spracherweiterung wird eine Bibliothekserweiterung von vorgefertigten grundlegenden Concepts bereitgestellt. 2018 folgt mit Ranges, die erste Concepts-basierte Bibliothek – sie enthält unter anderem die mit Concepts überprüften STL-Algorithmen. C++20 wird im Februar 2020 verabschiedet und im September 2020 einstimmig angenommen. ◆

[1] Das implizite Pendant der generischen Programmierung zu den expliziten Interfaces aus der objektorientierten Programmierung.

[2] So fehlen unter anderem Sprachmittel zu Axiomen und Definitionschecks.

[3] Eigenständiges ISO Dokument das mögliche Erweiterungen für einen Standard enthält. Das Ziel ist Nutzerfeedback zu erhalten und eine (überarbeitete) Version des Inhalts in einen ISO Standard einzubringen.

Die besseren Entscheidungen treffen dank Prescriptive Analytics

– DI Dr. Michael Bögl
Mathematical Optimization Specialist in der Unit Logistics Informatics

Schrittweise von der deskriptiven zu präskriptiven Analytik

Die Digitalisierung und Themen wie Industrie 4.0 und Internet of Things haben dazu geführt, dass viele Unternehmen ihre Daten in großem Umfang strukturiert sammeln. Diese gesammelten Daten werden dann auf unterschiedliche Weise verwertet. Sind in diesen Daten Zusammenhänge vorhanden und lässt sich ein Prognosemodell ableiten, dann lässt sich dieses Modell auch in der Planung und Steuerung berücksichtigen. Dies garantiert einen signifikanten Mehrwert, wie die Verringerung von Kosten und Zeitaufwand, eine effizientere Ressourcennutzung etc. Wie so etwas konkret aussieht, was vorausgesetzt wird und welche Möglichkeiten entstehen, zeigt dieser Beitrag.

Von der Datensammlung zu Prescriptive Analytics

Die Digitalisierung der vergangenen Jahre hat die Basis für Unternehmen geschaffen, um laufend Daten zu ihren Prozessen und Abläufen zu sammeln und strukturiert zu speichern. Um einen Mehrwert zu schaffen, müssen Unternehmen diese Daten bestmöglich nutzen. Dabei werden aus den gesammelten Daten Prognosemodelle erstellt, um damit zukünftige Entwicklungen, Ereignisse oder Zustände abschätzen zu können. Das können bspw. Modelle für Absatzprognosen, Abnutzungen von Werkzeugen in der Produktion, Kundenbedarfe, Lagerstände, verkehrsabhängige Fahrzeiten, etc. sein.

Durch die Kombination der Prognosemodelle mit Optimierungsmodellen (zur Berechnung von optimalen Entscheidungen) können unterschiedliche Szenarien automatisch berechnet und gegenübergestellt werden. Dadurch steht den Verantwortlichen eine solide Grundlage zur Verfügung, um optimale Entscheidungen treffen zu können. Die transparente Entscheidungsgrundlage garantiert, dass die getroffenen Entscheidungen stets nachvollziehbar und argumentierbar sind.

Abhängig davon, wie die Daten genutzt werden, spricht man von unterschiedlichen Anwendungsgebieten, bzw. von Phasen oder Stufen, wobei der Nutzen steigt, je höher die Stufe ist (siehe auch Abbildung 13 und Tabelle 1).

Werden die gesammelten Daten genutzt, um auszuwerten, was passiert ist, so spricht man von deskriptiver Analytik (Descriptive Analytics). Zusammenhänge in den Daten werden dabei noch nicht erkannt. Ein Ergebnis könnte bspw. sein, dass man erkennt, dass die Produktqualität sowohl im Tagesverlauf als auch saisonal schwankt.

1. In der diagnostischen Analytik (Diagnostic Analytics) werden die Ursachen für Zusammenhänge ermittelt. Die Voraussetzung, um gezielt Verbesserungsmaßnahmen zu setzen ist, die Ursachen zu kennen. Für das Beispiel aus Punkt 1 bedeutet das: Die Produktqualität schwankt saisonal und im Tagesverlauf, weil die Qualität ab einer Temperatur von 30 °C sinkt; oder nach den ersten Aufträgen nach einem Werkzeugwechsel sind die Maschinen neu zu kalibrieren (verlängert gegebenenfalls die nicht-produktive Zeit, führt allerdings zu höherer Qualität).
2. Sind alle relevanten Zusammenhänge und Einflussgrößen ermittelt, dann können Modelle erstellt werden, die diese Zusammenhänge abbilden. Mit diesen Modellen können Aussagen über das zukünftige Verhalten getroffen werden. Man spricht dabei von prädiktiver Analytik (Predictive Analytics). Prognostiziert werden z. B. zukünftige Absätze, der Ausschuss in der Produktion oder die Qualität des Produktes.
3. In der präskriptiven Analytik (Prescriptive Analytics) werden zunächst zukünftige Ereignisse prognostiziert, gegebenenfalls verschiedene Entscheidungsmöglichkeiten simuliert und bewertet und dann die beste Handlungsalternative berechnet. Für diese Berechnung können Unternehmen auf bestehendes Know-how und vorhandene Planungsalgorithmen zurückgreifen. Gegebenenfalls können die Planungsalgorithmen auch erweitert oder zusätzliche Methoden integriert werden.

Die beschriebenen Anwendungsgebiete sind in Abbildung 13 entsprechend ihres Fokus und ihres Nutzen dargestellt. Während die deskriptive und diagnostische Analytik in Richtung Vergangenheit gerichtet sind und ein tieferes Verständnis für die Daten und Zusammenhängen in diesen schaffen können, sind prädiktive und präskriptive Analytik in Richtung Zukunft orientiert. Dort werden die erkannten Zusammenhänge in den Daten genutzt, um daraus Nutzen für die Zukunft zu schaffen.

Präskriptive Analytik in der Entscheidungsunterstützung

Abbildung 14 zeigt die wesentlichen Komponenten eines präskriptiven Systems. Die Basis wird durch die gesammelten Daten aus unterschiedlichen Datenquellen gebildet. Aus den Daten sowie dem vorhandenen Expert*innenwissen werden die notwendigen Modelle (Prognosemodell und Entscheidungsmodell) erstellt. Anhand dieser Modelle können dann unterschiedliche Szenarien bewertet werden und die Entscheidungsträger die beste Entscheidung treffen.

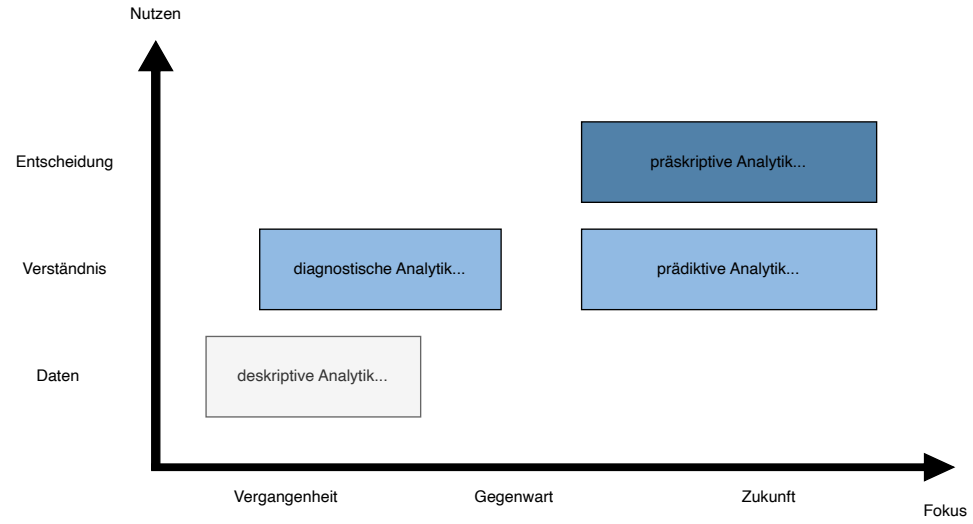


Abbildung 13: Anwendungsgebiete

Stufe	Bezeichnung	Anwendungsebene	Beispiele
1	Descriptive Analytics	Auswerten der gesammelten Daten und Reporting	• Die Produktqualität schwankt sowohl im Tagesverlauf als auch saisonal • Die Fahrzeit von LKW schwankt im Tages- und Wochenverlauf • Montag und Dienstag sind mehr Kundenbestellungen abzuarbeiten als an den anderen Wochentagen
2	Diagnostic Analytics	Ursachen für Zusammenhänge	• Die Produktqualität schwankt saisonal und im Tagesverlauf, weil die Qualität ab einer Temperatur von 30 °C sinkt • Die Fahrzeit der LKW schwankt aufgrund des höheren Verkehrsaufkommens morgens und abends und ggf. Montag und Freitag • Die Bestellungen sind höher, da die Kunden vermehrt am Wochenende bestellen
3	Predictive Analytics	Modelle für zukünftiges Verhalten	• Der erwartete Ausschuss in der Produktion • Die voraussichtliche Fahrzeit auf einer bestimmten Strecke an einem bestimmten Tag zu einer bestimmten Zeit • Die erwarteten Bestellmengen für die nächsten Wochen pro Wochentag
4	Prescriptive Analytics	Zukünftige Entscheidungsmöglichkeiten werden simuliert und die beste Handlungsalternative gewählt	• Wann sollte eine Wartung der Produktionsmaschinen durchgeführt werden, damit die Produktqualität die gegebenen Anforderungen erfüllt • Wann muss der LKW losfahren, um alle Ziele pünktlich zu erreichen • Wie viele Produkte müssen vorgehalten werden um out-of-stock Situationen zu vermeiden und wie viele Ressourcen werden benötigt um die Bestellungen möglichst schnell abarbeiten zu können

Tabelle 1: Anwendungsebenen

Hier ist ein System zur Entscheidungsunterstützung dargestellt, d.h. das System bewertet unterschiedliche Szenarien, letztlich wird die Entscheidung allerdings vom Verantwortlichen getroffen. Wird die Entscheidung vom System getroffen, dann spricht man von einer Entscheidungsautomatisierung.

Welche Methoden stehen zur Verfügung

Für die Umsetzung von präskriptiven Modellen stehen Methoden aus unterschiedlichen Disziplinen zur Verfügung. Das Konzept der präskriptiven Analytik schreibt dabei keine bestimmten Methoden vor. Überwiegend werden Methoden aus der Künstlichen Intelligenz eingesetzt. Dabei liegt der Schwerpunkt auf Machine Learning und Data Mining, statistischen Analysemethoden, mathematischer Programmierung und Simulation. Die Auswahl geeigneter Methoden hängt sehr stark von den verfügbaren Daten, den Rahmenbedingungen und vor allem auch von den Zielen ab (white-box vs. black-box Modelle, Integration von Expertenwissen, usw). Das verfügbare Methodenportfolio ist dabei so umfangreich, dass für viele Use-Cases geeignete Methoden eingesetzt werden können.

Fragestellungen und Anwendungsfälle

Neben den oben angeführten Beispielen gibt es zahlreiche unterschiedliche Anwendungsgebiete für präskriptive Analytik, wie beispielsweise die folgenden:

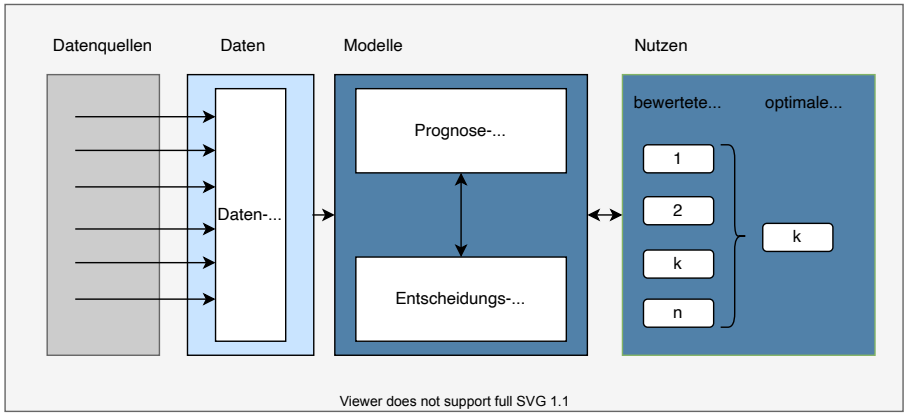


Abbildung 14: Komponenten eines präskriptiven Systems

Ermittlung der Restkapazität

Wie viele Aufträge können noch produziert werden und wann werden diese Aufträge fertiggestellt?

Integrierte Produktions- und Tourenplanung

Wann sollen welche Aufträge produziert werden, damit die Kund*innen transportoptimal beliefert werden können?

Digitaler Zwilling zur Entwicklung von Produktionsmaschinen

Wie soll die Produktionsmaschine genau ausgestaltet sein, damit der Betrieb möglichst effizient ist?

Preisfindung

Wie hoch soll der Preis für ein Produkt sein, damit eine breite Käuferschicht angesprochen wird und dabei der Deckungsbeitrag hoch bleibt?

Zusammensetzung der Rezeptur

Wie soll die Rezeptur (Mengenverhältnisse, Umgebungsbedingungen) aussehen, damit das Produkt bestimmte Kriterien erfüllt?

Gesundheitsmanagement

Wie sind Platzierung, Durchmesser, Dauer und Intensität des Strahls bei der Strahlentherapie (optimal) zu wählen, damit die Schäden am umliegenden Gewerbe so gering wie möglich sind?

Der Weg zur präskriptiven Analytik

Für Unternehmen, die bereits eine große Datenbasis haben, bietet präskriptive Analytik die Chance, aus den bereits verfügbaren Daten zusätzlichen Wert zu generieren, indem die aus den Daten gewonnenen Erkenntnisse in die Planung einfließen. Unternehmen, die bis jetzt noch keine durchgängige Datenerfassung haben oder ihre Daten noch nicht für solche Aufgaben nutzen, haben die Möglichkeit, den Wert, den sie aus ihren Daten ziehen, schrittweise zu steigern. Die Stufen von der deskriptiven bis zur präskriptiven Analytik bauen aufeinander auf und bereits in der diagnostischen Analytik können (gemeinsam mit den Unternehmensexperten) wertvolle Erkenntnisse gewonnen werden. Dann können die nächsten Schritte geplant werden, um den Weg hin zur präskriptiven Analytik erfolgreich gestalten zu können. ◆

Datenversteher: Durch intelligente Graphdatenbanken Unternehmensdaten nutzen

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

Mit moderner Datenbanktechnologie die eigenen Daten intuitiv erfassen und verstehen

Die große Stärke von Graphdatenbanken – das sind Datenbanken, die anhand von Graphen vernetzte Informationen in Form von Knoten und Kanten in Verbindung bringen und speichern – ist die umfangreiche Abbildung von Beziehungen zwischen Datenpunkten. Dies ermöglicht eine intuitive Abbildung vieler Real-World-Szenarien, die gerade während der letzten Jahre stark an Bedeutung gewonnen haben. Beispiele dafür sind die Modellierung von Beziehungen zwischen Personen in sozialen Netzwerken, die Erstellung von Kaufempfehlungen im E-Commerce, oder die Erkennung von betrügerischen Vorgängen im Finanzbereich. Neben diesen Anwendungsbereichen sind Graphdatenbanken auch in den Bereichen der industriellen Fertigung, der Verkehrsdatenanalyse oder der IT-Infrastrukturüberwachung für die Ermittlung kausaler Zusammenhänge hilfreich.

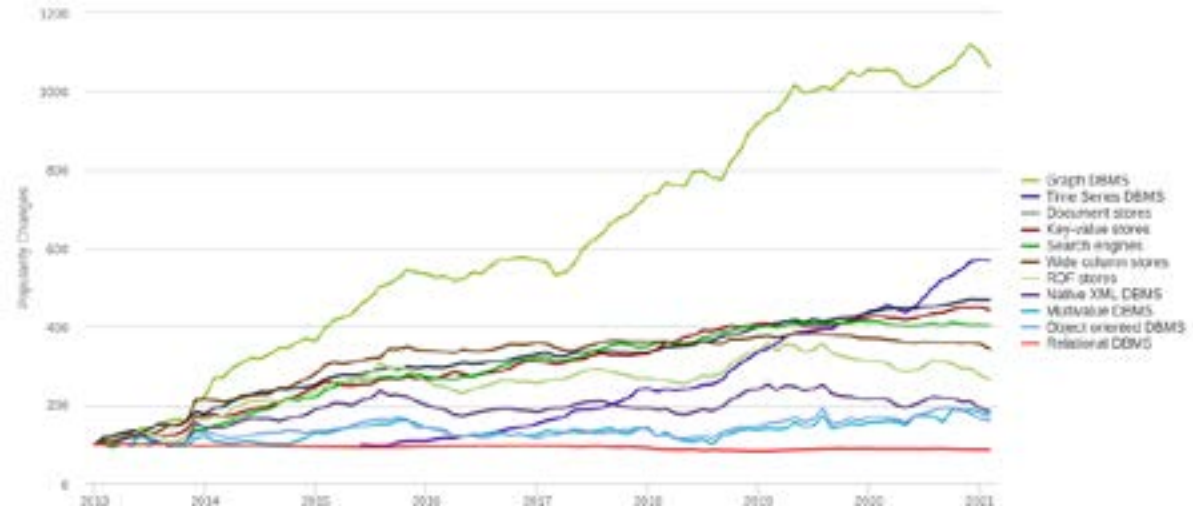


Abbildung 15: Laut einer Untersuchung über die Beliebtheit von Datenbankkategorien (https://db-engines.com/en/ranking_categories) stellen Graphdatenbanken die über die letzten Jahre die am schnellsten wachsende Kategorie von Datenbanktechnologien dar. Graphdatenbanken repräsentieren ihre Daten als Mengen von Knoten und Kanten, wobei Knoten mit Attributen versehene Datenobjekte darstellen, während die Kanten die Verknüpfungen zwischen den Objekten repräsentieren.

Graphdatenbanken im Vergleich zu relationalen Datenbanken

Relationale Datenbanken sind ausgezeichnet dafür geeignet, tabellarische Strukturen abzubilden, wie sie beispielsweise im kaufmännischen Bereich gängig sind. Durch den Einsatz der dritten Normalform werden die Daten nach den Objekten die sie beschreiben klar getrennt in Tabellen abgelegt, wobei andere Objekte, mit denen sie in Beziehung stehen, über Fremdschlüssel referenziert werden. Das Ziel hierbei ist unter anderem eine Datenduplikation zu vermeiden und eine Verknüpfung durch flexible Abfragen in der Structured Query Language (SQL) zu ermöglichen. Diese Anwendungen zeichnen sich weiters dadurch aus, dass ein einmal entworfenes Datenmodell üblicherweise über einen längeren Zeitraum konstant bleibt. In Problemdomänen, in denen man primär an den Verknüpfungen zwischen den Daten interessiert ist, weist das relationale Datenmodell aber Schwachpunkte auf. Da Verknüpfungen über Fremdschlüsselbeziehungen abgebildet werden, müssen Abfragen oft als mehrstufige Joins umgesetzt werden, welche gerade bei großen Tabellen sehr laufzeitintensiv werden können. Zusätzlich treten in zahlreichen Anwendungsdomänen semistrukturierte Daten auf, deren Struktur sich über die Zeit ändert. Solche Daten lassen sich schwer in

dem rigiden Datenmodell einer relationalen Datenbank abbilden. Zusätzlich ermöglicht das flexiblere Datenmodell einer Graphdatenbank oft, die zu modellierende Realität direkter abzubilden und somit sowohl den Entwurf des Datenmodells als auch die darauf angewandten Abfragen intuitiver zu gestalten. Darüber hinaus ist es bei Änderungen in der Anwendungsdomäne leichter anpassbar, da der Graph ohne einschneidende Änderungen im Datenmodell erweitert werden kann.

Der Weg zum besseren Datenverständnis

Überall dort, wo Zusammenhänge zwischen Datenpunkten im Zentrum des Interesses stehen, bieten Graphdatenbanken eine solide Basis für die weitere Analyse. Sie ermöglichen eine direktere und flexiblere Abbildung der Problemdomäne als relationale Datenbanken.

Darüber hinaus zeigen sie den Weg von den gesammelten Daten zur Beantwortung der Fragekategorie nach der Ursache von potenziellen oder aktuellen Problemen: Warum ist dieses Teil ausgefallen?

- Warum gibt es Engpässe und Preissteigerungen bei der Beschaffung von Teilen für die Produktion?
- Warum schlägt eine Therapie bei einem Patienten besser an?
- Warum ist diese Serverapplikation überlastet? ◆

Use Case: Lieferketten-Modellierung

Komplexe Lieferketten können abgebildet und so potenzielle Engpässe oder Abhängigkeiten von einzelnen Lieferanten erkannt werden.

Use Case: Pharmazeutische Industrie

Die Abbildung der Zusammenhänge zwischen biologischen und chemischen Daten kann die Entwicklung neuer Pharamzwutika bschleunigen

Use Case: Medizin

Graphdatenbanken können Zusammenhänge zwischen den Krankheitsgeschichten von Patienten und der Wirksamkeiten von Therapien aufzeigen. Ebenso können Wechselwirkungen von Medikamenten ermittelt werden.

Use Case: Ursachenermittlung für Maschinenprobleme

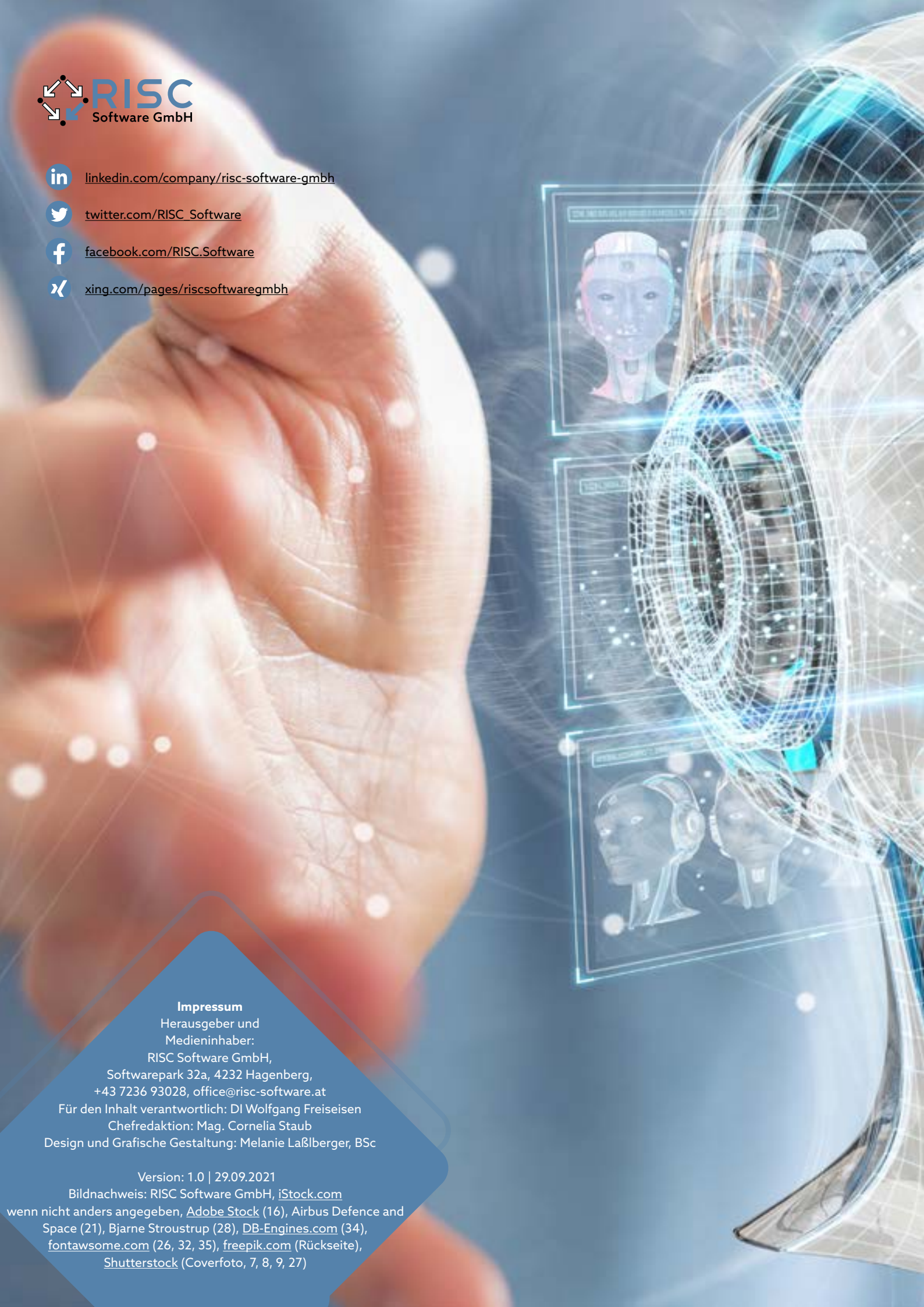
Zusammenhänge zwischen Sensorwerten und Maschinenzuständen werden oft nur vermutet oder sind unbekannt. Eine Sammlung und zeitbasierte Verknüpfung von Daten in einer Graphdatenbank kann versteckte Korrelationen aufzeigen.

Use Case: Nachvollziehbarkeit der Schaltungen in komplexen Anlagen

Von Verkehrbeeinflussungsanlagen bis zu Raffinerie- und Fabrikanlagen: Steuerungsalgorithmen nehmen automatisiert Schaltungen vor. Graphdatenbanken können helfen, diese für den Menschen nachvollziehbar zu machen, Fehler zu entdecken und die Effizienz solcher Anlagen zu steigern.

Use Case: Überwachung und Optimierung von IT-Anlagen

Parameter von Servern und Applikation können automatisiert gesammelt werden. Basierend darauf können Graphdatenbanken eingesetzt werden, um transitive Abhängigkeiten zwischen Diensten sowie Überlastungen oder kritische Elemente in IT-Infrastrukturen zu erkennen.



RISC Software GmbH

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, BSc

Version: 1.0 | 29.09.2021
Bildnachweis: RISC Software GmbH, iStock.com
wenn nicht anders angegeben, Adobe Stock (16), Airbus Defence and Space (21), Bjarne Stroustrup (28), DB-Engines.com (34), fontawsome.com (26, 32, 35), freepik.com (Rückseite), Shutterstock (Coverfoto, 7, 8, 9, 27)