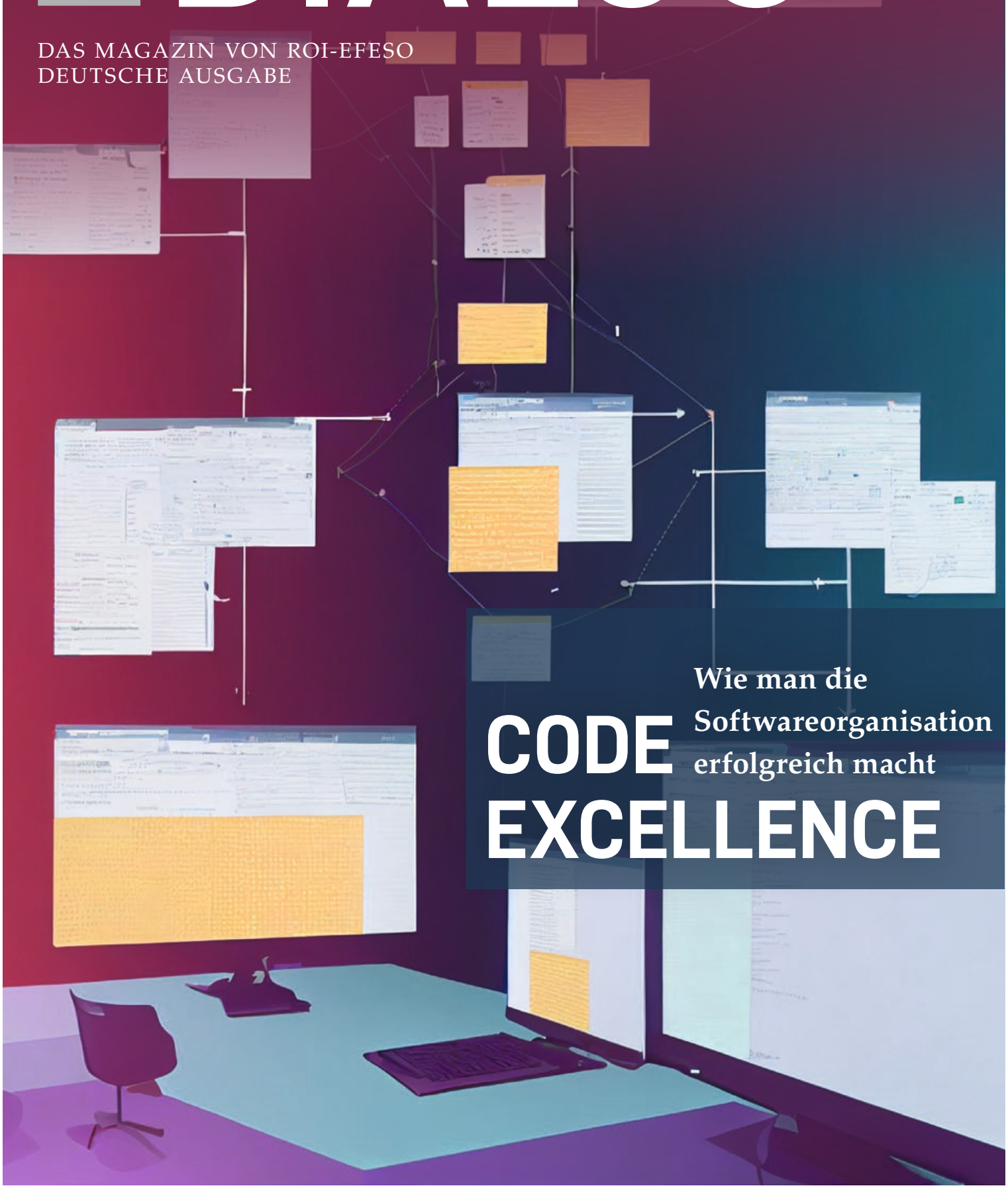
#67

ROI EFESO
MANAGEMENT CONSULTANTS

DIALOG

DAS MAGAZIN VON ROI-EFESO
DEUTSCHE AUSGABE



CODE EXCELLENCE

Wie man die Softwareorganisation erfolgreich macht

INHALT

AUSGABE #67

Wo liegen die Grenzen der Automatisierung und Effizienz? Das Titelbild dieses Magazins wurde von einem KI-Programm des unabhängigen Forschungslabor Midjourney erzeugt. Solche Lösungen stehen am Kreuzungspunkt gleich mehrerer großer Debatten – über Relevanz und Substituierbarkeit der menschlichen Kreativität, über das Verständnis von Innovation und Kunst, oder über Fragen des intellektuellen Kapitals. AI Generated Art entsteht über sogenannte Prompts – Textzeilen, die von der KI immer wieder neu in Bilder übersetzt werden – hier das Zusammenspiel der Beitragstitel dieser Ausgabe. https://www.midjourney.com/

IMPRESSUM
V. i. S. d. P.: Hans-Georg Scheibe | ROI-EFESO Management Consulting AG
Infanteriestraße 11 | D-80797 München | Tel. +49 (0)89 121590-0 | E-Mail: dialog@roi.de | Vorstand: Michael Jung, Hans-Georg Scheibe, Sebastian Diers
Grafik-/Bildrechte: Soweit nicht anders vermerkt, liegen die Bildrechte bei der ROI-EFESO Management Consulting AG
und den einzelnen Autoren, Adobe, Shutterstock sowie iStock. Redaktion: WYZE Projects GmbH | Gestaltung: miramarketing GmbH | © ROI-EFESO Management Consulting AG

NACHHALTIGKEIT
ROI-EFESO kompensiert seine Emissionen nach dem „Clean Development Mechanism" mit Zertifikation von „FOKUS ZUKUNFT".

Das Magazin DIALOG wurde auf FSC® zertifiziertem Papier produziert

AUF DEM WEG ZUR SOFTWARE-EXZELLENZ

Hans-Georg Scheibe
Management Board

Stellen Sie sich vor, Sie sind Werksleiterin einer Fabrik. Die Menschen in Ihrem Team sind absolute Profis – schwer zu finden, schwer zu halten, schwer auf Wettbewerbsniveau zu entlohnen. Die in Ihrer Fabrik hergestellten Produkte sind kritisch für die Zukunfts- und Wettbewerbsfähigkeit des gesamten Unternehmens. Sie wissen es. Ihre Mitarbeiter wissen es.

Und auch Ihr CEO. Ihre Fabrik sollte also ein Musterbeispiel an Fokussierung, Effizienz und Qualität sein. Aber sie ist es nicht.

Denn 30 Prozent ihrer verfügbaren Arbeitskraft verbrauchen Ihre Mitarbeiter damit, Fehler auszumerzen. Weitere 40 Prozent gehen drauf, um sich mit technischen Schulden, die in Jahren, wenn nicht Jahrzehnten angehäuft wurden, herumzuschlagen und sie, bildlich gesprochen, zu begleichen. Weitere 10 Prozent verdampfen durch Verteilung unnützer und fehlerhafter Informationen. Und lediglich 20 Prozent ihrer Zeit können Ihre talentierten, hochbezahlten und kreativen Mitarbeiter auf ihre Kernaufgabe, das Produkt, das ihr Unternehmen so dringend braucht, verwenden.

Dieses Szenario mag in einer klassischen Fabrik undenkbar erscheinen – in Software Factories ist es tägliche Realität. So immens wichtig Software Engineering heute ist, so weit ist es vom Reifegrad einer klassischen Ingenieurswissenschaft entfernt. Dafür gibt es eine ganze Reihe von Gründen. Dazu zählt einerseits die besondere Beschaffenheit von Software, die sich fundamental von Hardware unterscheidet. Andererseits die Ungreifbarkeit und Abstraktheit global verteilter, dynamischer und hochvernetzter Softwarearchitekturen und ihre gigantischen Dimensionen. Diese Wesensunterschiede zwischen Hardware und Software machen es so komplex, die beiden Sphären zu integrieren – was in einer Welt, die immer stärker software-defined ist, aber unabdingbar ist.

Vor diesem Hintergrund gewinnt die Aufgabe, Softwareentwicklung zu einem transparenten, effizienten und auf kontinuierliche Verbesserung ausgerichteten Prozess zu machen, strategische Relevanz und hohe Dringlichkeit für jedes produzierende Unternehmen.

Diese Ausgabe unseres Magazins ist deshalb der Frage gewidmet, was Software-Exzellenz bedeutet, wie die Effizienz in der Softwareentwicklung verbessert werden kann und welche Ansätze erfolgversprechend sind, um eine schlagkräftige Software-Organisation aufzubauen.

WAS SOFTWAREENTWICKLUNG HERAUSFORDERND MACHT

Ein Artikel von Josef Wohlrab, ROI-EFESO

Die Bedeutung von Software für die zukünftige Wettbewerbsfähigkeit zeigt vor allem der disruptive Wandel der Automobilindustrie. Vor wenigen Jahren noch belächelt, hat sich Tesla zum Treiber der Branche entwickelt. Das Unternehmen macht sich nicht allzu viele Gedanken über Spaltmaße – und folgt stattdessen der Idee der „fahrenden Software", was etablierte Automobilmarken radikal herausfordert. Von der Börse beflügelt, hat Tesla die Traditionskonzerne damit in den Wettbewerb auf einem ungewohnten Terrain gezwungen. Und der Auftritt der Großen der Branche lässt bislang zu wünschen übrig. Was aber macht die Softwareentwicklung so herausfordernd und wo liegen die Besonderheiten und Unterschiede zu traditionellen industriellen F&E-Projekten?

Software lebt

Ein wesentlicher Unterschied zur Hardware ist, dass diese zum SOP (Start of Production) komplett fertig sein muss. Software hingegen „lebt" – sie wird auch nach der Auslieferung an den Kunden kontinuierlich weiterentwickelt, kann angepasst und erweitert werden – und muss das auch. Zum einen als Teil eines sich ständig ändernden Eco-Systems, zum anderen, um dem Nutzer verbesserte, erweiterte oder neue Features anzubieten und durch den Nachverkauf weiteres Geschäft zu generieren. Der limitierende Faktor hierfür ist die Kapazität der zugrunde liegenden Hardware, d.h. Speicher, Prozessor und Peripherie. Entgegen der verbreiteten Annahme, dass Software schnelllebig sei, lebt erfolgreiche Software im Kern von evolutionärer Weiterentwicklung – und kann dabei älter werden als manch eine Hardware, auf der sie läuft.

Trotz dieser Langlebigkeit ist die Entwicklung von Software sehr viel dynamischer und kurzzyklischer als die von Hardware. Dafür gibt es verschiedene Ursachen. So existieren häufig bereits Teile des gewünschten Codes und stehen sogar als Open Source kostenlos zur Verfügung. Zeitaufwendiges Prototyping ist nicht notwendig, neue Versionen können in kurzen Zyklen geliefert werden. Software lässt sich im Wesentlichen in System- und Applikationssoftware unterscheiden. Die Systemsoftware, z.B. Betriebssysteme, stellt hierbei die Grundfunktionalität für die Applikationssoftware des Nutzers sicher. Die Annahme scheint naheliegend, dass die im Hintergrund laufende Systemsoftware, sofern sie einmal gut läuft, analog zu guter Hardware keinen wesentlichen Überarbeitungsbedarf aufweist. Doch beide Softwaretypen müssen gleichermaßen gepflegt und angepasst werden, da sich die Peripherie (Hardware) und vor allem die Schnittstellen des Ökosystems weiterentwickeln. Die kontinuierliche Gestaltung, Pflege und Wartung der Softwarearchitektur ist daher eine wesentliche Kernkompetenz in der Softwareentwicklung. Die immer größer werdende Anzahl der Hardware-Varianten, die unterstützt werden müssen, und die Notwendigkeit langfristiger Softwarepflege fordern eine hohe Kompetenz im Versions- und Konfigurationsmanagement. Dafür braucht es eine saubere Dokumentation und eine stringente Nachvollziehbarkeit.

MVP is key

Hardware-Prototypen sind teuer und treiben mit langen Lead Times maßgeblich die Entwicklungsdauer von der Idee bis zum SOP. Änderungen in späten Entwicklungsphasen sind kostenintensiv und Fehler im Entwicklungszeitraum führen zu exponentiellen Folgekosten. Daher setzt man bei der Hardware nach dem Prinzip des Systems Engineering auf ein möglichst gutes Frontloading in der Erhebung und Beschreibung der Anforderungen und konzentriert sich darauf, fehlerfreie Produkte an Kunden auszuliefern. Anforderungen an die Software werden hingegen im Laufe der Entwicklung kontinuierlich verfeinert oder sie entstehen überhaupt erst während der Entwicklung.

Hardware wird mit komplexen und teuren Werkzeugen produziert, wobei Entwicklungsfehler nachträglich korrigiert werden können. Software hingegen ist nach der Entwicklung fertig. Sie wird nicht produziert, sondern kopiert. Daher ist es umso wichtiger, dass das Original bereits eine hohe Qualität und Reife aufweist, was jedoch selbst bei einfacher Software alles andere als selbstverständlich ist.

Um Fehler in der komplexen Softwareentwicklung zu reduzieren und die Qualität so früh wie möglich sicherzustellen, existieren unterschiedliche Methoden, wie z.B. automatisierte Tests, statische Code-Analysen, Test-driven Development oder Pair Programming. Der entscheidende Hebel liegt jedoch in der Möglichkeit, sehr früh Nutzerfeedback einzuholen, um eine Entwicklung am Nutzer vorbei zu verhindern. Durch die schnelle Erzeugung eines MVP (Minimal Viable Product), d.h. eines Produkts mit minimalem für den Kunden sinnvollem Funktionsumfang, kann kurzfristig das Feedback (und ggf. neue Anforderungen) abgeholt und damit die weitere Entwicklung gesteuert werden. Die Entwicklung von Software erfolgt deshalb deutlich kurzzyklischer als die von Hardware: mit einem lebenden Anforderungs-Backlog statt eines festgeschriebenen Lasten- und Pflichtenhefts.

Etwas schwieriger wird dies bei technisch komplexen Systemen, etwa bei komplexer Software, die für Hardware-basierte Produkte, wie das Automobil, produziert wird. Hier ist es notwendig, beide Projektphilosophien – den agilen Softwareansatz und den für die traditionelle Produktion typischen Wasserfall-Ansatz – im Rahmen eines hybriden Entwicklungsmodells gut zu verzahnen.

Architektur dominiert den Lebenszyklus

Besonders hoch ist die Relevanz sauberer Architekturarbeit bei technisch komplexen Systemen, die sicherheitsrelevante Funktionalitäten realisieren und daher höchsten Anforderungen an Zuverlässigkeit, Qualität und Sicherheit unterliegen.

Vor allem bei Greenfield-Ansätzen in der Softwareentwicklung liegt der Schlüssel für eine hochwertige Software in einer guten Architektur, die stimmig zur zugehörigen elektrisch-elektronischen (E/E-) und Hardware-Architektur ist und von einer logischen und funktionalen Architektur abgeleitet wurde. Gute Architekturarbeit setzt eine hohe Kompetenz und Disziplin voraus. Dabei basiert sie vor allem auf zwei Prinzipien: der Hierarchisierung und der Modularisierung. Zum einen können nur dadurch Änderungen, Implikationen und Abhängigkeiten systematisch bewertet werden. Zum anderen lassen sich Module – standardisierte Blöcke mit definierten Schnittstellen – in der Software deutlich besser skalieren als in der Hardware, da sie nicht durch physikalische Eigenschaften in ihren Einsatzmöglichkeiten limitiert sind.

Bei diesen Überlegungen gilt es, einen wichtigen Aspekt zu beachten. Software wird selten auf einem weißen Blatt entwickelt. Deshalb spielt die kontinuierliche Verbesserung der Software („Refactoring“) eine wesentliche Rolle in der Softwareentwicklung.

Dadurch wird laufend die Komplexität reduziert, die Lesbarkeit erhöht und damit die Wartbarkeit und Erweiterbarkeit sichergestellt. Erfolgt dies nicht kontinuierlich in ausreichendem Umfang, läuft man Gefahr, durch die Anhäufung technischer Schulden eine nicht mehr beherrschbare Komplexität zu erzeugen. Das würde das Ende der Software bedeuten. Für die stetige Weiterentwicklung ist deshalb ein sauberes Versions- und Konfigurationsmanagement notwendig.

Auch Hardware bzw. deren Struktur lässt sich optimieren. Doch ist dies deutlich schwieriger und aufgrund der benötigten neuen Produktionswerkzeuge mit hohen Kosten verbunden. Außerdem entsteht deutlich seltener ein umfassender Optimierungsbedarf durch neue Umgebungsbedingungen oder Funktionserweiterungen.

Same same but different

Die Entwicklung von Software folgt in mehrfacher Hinsicht anderen Logiken, Dynamiken und technologischen Prozessen als die von Hardware. Gerade für Unternehmen, die stark auf klassische Produktentwicklung ausgerichtet sind, bedeuten diese Unterschiede eine Herausforderung. Sie müssen andere Organisationsformen, Metriken und Effizienzsteigerungsinstrumente etablieren, die eine erfolgreiche Steuerung der Softwareentwicklung erlauben. Gleichzeitig gilt es, die beiden Welten zu integrieren und eine erfolgreiche Zusammenarbeit zu organisieren, um eine nachhaltig erfolgreiche Transformation ins Zeitalter der smarten, digitalen Industrie sicherzustellen.

„DIE CODE-STADT IST IMMER IM UMBAU."

Gespräch mit Dr. Johannes Bohnet, CEO des Software-Process-Mining-Pioniers Seerene.

DIALOG: Herr Dr. Bohnet, Sie sprechen davon, dass Softwareorganisationen zu Softwarefabriken werden müssen. Was steckt hinter dieser Analogie und wo sind ihre Grenzen?

JB: Softwaresysteme sind heute das Fundament von zentralen Geschäfts- und Produktionsprozessen in jeder Industrie. Wenn man eine städtebauliche Analogie nutzt, dann sind Softwaresysteme große gewachsene Städte, die den Ablauf des Business ermöglichen. Diese Software-Städte aufzubauen, sie immer wieder um zusätzliche Funktionalitäten zu erweitern und die Code-Gebäude vor dem Verfall zu schützen, ist allerdings keine triviale Aufgabe. Das liegt zum einen an der Größenordnung, zum anderen an der Komplexität und Abstraktheit. Wir können Softwaresysteme nicht anschauen, wir können sie mit unseren Sinnen nicht erfassen. Aber würden wir den Source Code ausdrucken, würde sich das Papier so hoch stapeln, wie alle Wolkenkratzer Manhattans. Diese enormen Dimensionen sind selbst für Menschen, die unmittelbar mit Software zu tun haben, kaum greifbar. Solche Softwarestrukturen – große Kernbankensysteme oder eine Plattform wie Netflix – lassen sich nur erschaffen, wenn man Softwareentwicklung als Ingenieurswissenschaft betreibt. Die Softwareentwicklung hat allerdings noch nicht den Reifegrad einer klassischen Ingenieurswissenschaft, denn die gesamte Disziplin ist noch ziemlich jung.

Viel wichtiger ist aber die dynamische Natur der Software, die nie fertig ist – genauso wie eine Stadt. Die Code-Stadt ist immer im Umbau. Die Methodologien und Ansätze verändern sich, die Systematiken entwickeln sich weiter, es wird permanent gearbeitet. Was dabei bislang immer fehlt, ist die Transparenz.

Man kann sich Software-Engineering ein Stückweit wie den Turmbau zu Babel vorstellen. So treffen etwa Management-Logiken, deren Sprache Deadlines, Geschwindigkeit und Budgets sind, auf eine technologische Perspektive, die sich an Frameworks oder Modulen orientiert. Softwareentwicklung ist eine Herausforderung geradezu biblischen Ausmaßes und das ist aus meiner Sicht ein wesentlicher Grund, warum wir als Software-Ingenieurswissenschaft noch nicht da sind, wo wir sein könnten. Hinzu kommt, dass der Turm auch noch unsichtbar ist – und die Baustelle nicht darauf angelegt ist, jemals fertigzuwerden.

DIALOG: Wie kann man sich eine Fabrik vorstellen, deren Produkt nicht fertig werden kann, und was bedeutet das für die Steuerung und Effizienz von Softwarefabriken?

JB: Beim Software-Engineering gibt es eigentlich kein Greenfield. Das Produkt, an dem gearbeitet wird, verbleibt immer in der Fabrik. Es wird kontinuierlich weiterentwickelt, angepasst, erweitert. Das ist ein signifikanter Unterschied zur Hardwareproduktion – und ein Treiber für Komplexität und Intransparenz. Es gibt keine Dokumentation über diesen evolutionären Prozess, außer das Produkt selbst. Der Code ist die Dokumentation. Genau daraus erwächst auch die häufige Notwendigkeit für sehr aufwendiges Reverse Engineering, wobei Softwareentwickler die in Software eingebaute Geschäftslogik über das Lesen von Code nachzuvollziehen versuchen.

DIALOG: Lässt sich überhaupt analytisch über einen Prozess sprechen, der eine solche Komplexität und Abstraktheit aufweist?

JB: Global verteilte, dynamisch wachsende Software – das ist ja nicht nur die Software selbst. Es sind Tausende, vielleicht

Zehntausende von Menschen, die kontinuierlich daran arbeiten. Ich glaube, man hat nur eine Chance, das mit dem menschlichen Verstand zu erfassen, wenn man Metaphern verwendet. Deswegen verwenden wir die Metapher der Stadt, um greifbar zu machen, was in Jahren oder auch Jahrzehnten an Source Code entstanden ist, mit Code Units als Gebäuden. Wir zeigen eine hierarchische Modulstruktur, die sich durch Stadtbezirke, Häusergruppen und einzelne Gebäude darstellen lässt. Diese Symbolik erlaubt es, eine gemeinsame Sprache zu etablieren, in der Management und Technik sich austauschen können. Wieso ist mein Gebäude so hoch und rot? Warum brauche ich mehr Zeit oder Budget, um mein Gebäude zu modernisieren? Dadurch lassen sich abstrakte Themen wieder kommunizieren, objektivieren und in notwendige Entscheidungsprozesse einbringen.

DIALOG: Wie kann man vor diesem Hintergrund die Entwicklung hin zu einer echten Ingenieurswissenschaft unterstützen?

JB: Die Softwarefabriken sind heute sehr gut ausgerüstet. Die Entwickler haben Compiler, es gibt Ticketmanagementsysteme, Continuous Integration und vieles mehr. Es fehlt nicht an exzellenten Werkzeugen für die Produktion und da ist nicht mehr viel Raum für Optimierung. Was allerdings fehlt, ist die Anwendung von Methodiken, die in der Hardwareproduktion eingesetzt werden. Dort ist es Standard, dass der Durchfluss gemessen wird, dass jeder einzelne Teilproduktionsschritt erfasst wird, dass sich Aussagen über Qualität und Effizienz treffen lassen und vieles mehr.

Einen solchen Leitstand oder auch einen „Digital Boardroom“, wie wir es nennen, brauchen wir auch für die Softwareproduktion. Heute ist es für das Management kaum möglich, einen wirklichen Einblick in die Softwarefabrik zu gewinnen – ganz unabhängig davon, welche Methodologien und Programmiersprachen verwendet werden oder ob es um die Produktion eines Kernbankensystems geht, einer digitalen Plattform oder einer Software für smarte Wasserhähne. Wir brauchen die durchgängige Transparenz von der operativen bis hin zur strategischen Ebene, um Effizienzverluste zu erkennen und einen zahlengestützten, kontinuierlichen Verbesserungsprozess initiieren zu können. Dieses Vorgehen, bei dem ein Digital Boardroom aufgebaut wird und faktenbasierte Steuerungsprozesse initiiert werden, lässt sich auch als „Software Process Mining“ bezeichnen.

DIALOG: Worin unterscheidet sich Software Process Mining vom herkömmlichen Process Mining?

JB: Process Mining adressiert Prozesse, die per se schon da sind, die greifbar sind, hinter denen konkrete Artefakte oder physische Elemente stehen. Es funktioniert beeindruckend gut auf der Business-Ebene. Aber im Unterbau, auf der Ebene der Softwarefabriken, da greifen klassische Process-Mining-Lösungen nicht. Da ist heute eine Leerstelle, in die das Software Process Mining hineinstößt.

Dabei geht es zunächst darum, den Gegenstandsbereich überhaupt greifbar, darstellbar, kategorisierbar zu machen, Zielkonflikte und Verkantungen zu visualisieren. Die Basis zu schaffen, auf der die Prozesse nachvollzogen werden und ein Optimierungsansatz ablaufen kann. Dafür ist tiefes Domänenwissen erforderlich, das über die herkömmlichen Process-Mining-Systeme nicht zugänglich ist.

DIALOG: Wie ausgeprägt ist das Verständnis für das Thema auf der Business-Seite?

JB: Entscheider verstehen sehr genau, dass Softwarekompetenz existenziell ist. Und sie wissen auch, welche Konsequenzen es hat, wenn ihre Softwarefabriken nicht als Enabler der Transformation fungieren und die Softwareentwicklung nicht mit den Business-Anforderungen mithalten kann. Software wird künftig allgegenwärtig sein, sie durchdringt schon heute einen Großteil der produzierten Hardware und wird selbst in so komplexen und mit enorm hohen Eintrittshürden versehenen Industrien wie Automobilbau zum Unterscheidungsmerkmal. Deshalb wird in den Unternehmen intensiv an agilen Methoden gearbeitet, an der Frage wie man eine Softwareorganisation optimal aufstellt. Dafür wird sehr viel interne Expertise und externe Beratungskompetenz mobilisiert.

Das Problem ist nur, sobald eine Softwareorganisation aufgestellt ist, wird sie typischerweise sich selbst überlassen. Sie wird zur Blackbox. Man kann nicht nachvollziehen, warum Deadlines gerissen werden, Fehler passieren oder an den Bedürfnissen vorbei entwickelt wird. Man hat keinen Einblick, man kann die Prozesse nicht präzise beschreiben und messen. Man weiß nicht, ob man wirklich besser wird. Vielleicht wird man schneller, man hat vielleicht das Gefühl, innovativer zu sein. Aber wird man besser, effizienter? Das ist exakt die Leerstelle, von der ich gesprochen hatte. Und es gibt auf der Business-Seite enormes Interesse, das zu ändern.

DIALOG: Lässt sich denn auf der Grundlage von Software Process Mining Analytics auch die Effizienz der Prozesse in Softwarefabriken berechnen?

JB: Die Messung der Produktivität ist schwierig. Wir können heute kein sinnvolles Verhältnis zwischen Input, der Zeit,

die Entwickler investieren, und der Menge des erzeugten Business Value berechnen. Noch nicht. Das Problem ist, wie man berechnen kann, wie viele Business-Euro durch 100 neue Codezeilen entstanden sind. Wir beschäftigen uns natürlich intensiv mit dem Thema, ebenso das Hasso-Plattner-Institut, mit dem wir eng verbunden sind, aber auch einige andere Organisationen weltweit. Eine gute Nachricht ist aber, dass wir trotzdem Kenngrößen wie die Effizienz berechnen können. Ich kann nachvollziehen, wohin dieses kostbare Gut, die Zeit der Softwareentwickler, hinfließt, und in welche Themen. Ich kann nachvollziehen, wie viel dieser Zeit verloren geht und wie viel für die Schaffung von Business Value, von neuen Funktionalitäten genutzt werden kann.

DIALOG: Welche Erkenntnisse lassen sich dabei typischerweise gewinnen?

JB: Etwa die Erkenntnis, dass rund 80% der Ressourcen in der Softwarefabrik verpuffen. Das bedeutet, dass nur zwei von zehn investierten Euro wirklich Business-Wert generieren. Solche Relationen wären in der Hardwareproduktion unvorstellbar. Es ist extrem viel Sand im Getriebe, es knirscht an allen Ecken und Enden. Jeder, der Softwareentwicklung verantwortet, weiß das, fühlt das, hat aber kaum eine Möglichkeit zu sehen, wo genau in dieser unsichtbaren, dynamischen und verteilten Struktur der Sand ist, was zu tun ist, um diese Situation zu ändern.

Wir greifen da erneut auf die Metapher der Stadt und die entsprechende Visualisierung zurück. Jeder Entwickler kennt Code-Bausteine, die man lieber nicht anfassen will: marode Gebäude, bei denen die Fassade bröckelt, wenn man etwas anbauen will. Das sind Folgen der Altlasten, der „technischen Schuld". Aber man muss natürlich da ran und muss dabei sehr vorsichtig und langsam vorgehen, was sehr viel Zeit erfordert. Solche Strukturen, die natürlich Effizienzkiller sind, können wir zeigen und dafür sorgen, dass darüber disziplinübergreifend gesprochen werden kann.

DIALOG: Allerdings kann man – um in der Metapher zu bleiben – ja nichts daran ändern, dass diese Gebäude nun mal in der Stadt stehen. Sie sind in die Prozesse und Strukturen tief integriert und müssen gepflegt werden – ob effizient oder nicht. Was nutzt mir dann die Transparenz?

JB: Ja, die Gebäude machen schon Sinn. Es ist allerdings so, dass die Code-Stadt ohne Analytics nicht überschaubar ist. Und deshalb herrscht oftmals ein extremer Projektdruck auf die IT von der Business-Seite. Es sollen laufend neue Features und Funktionalitäten erzeugt werden. Um diesen Anforderungen gerecht zu werden, müssen die Entwickler häufig an die alten, maroden Gebäude der Code-Stadt ran. Und es ist essenziell, dass sie dabei die Komplexitäten, die Wechselwirkungen, insbesondere auch die technischen Schulden, die dabei entstehen, der Business-Seite erklären können.

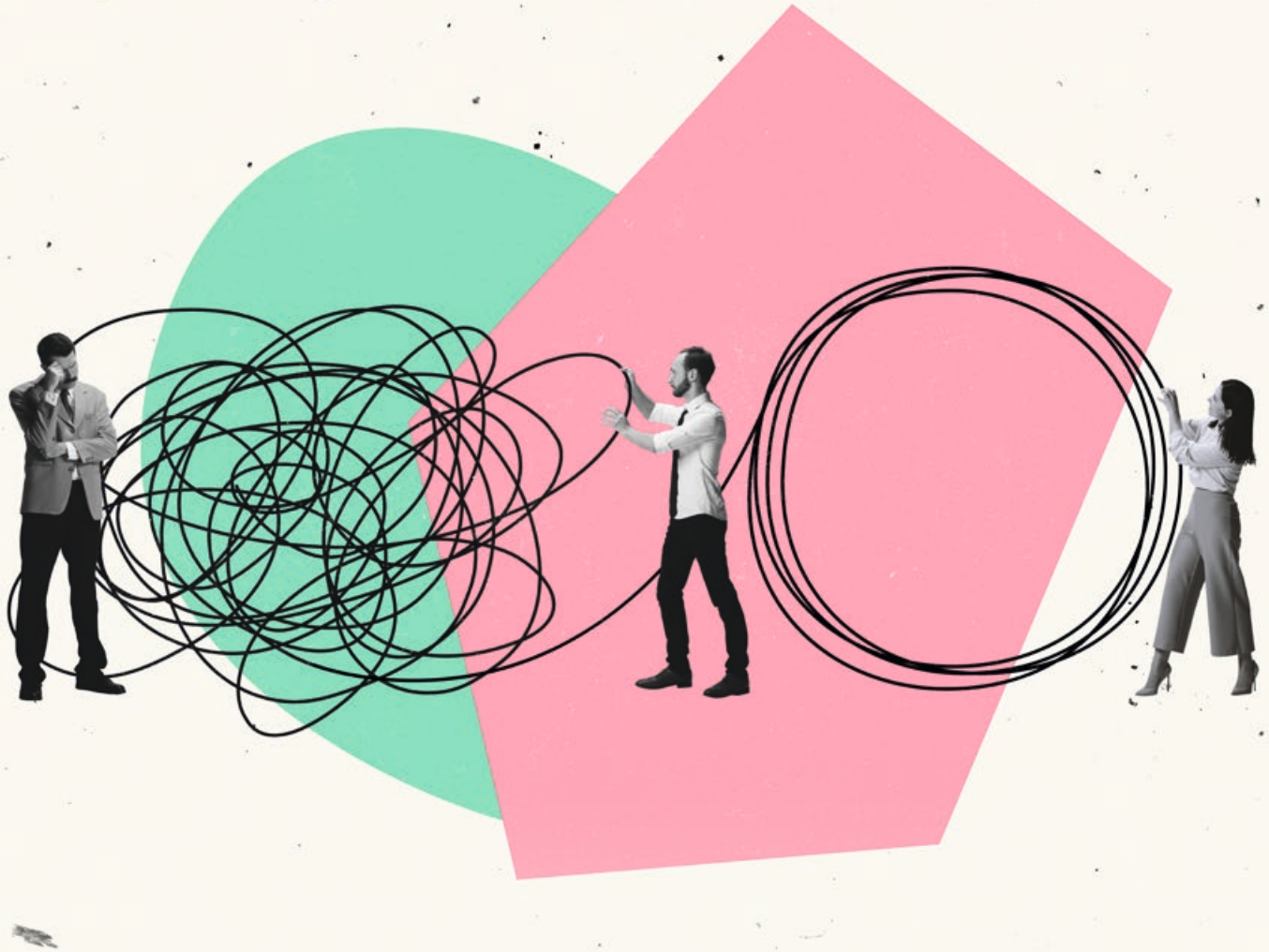
Denn diese Schulden führen einerseits zu teils gravierenden Risiken, die man in das System einbaut, und andererseits zu Zinszahlungen. Diese beiden Aspekte muss man transparent machen. Und das ist etwas, wofür die reine Code-Analyse nicht ausreicht. Denn in einer Fabrik kann man nicht nur die Vorprodukte und Produktionsverfahren betrachten. Es geht auch um die Menschen, die in dieser Fabrik arbeiten und ihre Zeit für bestimmte Themen investieren. Man braucht die Möglichkeit zu sehen, wo die technische Schuld dringend abgebaut werden muss, weil die Zinszahlungen gigantisch hochgeschnellt sind.

Wenn auf der Business-Seite erkannt wird, dass die Entwicklerteams vielleicht 5% ihrer Zeit dafür verwenden, den bestellten Balkon anzubauen, und 95% darauf, die Fassade zu stabilisieren, die auf der anderen Gebäudeseite herunterbröckelt, dann hat man sehr viel erreicht – man hat die sehr spezielle fachliche Perspektive der Entwickler in eine Sprache übersetzt, die an betriebswirtschaftliche Prozesse anschlussfähig ist und vice versa. Und damit lässt sich eine in der Softwareentwicklung absolut kritische Bruchstelle zwischen der Business-Seite und den Technik-Experten überbrücken.

DIALOG: Existiert diese Bruchstelle auch in agilen Setups, wo die Kommunikation enger und kurzzyklischer ist?

JB: Ja. Und ich würde sogar sagen, dass gerade dort der Einsatz von Analytics dringend notwendig ist. Denn in agilen Strukturen, erst recht bei großen Scaled-Agile-Organisationen, lässt sich kaum nachvollziehen, wie viel Entwicklerzeit für welche Themen investiert wird. Agile Teams sind wie kleine Schnellboote. Man kann jedes einzelne Boot betrachten, aber man sieht die Flotte nicht mehr. Es fehlt die übergreifende Perspektive, die sicherstellt, dass nicht nur jedes Boot für sich, sondern die ganze Flotte immer besser wird. Das fehlt in der agilen Welt. Das kann man mit Analytics wieder erreichen.

Beim Einsatz von Analytics sind allerdings zwei Aspekte sehr wichtig. Erstens darf die Einführung der Analytics-Apparatur nicht dazu führen, dass operative Prozesse verzögert werden. Denn die Softwareorganisationen stehen bereits unter hoher Belastung. Die Idee von Software Process Mining darf nicht dadurch diskreditiert werden, dass es diesen Druck erhöht. Zweitens ist eine sehr klare und transparente Kommunikation absolut kritisch. Software Process Mining darf nicht als Überwachung in „Big-Brother-Manier" wahrgenommen werden, sondern als ein Katalysator, um die Sprachverwirrung aufzulösen, als eine Chance, endlich zu zeigen, welche Komplexitäten, Risiken, Aufwände und Zeitverluste bei bestimmten Anforderungen entstehen und welche Faktoren in Entscheidungsprozesse auf der Business-Seite einfließen müssen. Es geht darum, endlich Licht in die Code-Stadt zu bringen.



„ENTSCHEIDEND SIND KUNDENNUTZEN UND DIFFERENZIERUNG.“

Interview mit Bodo Zeug, Executive Vice President und Head of EMEA bei Landis+Gyr sowie Aufsichtsrat bei ROI-EFESO, über Herausforderungen für die Softwareorganisation und ihren Kampf um Effizienz.

DIALOG: Herr Zeug, was bedeutet für Sie Exzellenz in der Softwareentwicklung?

BZ: Das hängt sehr davon ab, von welcher Art von Softwarelösung wir sprechen. Im Grunde geht es im Kern immer darum, wie man mit Softwarelösungen einen Mehrwert für Kunden generieren kann. Landis+Gyr bspw. entwickelt Software, mit der Stadtwerke digitale Energiezähler, sogenannte Smart Meter, betreiben und über die Cloud auswerten. Dabei sind viele Standards und rechtliche Vorgaben zu erfüllen, die gewissermaßen die Rahmenbedingen für die Softwareentwicklung liefern. Die Zähler auszulesen und Daten in die Cloud zu bringen folgt einem relativ strukturierten Vorgehen mit klaren Anforderungen. Hier geht es also um eine effiziente Umsetzung. Daneben verfolgen wir aber auch Ideen, wie sich diese Daten für neuartige Analysen und Use Cases nutzen lassen. Dieser Weg von der ersten Idee bis hin zu einem echten Kunden- und Endkundennutzen ist interativ und daher nicht klar vorgegeben. Hier steht die Effektivität der Softwareentwicklung im Vordergrund und wir arbeiten eng mit unseren Kunden und Partnern zusammen, um knappe Ressourcen möglichst effizient zu nutzen und somit Versorgungssicherheit gewährleisten zu können.

DIALOG: Wenn man nun Industrieunternehmen betrachtet, die gerade erst den Weg der Digitalisierung beschreiten: Wo liegen dort die größten Herausforderungen, bei der effizienten Umsetzung oder einer effektiven Lösungsentwicklung?

BZ: Beide Dimensionen bergen Herausforderungen. Denn zum einen sind die dort eingesetzten Softwarelösungen über

Jahre gewachsen. Hier die Effizienz aufrechtzuerhalten ist eine echte Aufgabe. Viele Services, die man bislang im Haus hatte, werden heute als Cloud-Architektur angeboten. Um diese neuen Möglichkeiten zu nutzen, muss man das Design der Architektur ändern.

Zum anderen gibt es immer mehr neue Use Cases, die Start-ups mit engem Fokus auf ein Thema adressieren. Viele Kunden möchten solche Lösungen in ihre Softwarearchitektur integrieren. Doch es gibt so viele dieser Services, dass man eigentlich ein Start-up im eigenen Unternehmen bräuchte. Auch über Zukäufe ist das nicht immer lösbar. Manche Themen müssen im Haus bleiben, man muss die eigene Technologie und eigene Ideen einbringen. Hier liegen die Herausforderungen in der Effektivität. In vielen dieser Fälle hat man vom Endergebnis noch kein klares Bild. Man begibt sich zusammen mit dem Kunden auf eine Reise, und wenn die in die richtige Richtung geht, ist man erfolgreich.

DIALOG: Im Unterschied zu den erwähnten Start-ups bringen die meisten Industrieunternehmen eine technische Legacy mit, etwa in Form von Geräten im Feld. Inwiefern steigt dadurch für sie die Komplexität?

BZ: Die Komplexität nimmt eindeutig zu. Die Frage ist, an welchen Stellen man sich von der Legacy emanzipiert und z.B. einen neuen Use Case verfolgt. Legacy klingt zunächst einmal negativ. Doch sie bedeutet auch Stabilität und Erfahrung, sie gibt Sicherheit und Vertrauen für unsere Kunden auf dem Weg in die Zukunft. Der zunehmenden Komplexität begegnet man mit der Entwicklung einer Plattform. Die Zukunft liegt nicht darin, alles selbst zu machen. Es geht auch darum, Schnittstellen anzubieten, an denen Kunden selbstentwickelte Applikationen oder bei Bedarf auch ein Start-up andocken können. Dies führt dazu, dass vorangegangene Investitionen nicht umsonst waren und am Ende eine ganzheitlich optimierte Lösung steht.

DIALOG: Für viele klassische Industrieunternehmen dürfte diese Welt der Plattformen und intensiven Kooperationen neu sein. Wie bewerten Sie die Herausforderungen rund um diesen Mindchange?

BZ: Das sind große Herausforderungen. Fast alle Unternehmen betreiben heute Server und nutzen Cloud-Technologien. Doch eine eigene Cloud aufzusetzen, die Cyber Security aufzustellen und die Zugriffsrechte zu managen – das erfordert grundlegend andere Skills. Uns bei Landis+Gyr war klar, dass wir das nicht allein schaffen, denn wir wollten unseren Kunden möglichst schnell eine State-of-the-Art-Lösung anbieten können. Darum haben wir uns mit Google einen Partner für die digitale Transformation an Bord geholt, der uns auf der Reise in die Cloud begleitet. Vor zwei Jahren sind wir gestartet, das Programm läuft insgesamt sieben Jahre. Die Transformation ist immens.

DIALOG: Sie sprachen von „grundlegend anderen Skills". Die passenden Entwickler dafür zu finden ist aufwendig. Könnte man diesen Part auch komplett extern vergeben?

BZ: Man kommt nicht darum herum, die Spezialisten einzukaufen. Auch wir hatten diesen Lerneffekt. Unternehmen wie Google verstehen ihre Technologie und Cloud Services, aber haben nicht unser Domänenwissen, etwa was das Ablesen und Analysieren von Zählerwerten angeht und wann welche Werte abrufbar sein müssen. Man kann sich nicht zurücklehnen und warten, bis die Lösung in der Cloud fertig ist. Sie müssen mit Ihrem Anbieter zusammenarbeiten. Cloud Services werden nach Klicks und Datenraten abgerechnet. Eine schlechte Lösung verursacht für den gleichen Endnutzen einen erhöhten „Traffic" bei den Datenraten oder eine höhere Last bei den Cloud-Ressourcen mit entsprechend höheren Kosten. Sie brauchen internes Know-how, um die Lösungen in der Cloud richtig aufzusetzen und so eine effiziente und sichere Lösung anzubieten.

DIALOG: Das klingt, als bräuchte man Allrounder mit tiefen Technologieverständnis und spezifischen Domänenwissen. Gibt es die überhaupt?

BZ: Ein tiefes Verständnis dafür, was Hyperscaler wie z.B. Amazon können, und wie man das Potenzial am besten nutzt, ist unerlässlich. Darüber hinaus ist zu erwägen, welche Domäne man im Haus behält. Dabei geht es um die Position des eigenen Unternehmens in der Wertschöpfungskette. Es gilt abzuwägen, welchen Value z.B. ein Cloud-Anbieter, eine Consulting-Firma und das eigene Unternehmen bringen. Davon hängt die moderne Softwareorganisation ab, und dieses Verständnis müssen Sie selbst aufbauen. Die Leute, die das können, sind sehr gefragt. Gerade für konservative Unternehmen kann es eine Challenge sein, diese für sich zu gewinnen. Bei Landis+Gyr sind wir in der Lage, mit unseren Softwarelösungen nachhaltiges Energiemanagement zu ermöglichen, und dieser Aspekt hebt uns auch bei der Talentsuche von vielen anderen positiv ab.

DIALOG: Wie kann man Entwickler, die über das passende technische Skill Set verfügen, an industriespezifisches Domänenwissen heranführen?

BZ: Das geht am besten über reale Pilotprojekte. Man kann davon ausgehen, dass sich Kundenbedürfnisse und Anforderungen im Laufe des Projektes ändern. Ein Beispiel: Die Energierechnung hat die meisten Menschen vor einem Jahr kaum interessiert. Als Resultat der derzeitigen Energiekrise möchte jeder wissen, wie hoch die Kosten werden und wie man diese kontrollieren und senken kann. Es gibt weitere Beispiele: Immer mehr Endkunden haben Elektrofahrzeuge und Solarpanels. Das führt zu Fragen wie: „Soll ich jetzt mein Auto laden, weil ich morgen früh zur Arbeit fahren möchte, oder lieber heizen, weil es kalt ist?“

Der Bedarf für Energiemanagementsysteme steigt rasant. Künftige Anforderungen lassen sich nur abschätzen. Aktuell haben z.B. viele Menschen Angst vor einem Blackout. Zur Früherkennung kann man Erfahrungswerte und Algorithmen nutzen. Das funktioniert aber nicht per Lastenheft, sondern über konstantes Co-Development zusammen mit einem Stadtwerk, einem Lead-Kunden, der offen für neue Ideen ist. Durch solche Projekte kann man die Entwickler am besten integrieren.

DIALOG: Gerade die Stadtwerke befinden sich ja momentan in einer Situation extremer Unsicherheit. Inwiefern ist da auch die Softwareorganisation gefordert, damit umzugehen?

BZ: Sie ist strategisch gefordert, mehr Agilität zu bringen. Dazu brauchen wir auch Vertriebsmitarbeiter und Produktmanager, um Kundenbedürfnisse besser zu verstehen. Branchen wie unsere sind an einem sehr sensiblen Hebel. Wenn ein Energieversorger die Algorithmen falsch einstellt, kann das zu Ausfällen führen. Bei einer Entertainment-App können Sie mit „Trial and Error“ arbeiten, bei der Grundversorgung mit Energie und Wasser geht das nicht. Die Versorgungssicherheit muss zu jedem Zeitpunkt gewährleistet sein. Die Branche ist also zu Recht sehr konservativ. Unsere Lösungen können Versorgungsunternehmen und Endkonsumenten dazu befähigen, Energie effizienter und auch nachhaltiger zu nutzen. Die Frage ist: Wie bringen wir die Welten zusammen, gerade im Hinblick auf die heutige Energiekrise? Wie bekommen wir die notwendige Agilität in eine konservative Branche? Dafür muss man die richtigen Leute finden.

DIALOG: Wie stark beeinflussen regulatorische Rahmenbedingungen, z.B. bei Betreibern kritischer Infrastrukturen (KRITIS), die Softwareentwicklung?

BZ: Wir wollen und brauchen vollumfängliche Compliance. Bevor unsere Lösungen live gehen, gibt es aufwendige Abschlusstest mit tausenden einzelnen Cases. Doch während viele die IT auf dem Schirm haben, wird die Operational Technology (OT) oft unterschätzt. Dieser Bereich ist viel komplexer und anfälliger, dort stehen teilweise noch Windows XP Server aus den 1990er Jahren. Auch hier muss man in ein stark reguliertes Umfeld die notwendige Geschwindigkeit hineinbringen. Manche Kunden versuchen, solche Probleme mit einer Start-up-App zu lösen. Aber dass etwas funktioniert und cool aussieht, heißt noch nicht, dass es regulative Anforderungen und Sicherheitsvorgaben erfüllt. Der Weg dahin kann sehr beschwerlich sein. Gleichzeitig stelle ich fest, dass wir in Deutschland teils völlig veraltete und viel zu komplexe Systeme haben. Da werden Spezifikationen seit Jahren immer wieder geändert, oder die Anforderungen sind so widersprüchlich, dass sie nicht wirtschaftlich erfüllt werden können. Daran sind schon einige Firmen gescheitert, andere konnten rechtzeitig aussteigen. Das ist natürlich ein Innovationshemmnis.

DIALOG: Mit Entwicklern kommen Menschen in Unternehmen, die einen völlig anderen Background und teilweise andere Arbeitsweisen haben als die übrige Organisation. Was bedeutet das im Hinblick auf das Thema Führung? Wo entstehen typischerweise Reibungspunkte?

BZ: Wir führen die Software als eigene Business Unit mit den nötigen Prozessen und Freiräumen bei strategischer Kollaboration mit anderen Bereichen. Der Hardware- und Produktionsbereich dagegen ist viel mehr nach der klassischen Wasserfall-Entwicklungsmethode gesteuert. Im Strategieprozess haben wir zunächst die Kernsoftware, die sich eher langsam entwickelt. Zusätzlich arbeiten wir an neuen Applikationen, die innerhalb der Softwareorganisation höhere Freiheitsgrade haben. Dabei geht es auch um Kapitalallokation und Budgets und darum, wie wir unser Kerngeschäft weiterentwickeln. In einem Industrieunternehmen bedeutet das eine tiefe Kulturänderung. Das klassische Geschäft ist typischerweise gut vorauszuberechnen. Bei neuen Softwareservices und Analytics-Plattformen weiß man dagegen nicht, wie das Geschäft in fünf Jahren läuft, ob ein Case funktioniert oder nicht. Wir gehen von bestimmten, grundlegendenden Entwicklungen aus und haben eine lange Liste spannender Use Cases, basierend auf einem tiefen Know-how der Branche und den engen Kundenbeziehungen, die wir über Jahre aufgebaut haben. In diese Felder investieren wir. Dazu braucht man Flexibilität und muss ein Investment ggf. wieder zurückfahren, um am Ende vermarktbare und natürlich auch profitable Lösungen offerieren zu können.

DIALOG: Aufbau und Ausrichtung des Softwaregeschäftes hängen also stärker als im klassischen Geschäft davon ab, mit welchen Szenarien man plant?

BZ: Auf jeden Fall. Sie müssen den Investments und der Organisation zugestehen, dass Dinge manchmal nicht funktionieren. Dazu brauchen Sie eine funktionierende Fehler- und Lernkultur. Zudem sollte man realistisch sein. Manche Use Cases klingen gut, generieren am Ende aber kein Geld. Consumer-Apps lassen sich über den Verkauf von Daten querfinanzieren. Stadtwerke dagegen können keine Zählerdaten verkaufen, und das ist gut so. Zudem ist es in vielen Bereichen gar nicht erforderlich, eine eigene Applikation zu entwickeln, weil man auf bestehende Lösungen zurückgreifen kann. Da geht es eher um Schnittstellen. Im Automobilbereich z.B. haben Player viel Geld in eigene End-User-Apps gesteckt. Doch viele Kunden nutzen stattdessen die Software auf ihrem Smartphone. Die Nähe zu Kunden und ihren Bedürfnissen ist bei der Entwicklung dieser Szenarien also unerlässlich.

DIALOG: Software wird immer häufiger zum Differenzierungsmerkmal. Bedeutet das umgekehrt, dass man heute als reiner Hardwarelieferant gar nicht mehr wettbewerbsfähig ist?

BZ: Entscheidend sind Kundennutzen und Differenzierung. Wir wollen uns im Markt über Software und integrierte End-to-End-Lösungen differenzieren. Allein in Deutschland vertrauen 800 Stadtwerke darauf, dass sie zukünftig eine vollumfassende, funktionierende Lösung erhalten. Darum entwickeln wir eine Plattform mit offenen Schnittstellen, sodass unsere Kunden auch selbstentwickelte Software und Lösungen von Dritten andocken können. Durch dieses Angebot wollen wir Stickyness, also Kundenbindung, generieren. Ansonsten wird man zum reinen Hardwarehersteller. Das ist nicht zwangsläufig schlecht. Doch jedes Unternehmen sollte sich klar machen, wo Potenziale für die eigene Weiterentwicklung liegen und wie man sich für die Zukunft aufstellt.

WIE GELINGT LEAN SOFTWARE DEVELOP-MENT?

Ein Artikel von Josef Wohlrab, ROI-EFESO

Bereits Ende der 1980er Jahre trat die Lean-Philosophie ihren Siegeszug durch die Industrie an. Sie entstammt der Produktionsorganisation von Toyota. Ihr Kern – japanisch „Muda" – ist das stetige Streben nach Vermeidung jeder Art von Verschwendung, was sowohl im Sinne von Effektivität als auch von Effizienz zu verstehen ist.

Auch wenn die Lean-Prinzipien ihren Ursprung in der Industrie haben, wurde schnell deutlich, dass der Erfolg von Toyota nicht allein auf ihrer Anwendung in der Produktion beruht. Vielmehr prägte das schlanke Denken und Arbeiten auch alle anderen Unternehmensbereiche, so auch die Produktentwicklung. Bereits 2003 befassten sich die IT-Forscher und Programmierer Mary und Tom Poppendieck mit der Anwendung der Lean-Philosophie auf die Softwareentwicklung. Dabei hatten sie vor allem die „reine" Softwareentwicklung im Blick. In den vergangenen Jahren ist diese Perspektive weiterentwickelt worden, sodass sich die Idee schlanker Produktentwicklung auch im Kontext komplexer Systeme und aktueller Technologien wie IoT (Internet of Things), Cloud oder Augmented Reality anwenden lässt. Dabei haben sich fünf wesentliche Prinzipien herauskristallisiert, die im Folgenden näher betrachtet werden:

1. Kundennutzen definieren und maximieren
2. Wertstrom identifizieren und Abfall beseitigen
3. Wertstrom fließen lassen
4. Team stärken
5. Kontinuierlich lernen und verbessern

Kundennutzen definieren und maximieren

Für jedes Produkt ist der wahrgenommene Kundennutzen im Vergleich zu den Wettbewerbsprodukten das entscheidende Erfolgskriterium. Eine Entwicklung am Nutzer vorbei ist daher die schlimmste Form der Verschwendung. Um dies zu vermeiden, gilt es, den tatsächlich wahrgenommenen Wert frühestmöglich zu validieren und weitere Kundenbedürfnisse zu ermitteln. Dies gelingt über die schnelle Erzeugung eines Minimum Viable Product (MVP) und das Einholen von Kundenfeedback. Die Produktveröffentlichung aufzuschieben, um eventuell mehr Features bringen zu können, erhöht deutlich das Risiko, den durch den Kunden wahrgenommenen Nutzen oder das Window of Opportunity im Markt zu verfehlen, weil Wettbewerber alternative Angebote in den Markt bringen. Diese müssen noch nicht einmal einen vergleichbaren Funktionsumfang oder die gleiche Qualität aufweisen, wenn sie entweder auf einen dringenden Bedarf treffen oder die Wechselwilligkeit der Kunden gering ist. Geschwindigkeit in Form von schnellem Prototyping und engzyklischer Kundeneinbindung ist also entscheidend. Dadurch wird auch das Risiko von Over-Engineering deutlich reduziert.

Um den Kundennutzen schnell zu maximieren und keine Zeit zu verlieren, kommen Methoden wie das A/B Testing zum Einsatz, bei dem gleichzeitig mehrere Varianten entwickelt und getestet werden, sodass durch das Verhalten der Nutzergruppen die bessere Variante erkennbar wird. Diese Methoden erscheinen aus der Perspektive klassischer Produktion wie reine Verschwendung, sind aber im Kontext der Softwareentwicklung genau das Gegenteil.

Wertstrom identifizieren und Abfall beseitigen

Ziel einer Wertstromanalyse und -optimierung ist es, den kritischen Pfad zu identifizieren und bestmöglich zu komprimieren sowie nicht wertschöpfende Aktivitäten zu eliminieren. Der kritische Pfad beschreibt die Mindestdauer der Feature-Entwicklung. Er ergibt sich aus dem Entwicklungsumfang und dem angewandten Entwicklungsmodell bei idealen Konditionen. Bei der Entwicklung von Systemen schließt dies auch die Entwicklung der benötigten Hardware mit ein. Häufig wird die Software nach einem agilen Modell entwickelt und die Hardware nach einem V-Modell. Um eine volle Ausrichtung auf den Wertstrom und den enthaltenen kritischen Pfad zu realisieren, müssen die beiden Welten in einem hybriden Modell verzahnt werden. Da das passende Entwicklungsmodell immer vom unternehmensspezifischen Kontext abhängt, sollten bekannte Modelle wie SAFe (Scaled Agile Framework) oder LeSS (Large Scale Scrum) nicht einfach kopiert, sondern immer an die eigenen individuellen Anforderungen angepasst werden.

Doch die Gestaltung des besten Modells ist wertlos, wenn es nicht gelebt wird. Ein fehlende Durchgängigkeit in der Anwendung hat oft Blindleistungen, Wartezeiten oder Qualitätsprobleme zur Folge. Die Implementierung erfordert deshalb einen klaren Fokus. Die Visualisierung des Wertstroms und konkrete Praxisbeispiele haben sich sowohl für die Optimierung des Wertstroms als auch für dessen Vermittlung bewährt. Um die geplante Entwicklung der Features abzusichern, werden dabei technische Reviews entlang des kritischen Pfads installiert.

Nicht selten werden diese Reviews jedoch überdimensioniert, da sie aufgrund der technischen Komplexität nicht mehr in der Verantwortung und Kompetenz einzelner Personen liegen können. Schlechte oder fehlende Architekturarbeit und ein schlechtes Schnittstellenmanagement (API-Management) können diese Situation weiter verschlimmern. Ein weiteres Phänomen ist der leichtfertige Umgang mit der Softwarereife zu den vereinbarten Meilensteinen, da sich Software ohne augenscheinlich große Kosten anpassen lässt. Dadurch häufen sich technische Schulden an und werden zunehmend zu einem schwer lösbaren Problem. Gut aufgesetzte und gelebte Qualitätspraktiken sowie ein stringenter Umgang mit Abweichungen von der vereinbarten Softwarereife sollten deshalb unbedingt gewährleistet werden.

Wertstrom fließen lassen

Der Wertstrom und das zugrunde liegende Prozessmodell betten sich in die Struktur einer Entwicklungsorganisation ein, mit entsprechenden Berichts- und Entscheidungswegen. Diese Elemente sind idealerweise stimmig zueinander und voll auf den Wertstrom ausgerichtet. Doch Entwicklungsorganisationen haben selten nur einen Wertstrom zu bedienen und sind üblicherweise nicht beliebig schnell und oft veränderbar. Daher ist es logisch, dass beim Streben nach einem globalen Optimum die Rahmenbedingungen für den einzelnen Wertstrom nicht ideal sein können. Der Fachkräftemangel tut dabei ein Übriges, wenn versucht wird, mit einer stark unterbesetzten Mannschaft das volle Produktportfolio zu bedienen. Die unzureichende Priorisierung führt zu einer Überbuchung bzw. Mehrfachzuordnung von Mitarbeitern zu parallel laufenden Projekten und zu einer zu hohen Zahl an Projektübergaben. Dadurch wird die Leistungsfähigkeit der Entwicklungsorganisation drastisch reduziert. Das Management reagiert darauf oftmals mit mehr störenden als helfenden Interventionen, was neben der fehlenden Fähigkeit zu priorisieren ein deutlicher Hinweis für Defizite in der Führungskompetenz ist. Darüber hinaus zwingt ein derart stark überlastetes System die Führungskräfte in einen reaktiven Modus. Präventives, gestaltendes Handeln und ein gutes Risikomanagement bleiben dabei typischerweise auf der Strecke.

Team stärken

Softwareentwicklung erfolgt üblicherweise in agilen Teams. Im Idealfall ermöglicht die Zusammensetzung der Teams ein weitgehend autonomes Bearbeiten der Aufgabenumfänge. Das schließt die Ermächtigung zur Entscheidung, das Empowerment, mit ein, was voraussetzt, dass alle benötigten fachlichen und sozialen Kompetenzen im Team vertreten sind. Die übertragene Entscheidungskompetenz fördert die Eigenverantwortung und steigert die Motivation. Somit können agile Teams in einem dynamischen Umfeld, d.h. bei sich stetig verändernden Anforderungen, besonders produktiv agieren. Die Führungskraft spielt hierbei eine komplett andere Rolle als im klassischen Modell. Ihre wesentliche Aufgabe ist es, optimale Rahmenbedingungen zu schaffen sowie gleichzeitig die Governance zu stärken und die Accountability sicherzustellen.

Darüber hinaus gibt es in einem agilen Set-up Rollen, die für die Performance der Teams von besonderer Bedeutung sind: Architects, Product Owners und Agile Masters bzw. Scrum Masters. Der Architekt ist dabei als inhaltlich tragende Rolle hervorzuheben, da die Architektur nicht nur für die Lebensdauer von Software entscheidend ist, sondern auch für die verteilte Entwicklung, d.h. die Parallelisierung der Entwicklungsaktivitäten. Um diese Rolle einnehmen zu können, braucht es sowohl eine hohe fachliche als auch Führungskompetenz. Der Product Owner richtet die Softwareentwicklung maximal auf den Kundennutzen aus. Der Agile Master hat die Aufgabe, die Zusammenarbeit zu optimieren. Er verantwortet die Einhaltung und Verbesserung des agilen Entwicklungsmodells sowie die Beseitigung von Hindernissen („Impediments“ in der agilen Nomenklatur) jeglicher Art.

Kontinuierlich lernen und verbessern

Kontinuierliches Lernen und Verbessern ist ein integraler Bestandteil agiler Frameworks. Das Streben nach kontinuierlicher Verbesserung kann jedoch nicht erzwungen werden, wie zahlreiche Praxiserfahrungen zeigen. Eine gute Chance auf nachhaltigen Erfolg besteht erst dann, wenn die kontinuierliche Verbesserung zu einem integralen Bestand der Unternehmenskultur wird. Dafür sind insbesondere eine offene Fehlerkultur und ein kollaboratives Mindset starke Indikatoren. In einer offenen Fehlerkultur setzen sich alle Beteiligten aktiv und konstruktiv mit Fehlern auseinander, um von ihnen zu lernen und sich weiterzuentwickeln. Die Offenheit setzt hierarchieübergreifend ein großes Vertrauen voraus. Bei der Zusammenarbeit steht das übergeordnete Ziel und nicht die individuellen Ziele im Vordergrund. Man unterstützt sich gegenseitig und teilt proaktiv Wissen, um jenseits hierarchischer Selbstverständnisse und Silo-Strukturen schneller und besser zum gemeinsamen Ziel zu kommen. Gerade in der Softwareentwicklung stellt die Pflege einer solchen Fehler- und Zusammenarbeitskultur oftmals eine Herausforderung dar, die sich im multikulturellen Umfeld zusätzlich verschärfen kann. Teambuilding und vertrauensbildende Maßnahmen sind daher sehr wichtig, kommen jedoch bei hohem Lieferdruck mit einer unterbesetzten Mannschaft oftmals zu kurz. Neben den „soften“ Themen, die den wesentlichen Stellhebel darstellen, lassen sich auch Messgrößen definieren, anhand derer gesteuert werden kann. Diese sind aber immer nur so gut wie die zugrunde liegende Datenbasis und wirken bei schlechter Wahl kontraproduktiv.

Vor dem Hintergrund dieser Betrachtung wird deutlich, dass es viele Stellhebel für Lean Software Development gibt. Sie alle dienen dazu, Verschwendung und Verbesserungspotenziale zu erkennen. Externes Feedback erweist sich dabei als besonders hilfreich, um einseitigen Perspektiven und Betriebsblindheit entgegenzuwirken und den Blick für das Ganze nicht zu verlieren.

Gespräch mit Martin Nanke, Managing Director der OBI E-Commerce GmbH, über den Aufbau digitaler Organisationen

„JEDE MINUTE, DIE WIR IN DAS HIRING INVESTIEREN, IST EINE SINNVOLL INVESTIERTE MINUTE."

DIALOG: Herr Nanke, wie startet man als klassisches Unternehmen den Weg in die Digitalisierung?

MN: Am Anfang dieser Reise steht immer ein sehr klares Zielbild, das auch in die Gesamtstrategie und -vision des Unternehmens eingebunden ist. Man sollte kein Digitalgeschäft aufbauen, nur um ein Digitalgeschäft zu haben, sondern man braucht ein klares Verständnis darüber, welchen Beitrag die Digitaleinheit im Kontext der Gesamtstrategie und -vision des Unternehmens leisten soll. Dieses Zielbild muss man mit Blick auf den Aufbau der Organisation und das Hiring der Expertinnen und Experten runterbrechen und klar kommunizieren. Denn beim Aufbau der Digitaleinheit steht man unvermeidlich im War for Talents mit Unternehmen, die es erstmal leichter haben; die vielleicht nicht aus dem klassischen Geschäft kommen; die schneller skalieren und Ressourcen mobilisieren können; die eine hohe Flexibilität haben. Klarheit und Fokus sind deshalb sehr wichtig, um Mitarbeitende zu gewinnen. Und am Ende sind es die Kolleginnen und Kollegen, die den Unterschied machen.

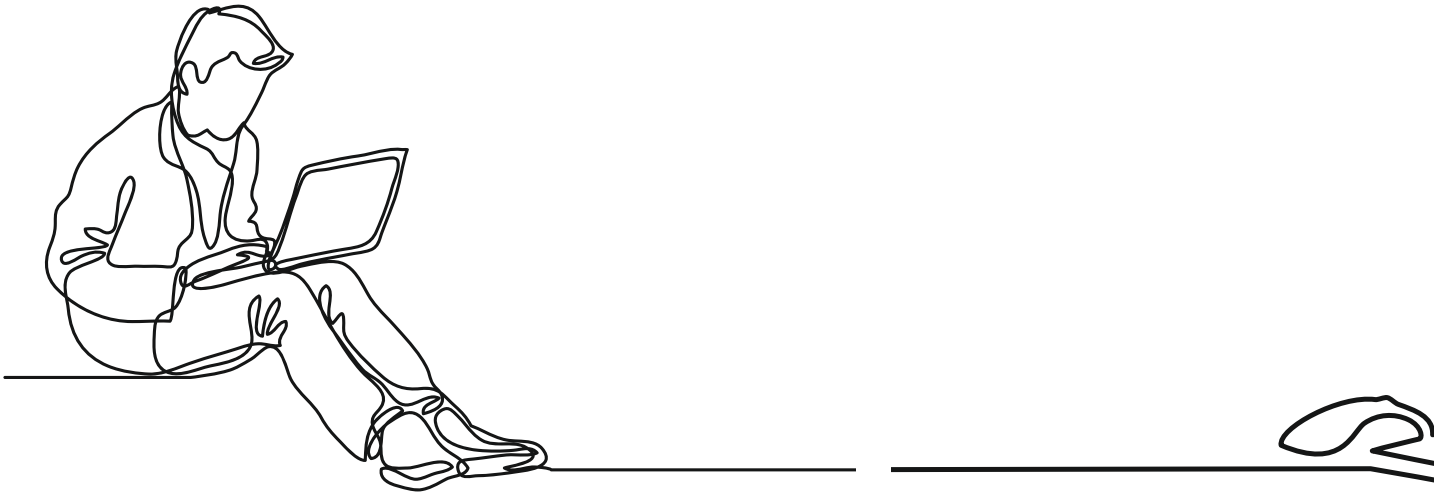
DIALOG: Wie sollte man konkret beim Aufbau eines solchen Digitalteams vorgehen?

MN: Es ist ganz entscheidend, am Anfang bereits ein kleines, schlagkräftiges Team zu haben, das mit dem Aufbau der Digitaleinheit beginnt. So wie wir das bei OBI mit unserer Transformationseinheit OBI NEXT getan haben. Da braucht es einen Mix aus Expertinnen und Experten, die das Unternehmen und das Geschäftsmodell kennen, und solchen, die ein sehr klares Verständnis für digitale Geschäftsmodelle haben – eine diverse Mannschaft mit unterschiedlichen Skills, darunter Kompetenzen, die neu sind, die typischerweise im Unternehmen aktuell nicht zu finden sind. Wir betreiben bei OBI bewusst einen großen organisatorischen Aufwand und investieren sehr viel Zeit, um die für uns richtigen Leute an Bord zu holen. Dafür gibt es ein klares unternehmerisches Commitment. Dieses Commitment gilt bei OBI natürlich nicht nur für klassische Digital-Teams, sondern für alle Bereiche, z.B. auch bei der digitalen Transformation unseres Category Management.

DIALOG: Reicht das, um die Talente für sich zu begeistern, an denen auch die Digital Champions interessiert sind?

MN: In der heutigen Zeit Top-Kräfte für Digitalisierungsthemen zu finden ist wahnsinnig schwer, das lässt sich nicht wegzaubern. Der Anfang ist am schwersten, denn gute Leute folgen guten Leuten. Deswegen ist es so enorm wichtig, ein klares Zielbild und auch ein kleines Top-Team zu haben, über das man die weitere Mannschaft aufbaut. Ich sehe da verschiedene Stoßrichtungen:

Einmal ist es wichtig, als Unternehmen sicherzustellen, dass man eine Kultur und ein Umfeld hat, in dem sich Mitarbeiterinnen und Mitarbeiter mit einem starken digitalen Fokus verorten und wohlfühlen. Das betrifft das Miteinander, aber auch die Rahmenparameter der



Zusammenarbeit, die Rolle des Büros, die gelebten Rituale. Man muss Gemeinsamkeit neu interpretieren. Es geht darum, Teams regelmäßig zusammenbringen, die gemeinsamen Ziele und Werte lebendig halten. Über eine starke, inspirierende Kultur können wir uns von Wettbewerbern differenzieren.

Den anderen Faktor habe ich bereist genannt. Wir glauben, dass jede Minute, die wir in das Hiring investieren, eine sinnvoll investierte Minute ist. Deswegen schaffen wir die Möglichkeit, dass die Kandidatinnen und Kandidaten maximal viele Kolleginnen und Kollegen kennenlernen und mit uns viel Zeit verbringen können. Dass sie sehen können, dass wir uns wirklich für sie interessieren und wie viel Substanz hinter unserer Vision steckt. Und wir haben dann wiederum auch viel Zeit, um Kandidatinnen und Kandidaten gut kennenzulernen. Wenn wir uns für jemanden entscheiden, machen wir das sehr bewusst, und die Kandidatinnen und Kandidaten können sicher sein, dass wir sie unbedingt im Team haben wollen.

DIALOG: Sie haben eingangs die große Bedeutung des Zielbildes erwähnt. Wie lässt sich sicherstellen, dass in diesem die alte Welt nicht einfach fortgeschrieben wird?

MN: Das Gefährlichste, was man machen kann, ist, das bestehende Geschäftsmodell einfach in ein digitales zu übersetzen, ohne zu berücksichtigen, dass digitale Wirkungskräfte ganz anders laufen als im stationären Geschäftsmodell und mitunter ganz andere Dinge erforderlich sind. Deshalb muss dieses Ziel- oder Leitbild aus meiner Sicht von dem kleinen Kernteam formuliert werden, das ich eben beschrieben habe. Darin muss sich natürlich auch die bestehende Organisation wiederfinden. Aber es gilt, die digitale Vision in den dafür passenden Kategorien und Begriffen zu formulieren. Uns ist wichtig, dass wir ein Team haben, das versteht, wie Unternehmen heute funktionieren, und gleichzeitig auch weiß, was wir brauchen, um in eine digitale Welt hineinzuwachsen und hier mindestens genauso bedeutsam zu werden.

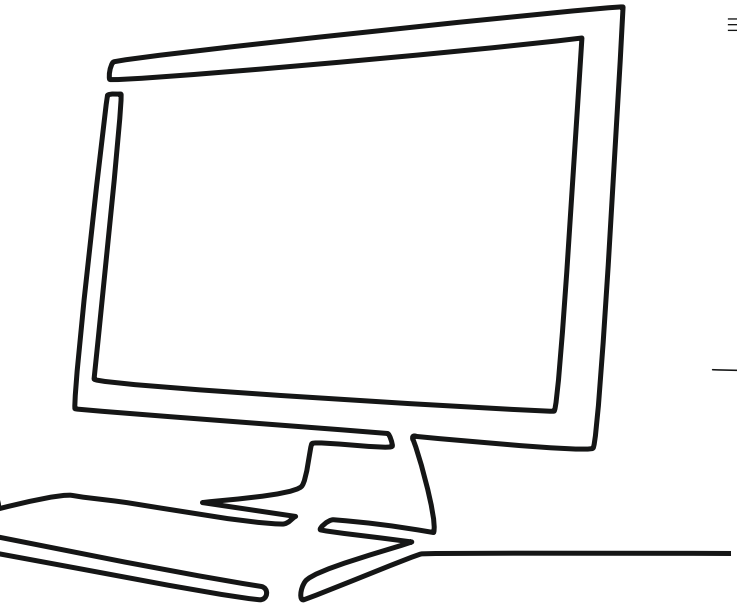
DIALOG: Warum eigentlich darf man das bestehende Geschäftsmodell nicht einfach in die digitale Welt übertragen?

MN: Die Denke, ich verkaufe ein Produkt, aber wie ich es verkaufe, ist eigentlich egal – das funktioniert nicht, weil die Kanäle und das Kundenverhalten sehr unterschiedlich sind. Und es stellen sich für die Kunden im digitalen Kaufprozess ganz andere Fragen. Verstehe ich überhaupt anhand der Produktbeschreibung das Produkt? Ich kann nicht die Verkäuferin oder den Verkäufer meines Vertrauens ansprechen, was mache ich bei Rückfragen? Wie komme ich zu einer Transaktion und wie wickele ich sie ab?

Nehmen wir ein einfaches Beispiel aus unserem Geschäft, eine Bohrmaschine: Zu diesem Produkt habe ich bereits bestimmte Stammdaten, die ich 1:1 in einen Online-Shop packe. Das wird überhaupt nicht funktionieren, weil plötzlich ganz viele Informationen fehlen: eine detaillierte Artikelbeschreibung, die technischen Daten, Produktbilder. Warum? Wenn die Kundinnen und Kunden in den stationären Markt gehen, sehen sie die Bohrmaschine. Sie brauchen keine Produktbilder, sie können das Verkaufsteam fragen, sie können sich beraten lassen.

Sie sind in einer begrenzten, gut strukturierten physischen Welt, in der sie sich gut zurechtfinden. Online habe ich ganz andere Anforderungen. Das betrifft schon den Content, der wichtig ist, um zu verkaufen. Zudem gibt es spezifische Regeln, wie der Weg in den Online-Shop überhaupt gefunden wird. Der nächste Online-Shop ist ja bekanntlich nur einen Klick entfernt. Man muss also bei den Kundinnen und Kunden sowie bei den Customer Journeys starten und verstehen, was sie in einem digitalen Kanal im Gegensatz zum stationären Kanal brauchen, um eine bewusste Kaufentscheidung zu treffen. Das muss genau berücksichtigt und verstanden werden. Viele tradierte und auch erfolgreiche „stationäre Logiken“ greifen nicht im Digital Commerce.

Das Gleiche gilt auch für die Digitalisierung von Arbeitsabläufen – das Gefährlichste, was ich machen kann, ist es, 1:1 meine bisherigen Prozesse über Systeme digitalisieren zu wollen. Im Kontext Category Management schauen wir uns aktuell unsere Prozesse sehr genau an und prüfen, wie wir ggf. adaptieren müssen, um Prozesse bestmöglich zu optimieren.



DIALOG: Wo ist beim Aufbau der digitalen Welt der Sprung, den ein Unternehmen machen muss, am größten?

MN: Ich würde sagen, bei dem wirklich ersthaften Commitment des Unternehmens und des Management-Teams, den Aufbau einer Digitaleinheit mit Nachdruck zu verfolgen. Das klingt erstmal banal. Aber ich glaube, das ist wirklich der große Schritt, den ein Unternehmen gehen muss. Denn am Anfang fließen Investitionen in das digitale Projekt, die eben nicht in das bestehende Geschäft fließen. Das neue Geschäft ist klein, im Vergleich zum bestehenden anfangs kaum relevant. Und da trotzdem mit Überzeugung Gas zu geben, das ist aus meiner Erfahrung einer der größten und wichtigsten Schritte. Wirklich erfolgreiche Unternehmen schauen mit Stolz auf diese neuen Pflänzchen und setzen bewusst ihre besten Mitarbeiterinnen und Mitarbeiter auf die neuen Themen. Wer das schafft, hat einen großen Schritt in der Transformation schon getan. Wenn ein etabliertes, großes Unternehmen erstmal in eine Richtung läuft, dann sehr, sehr häufig mit einer extremen Kraft.

DIALOG: Wie steuert man aus der Managementperspektive diesen Prozess, ist er einmal angestoßen?

MN: Ist das Commitment hergestellt, bedarf es einer grundsätzlichen Fehlertoleranz und der Bereitschaft zu Experimenten. Man findet ja selten gleich im ersten Schritt das richtige digitale Geschäftsmodell, trifft nicht direkt die Erwartungen der Kundinnen und Kunden. Hat man im Transformationsprozess keine Fehler gemacht, hat man wahrscheinlich auch zu wenig ausprobiert. Es gibt allerdings auch bestimmte Dinge, bei denen man sich keine Fehler erlauben sollte – beim Hiring, bei Prozessen, bei der Organisation. Da muss man sehr klar sein.

Gleichzeitig geht es auch darum, den Mut zu haben, Dinge einzustellen, an die man vielleicht mit Leidenschaft und Geld herangegangen ist, die aber dennoch keine Kundenakzeptanz erreichen. Die Einstellung: „Jetzt haben wir sechs Monate daran gearbeitet. Ich bin für dieses Thema verantwortlich. Das muss jetzt funktionieren. Ich darf das gar nicht scheitern lassen.“, darf nicht Einzug halten. Ich glaube, die große Kunst an der Stelle ist, den richtigen Moment zu erkennen. Auf der einen Seite genügend Zeit und Geduld zu haben, eine Idee zu entwickeln, und gleichzeitig den richtigen Zeitpunkt zu erkennen, eine Idee auch fallenzulassen und etwas Neues auszuprobieren.

DIALOG: Lassen Sie uns abschließend noch einmal die Spannung zwischen Distanz und Nähe zur Kernorganisation betrachten. Wie wichtig ist diese Frage für das Team, das die Digitaleinheit aufbaut?

MN: Das ist ein ganz wichtiger Erfolgsfaktor im Sinne der Gesamtunternehmensstrategie. Eine Digitaleinheit darf kein Alien im Konzern sein, sie muss auf den Kern der Unternehmensziele einwirken können. Gleichzeitig birgt eine zu frühe Vernetzung auch Gefahren. Es wird unvermeidlich viel hinterfragt, es werden möglicherweise zu viele Fragezeichen gesetzt, Freiräume und Geschwindigkeit gehen verloren. Auf Managementebene muss für eine Verbindung zum Kerngeschäft gesorgt werden, sodass Touchpoints zum Rest der Organisation entstehen und dort ein Austausch stattfinden kann. Der darf aber auf keinen Fall dazu führen, dass die digitale Einheit nicht in ihrer Taktung an der inhaltlichen Agenda arbeiten kann. Diese kleine Einheit muss schon Power entwickeln und einen Teamspirit sowie ein volles Commitment auf das Thema. Ich bin sehr froh, dass wir bei OBI dieses „Halten der Balance“ zwischen Zentrifugal- und Zentripetalkräften sehr gut beherrschen.

„LOW-CODE KANN EIN UNTERNEHMEN SEHR VIEL SCHNELLER MACHEN."

Florian Rühl, Vorstandsmitglied des Low-Code-Plattform-Anbieters Simplifier, im Interview.

DIALOG: Herr Rühl, seit Forrester den Begriff „Low-Code" prägte, hat der Ansatz schnell an Popularität gewonnen. Simplifier war schon vorher auf dem Markt. Was ist Low-Code und was macht die Entwicklung attraktiv?

FR: Wir sind mit unserer Plattform ungefähr zur gleichen Zeit gestartet – und haben uns mit dem Begriff „Low-Code" sehr wohlgefühlt. Allerdings muss man heute sehr klar unterscheiden, was wirklich eine Low-Code-Plattform ist und was schlicht die Verwendung eines Hype-Begriffs. Bei Low-Code geht es nicht einfach darum, einzelne Felder zu konfigurieren oder Workflows aufzubauen: Eine Low-Code Plattform leistet deutlich mehr.

Welchen Vorteil Low-Code bringt, zeigt sich am Vergleich mit herkömmlicher Softwareentwicklung. Dort wird der Quellcode einer Applikation mit einer bestimmten Programmiersprache erstellt. Dafür braucht es Experten, die diese Programmiersprachen beherrschen. Doch bei diesen Experten herrscht im Markt ein dramatischer Mangel, der täglich zunimmt. Dadurch kommt es bei der Softwareentwicklung zu langen Projektlaufzeiten und hohen Kosten.

Vor diesem Hintergrund kann Low-Code seine Vorteile ausspielen. Denn dabei wird der Source Code eben nicht Zeile für Zeile erzeugt, sondern über die Verwendung vorgefertigter Elemente zusammengestellt, die in der Plattform mitgeliefert werden. Das gilt nicht nur für die Erstellung der Benutzeroberfläche, sondern auch für die Applikationslogik und die Integration in eine bestehende Systemlandschaft. Die Entwicklung erfolgt dabei unabhängig von dem darunter liegenden System oder Use Case. Kurz gesagt: Low-Code ermöglicht eine grundlegende Anwendungsentwicklung über grafische Oberflächen. Dieser Ansatz eröffnet darüber hinaus auch die Möglichkeit, Menschen außerhalb der IT, der klassischen Entwicklerteams zu befähigen, Applikationen mit Low-Code zu erstellen.

Wenn Unternehmen also Low-Code einführen, geht es darum, effizienter und schneller zu werden, fachliche Anforderungen

zügig in Software zu übersetzen und enger zusammenzuarbeiten, als das heute der Fall ist. Wir sehen Low-Code deshalb auch als Brückentechnologie, die Fachbereich und IT-Experten an einen Tisch bringt und die gemeinsame, konfigurative Erstellung notwendiger Anwendungen ermöglicht.

DIALOG: Wie sehen Sie den Unterschied zwischen Low-Code und No-Code?

FR: Im Prinzip versprechen No-Code-Plattformen, dass jeder Anwender auch ohne jegliche technologische Skills Applikationen erstellen kann. Wir sehen das Thema jedoch kritisch – gerade im industriellen Kontext. Denn typischerweise sind Anwendungen, die in No-Code-Plattformen aufgebaut werden, nicht in die bestehende Systemlandschaft integrierbar. Wenn man bspw. schlicht ein Formular erstellen möchte, kann No-Code ein interessanter Ansatz sein. Low-Code erweitert die Möglichkeiten aber signifikant, da bei Bedarf eine Individualprogrammierung vorgenommen werden kann. So werden hochkomplexe und einfache Nutzungsszenarios gleichermaßen abgedeckt.

DIALOG: Sie sehen die Grenzen von No-Code im komplexen Enterprise-Umfeld?

FR: Ja. Wenn es um Enterprise-Anwendungen geht, braucht es eine tiefe, nahtlose Integration. Die Applikationen müssen sicher auf allen relevanten Endgeräten laufen. Low-Code-Plattformen müssen deshalb auch für individuelle Code-Ergänzungen bzw. Erweiterungen offen sein. Insbesondere im Kontext von Industrie 4.0 nutzen Unternehmen eine Vielzahl von Systemen und Datenquellen wie etwa das Enterprise Ressource Planning (ERP), das Customer Relationship Management (CRM), das Manufacturing Execution System (MES), Maschinendaten und sehr spezifische Schnittstellen, die eine hochprofessionelle Programmierung erfordern. Das ist nichts, was eine Fachabteilung umsetzt. Ablösen kann man diese Systeme aber auch nicht – dafür sind sie zu komplex, zu wichtig, die Investitionen dafür wären extrem hoch und die Zeitverluste einfach nicht vertretbar. Die Fähigkeit, die auf einer Low-Code-Plattform aufgebauten Applikationen in diese vorhandenen Strukturen zu integrieren, ist also ein absolutes Muss – ein zentraler Baustein der Enterprise-App-Erstellung.

DIALOG: Gibt es auch organisatorische Voraussetzungen für den Erfolg von Low-Code-Plattformen?

FR: Hier gilt der Grundsatz „People – Process – Technology“. Eine Low-Code-Plattform ist erstmal nur eine Technologie. Für den erfolgreichen Einsatz von Low-Code, ist es notwendig zu wissen, welche Kompetenzen und Fähigkeiten die Menschen bereits haben, die mit dem Tool arbeiten sollen, und welche sie noch zusätzlich aufbauen müssen. Zusätzlich stellt sich die Frage: Sind bereits moderne Prozesse etabliert? Wird schon agil gearbeitet oder sind Lastenhefte immer noch der Stand der Dinge?

Low-Code-Plattformen fördern häufig neue digitale Geschäftsmodelle und Services, denn sie unterstützen sehr effektiv dabei, die Time-to-Market zu verkürzen und mit geringerem Ressourceneinsatz schnell auf Änderungen zu reagieren. Sie können diese Wirkung aber nur entfalten, wenn sie auf effiziente und agile Prozesse aufsetzen. Nicht zuletzt geht es jedoch auch um das Mindset in der IT. Da herrscht teils noch die Auffassung, dass alles individuell und von Null aufgebaut

> Individualisierungen erfolgen hingegen durch anschlussfähige Systeme wie Low-Code-Plattformen.

werden muss. Dieses Mindset muss sich weiterentwickeln, damit der Low-Code-Ansatz überhaupt funktionieren kann.

DIALOG: Verändert sich dabei auch die Rolle der IT im Unternehmen?

FR: Davon bin ich überzeugt. Was aber keinesfalls bedeutet, dass die Hoheit über die Applikationen in die Fachbereiche wandert – im Gegenteil. Die Kontrolle über die eingesetzten Plattformen, die Governance, muss trotzdem weiter in der IT liegen. Sie muss dafür Sorge tragen, dass es keinen Wildwuchs gibt, dass bereichsübergreifend Best Practices genutzt werden, dass Wissen und bereits entwickelte Module wiederverwendet werden. Oder die IT muss auch dafür sorgen, bestehenden Wildwuchs einzugrenzen, um dann nach der Konsolidierung zukunftsfähige und effektive Technologiestandards zu setzen. Unsere Plattform kommt ziemlich oft in solchen Projekten zum Einsatz.

Gleichzeitig wandelt sich die Rolle der IT vom Dienstleister der Fachabteilungen zum Innovator. Sie stellt Technologien zur Verfügung und befähigt Menschen im Unternehmen, damit umzugehen. Ganz konkret: Das Konzept Citizen Development erfordert ein intensives Engagement seitens der IT. In diese Rolle muss die IT sich jedoch erst einfinden und die neuen Aufgaben annehmen. Das passiert nicht von heute auf morgen.

Neben der IT-Abteilung verändert sich natürlich auch das Ökosystem externer Dienstleister: Der Bedarf an externen Ressourcen sinkt zwar, aber sie kommen gezielt für hochkomplexe und kritische Prozesse zum Einsatz. Diese Entwicklung korrespondiert auch mit dem Mind Change der großen Softwarehäuser. Diese konzentrieren sich zunehmend auf ihre Kernbereiche. Individualisierungen erfolgt hingegen durch anschlussfähige Systeme wie Low-Code-Plattformen.

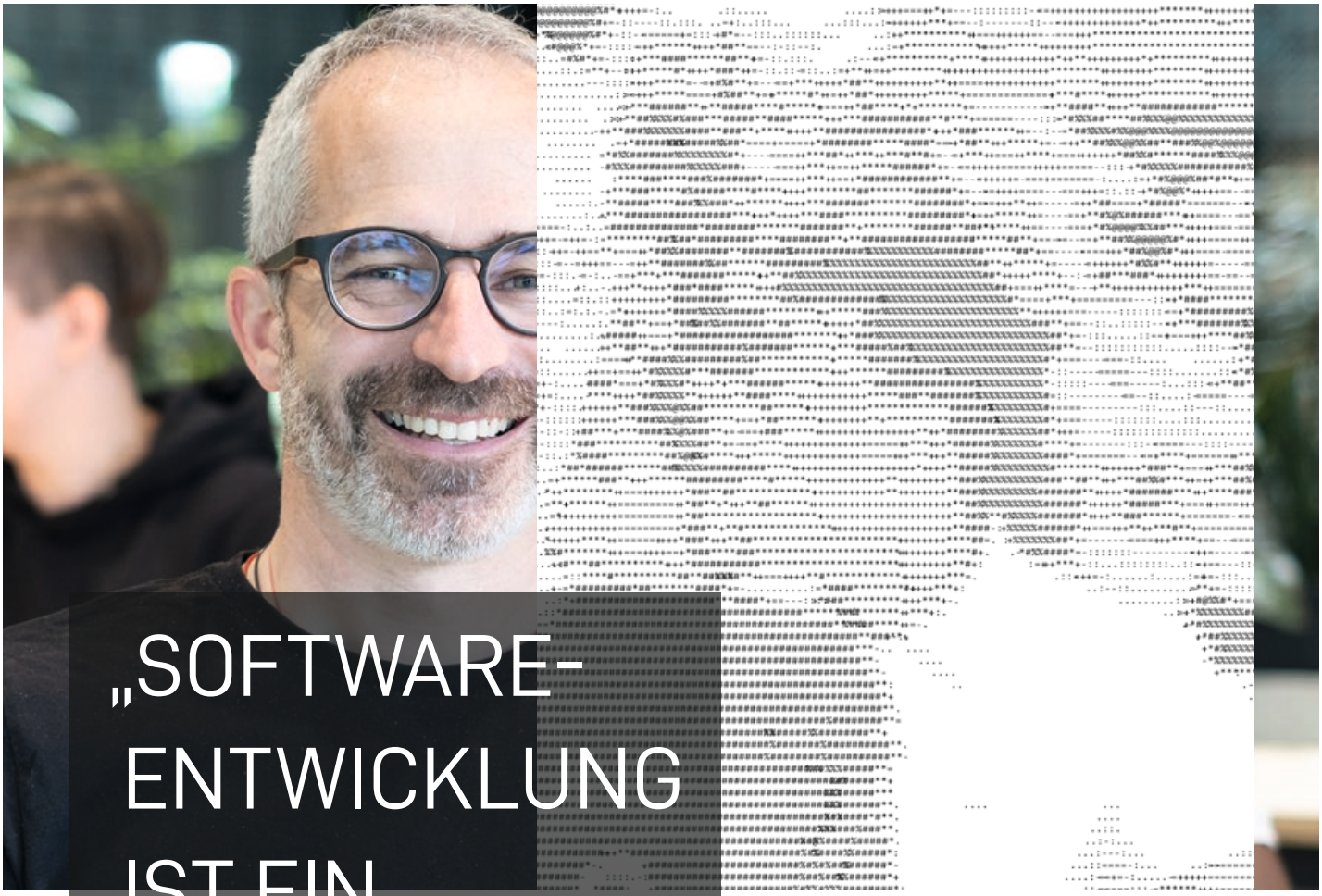
DIALOG: Digitale Souveränität wird zu einem zunehmend wichtigen Faktor für die Zusammenarbeit mit Softwareanbietern – sowohl im öffentlichen Sektor als auch in der Industrie. Begegnet Ihnen das Thema auch im Hinblick auf Low-Code?

FR: Ja, das ist ein essenzielles Thema und ein sehr wichtiges Entscheidungskriterium für unsere Kunden. Stichwort: Sourcecode Ownership. Wir setzen hier auf Open Source. Unsere Kunden können deshalb die mit uns entwickelten Applikationen auch abseits unserer Plattform weiternutzen. Es sind lediglich geringe Anpassungen im Code notwendig. Aber auch die Ownership der aufgebauten Prozesse ist relevant. Letztlich ist das eine Philosophiefrage der Plattformhersteller.

DIALOG: Die Low-Code-Plattform von Simplifier ist nun seit gut fünf Jahren auf dem Markt. Was ist Ihre Vision für die nächsten fünf Jahre?

FR: Wir sehen uns als europäischer Low-Code-Marktführer, insbesondere für die produzierende Industrie und komplexe Enterprise-Bereiche, die integrierte Business-Applikationen brauchen. Gleichzeitig arbeiten wir auch daran, jenseits von Europa in den wichtigen Märkten präsent zu sein. Hierzu gibt es bereits erste Erfahrungen mit Kunden, die unsere Plattform global einsetzen.

Herr Rühl, auf diesem Weg wünschen wir Ihnen viel Erfolg und bedanken uns für das Gespräch!



„SOFTWARE-ENTWICKLUNG IST EIN KREATIVER PROZESS."

Ein Gespräch mit Vincent Stüger, Office of the CTO, Dynatrace.

DIALOG: Herr Stüger, welchen Impact hat Software auf Menschen und Organisation?

VS: Je mehr das Produkt Software wird, desto mehr löst es sich von physischen Dingen, die gewisse Prozesse und Arbeitsweisen vorgeben. Und auch die Geschwindigkeit wird eine völlig andere. Physische Produkte haben bestimmte Fertigungszyklen, die Vorlaufzeiten erfordern. Einen Fabrikbau kann man immer noch nicht viel schneller hinstellen als vor ein paar Jahren. In der Softwareentwicklung ist das anders, wobei Plattformen, Technologien und Modulbaukästen den Prozess immer weiter beschleunigen. Der Umsetzungszyklus von einer Idee zu einem Minimum Viable Product wird immer kürzer, bei Services ist die Entwicklung nochmal schneller. Diese Dynamik hat auf die Menschen und die Organisation sehr großen Einfluss.

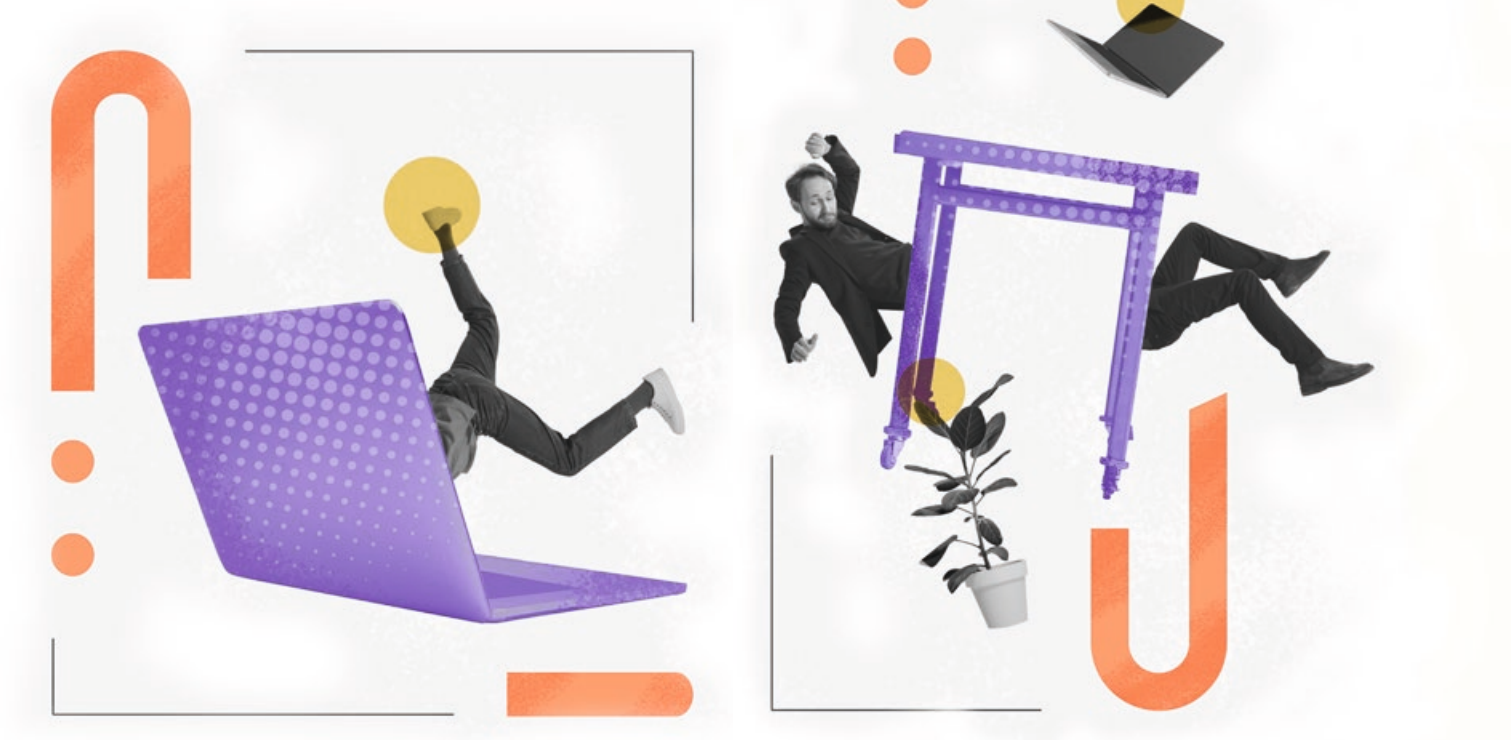
DIALOG: Eine hohe Geschwindigkeit hat mehrere Implikationen – Passagiere sind ihr ein Stück weit ausgeliefert, Fahrer müssen ab einer gewissen Geschwindigkeit überproportional viel Energie aufwenden

VS: Die entscheidende Frage ist immer: „Was ist wichtig?" Dafür braucht es einen guten Kommunikationsfluss. Bei hoher Geschwindigkeit brauche ich eine Vision und die daraus abgeleitete Strategie, die beim Mitarbeiter ankommt. Jeder muss klar wissen, wo die Reise hingeht, warum sie dorthin geht und was die Rahmenbedingungen sind. Bei hoher Geschwindigkeit muss man dafür sorgen, dass das jeder versteht. Darum sind Softwareorganisationen sehr flach, oft mit nur zwei Hierarchieebenen im klassischen Sinn. Das erfordert neue Kommunikationsformen, Klarheit, eine klare Verbindung zwischen Strategie und Vision, effektives Alignment, ohne das Zielsystem zu eng zu machen.

Dann bekommt man die zum Problemlösen nötige Kreativität. Aber wenn man das Zielsystem zu weit macht, endet man im Chaos. Dafür sind diese „schnellen" Organisationsformen sehr anfällig.

DIALOG: Warum entsteht dieses Chaos?

VS: Alignment passiert ja selten am Frühstückstisch, wie bei einem Start-up. In einer großen Organisation mit Hunderten Entwicklerteams, die an einem Produkt arbeiten, muss ich anders dafür sorgen. Flache Hierarchien generieren keine Struktur, die das Alignment sichert. Ich muss sicherstellen, dass die Leute abgestimmt arbeiten und nicht einer links, weil er nicht weiß, dass der rechts



das schon baut, das noch einmal baut und dann auch noch anders. Es geht darum sicherzustellen, dass die Leute ausgerichtet an einer gemeinsamen Vision arbeiten, die sich auch noch immer wieder ändern kann. In diesem Umfeld ist ein Strategieprozess eigentlich gar kein Prozess mehr, sondern passiert laufend und verändert sich immer wieder mit neuen Kunden und Technologien. Das muss man absichern, dabei aber auch Autonomie erlauben, die für die Kreativität unabdingbar ist. Dabei stehen Autonomie und Steuerung prinzipiell in einem Widerspruch, den ich durch intensive und inklusive Kommunikation auflösen muss.

DIALOG: Wie groß ist die Gefahr bei einer Softwareentwicklung, dass man aufgrund dieser hohen Geschwindigkeit immer wieder zurück muss? Man dreht ein riesiges Rad, hunderttausend Leute, alle schnell, agil – und merkt dann aber nach einer recht langen Zeit, irgendwo ist man falsch abgebogen oder an der Station irgendwie komplett vorbeigefahren.

VS: Da ist jetzt die Frage, ob es die Geschwindigkeit ist oder die notwendige Arbeitsweise. Agile Setups funktionieren bis zu einem gewissen Grad über einen Selbststeuerungsmechanismus, der stark vom Feedback, vor allem Kundenfeedback, lebt. Nur gibt es ab einer gewissen Komplexitätsebene den Kunden, der alles sieht, auch nicht mehr. Also müssen mehrere Feedbackzyklen synchronisiert werden, das ist die eine Herausforderung. Die andere liegt in der Superkomplexität solcher Systeme. Es geht ja nicht mehr darum, dass Hundert Kunden zehn Funktionen einer App testen – nach dem Motto „geht, geht nicht". Das kann man organisieren.

Aber ein Produkt, das global verteilt läuft, ist in Gänze irgendwann nicht mehr beurteilbar. Hochverteilte Systeme sprengen die menschliche Denkmöglichkeit. Das kann man nur noch mit Werkzeugen betrachten. Und darum ist die Komplexität der Systeme verbunden mit der Geschwindigkeit, in der sie sich ständig ändern, und der Virtualität, der Nichtgreifbarkeit, eine Mischung, die uns an Grenzen bringt. Das birgt Risiken und steht oft an der Kante zum Chaos.

DIALOG: Braucht die Softwareentwicklung Menschen, die das alles aushalten können, die auch so schnell surfen können? Es kann ja nicht nur die Frage der fachlichen Ausbildung sein.

VS: In einer Softwareorganisation sind Fünfjahrespläne, in Stein gemeißelte Visionen und langfristig stabile Teamstrukturen vollkommen unmöglich. Man braucht deshalb vor allem Menschen, die sehr gut kommunizieren können, was gerade im technischen Umfeld weder einfach noch selbstverständlich ist. Die in einem dynamischen Setup Teams formen, verändern und führen können und dafür sorgen, dass eine Strategie, die ja in sich ebenfalls dynamisch ist, richtunggebend bleibt.

DIALOG: Wie kann man angesichts dieser Dynamik und Komplexität die Effizienz in der Softwareentwicklung beurteilen und verbessern?

VS: Ich glaube, dass Effizienz in der Softwareentwicklung anders zu betrachten ist als in der klassischen Produktion. Allein schon, weil bei Software Fertigung und Entwicklung quasi immer miteinander verzahnt sind. Das Produkt entsteht, während man es denkt, es ist hundertprozentig „mind-made", der Prozess ist in weiten Teilen nicht automatisierbar. Das heißt, wenn Menschgen motiviert sind, einen guten Informationsfluss und eine gute Ausrichtung haben, trägt das sehr stark zur Effizienz bei. Messen kann ich das vor allem retrospektiv, etwa anhand der Frage, wie schnell man Themen bestimmter Komplexität durchkriegt. Dann kann ich den Output von Teams bewerten.

Die Problematik entsteht häufig durch Redundanzen, weil Teams parallel an den gleichen Themen arbeiten und Dinge immer wieder neu erfunden werden, z.B. weil eine transparente Plattform fehlt, ein Baukasten, auf den alle zugreifen, und weil die Kommunikation nicht stimmt. Das ist der eine Fall. Der andere, deutlich häufigere Fall ist: Man baut einfach etwas Falsches, was durch die hohe Geschwindigkeit noch verstärkt wird.

DIALOG: Dann lassen sich in der Softwareentwicklung Effizienz und Effektivität gar nicht scharf trennen?

VS: Die beiden Themen überlagern sich. Softwareentwicklung ist ein stark kreativer Prozess. Und die Effizienz kreativer Prozesse lässt sich grundsätzlich nicht nur schlecht messen – auch der Wert solcher Messungen ist nicht immer eindeutig. Weil

man kreative Prozesse nicht beliebig beschleunigen und um jeden Holzweg und jede Redundanz bereinigen kann. Letztlich dominiert die Frage, ob man das Richtige macht. Deswegen sind ja die Klärung der Zielrichtung, die Kommunikation, das Alignment so wichtig.

Man kann und muss natürlich vermeiden, dass man auf einen völlig falschen Weg gerät oder einen kompletten Overlap bekommt. Aber im Detail muss man akzeptieren, dass auch mal zwei Leute die gleiche Idee haben und sie auch umsetzen. Und vor allem muss man die Schwierigkeit meistern, eine oft sehr abstrakte und vage Anforderung so gut zu formulieren, dass die Richtung für die Entwickler klar wird und ein technisches Bild entsteht. Dafür ist ein sehr enger, sehr tiefer Austausch im Team, vor allem aber mit dem Kunden, absolut kritisch. Je besser man die Anforderung versteht, desto besser läuft der Prozess. Was tut der Kunde mit dem Produkt? Welchen Anwendungsfall hat er? Was ist seine Problemstellung? Wie fühlt er sich dabei? Es ist letztlich Storytelling. Und je besser die erzählerische Qualität des Inputs ist, desto höher sind Effizienz und Effektivität in der Softwareentwicklung.

DIALOG: Sie hatten die Herausforderung erwähnt, dynamische und verteilte Softwareprodukte überhaupt als Ganzes betrachten zu können. Wie hoch ist vor diesem Hintergrund die Chance auf einen so guten Input, dass effektiv und effizient entwickelt werden kann?

VS: Das ist nun mal die Komplexität, mit der man umgehen muss. Es geht darum, die große Story, die Eckpunkte, gut zu verstehen und die Menschen zusammenzubringen, die man vermutlich für das Projekt braucht. Das Zielprodukt wird sich im Prozess konkretisieren, ein wenig wie bei einem Buch, an dem mehrere Autoren gemeinsam schreiben. Dabei muss man die Qualität und Effizienz des Prozesses unterstützen, durch Kommunikation, Qualitätssicherung einzelner Module, gezielte laufende Feedbacks. Das schaltet nicht das Risiko aus, dass das Gesamtprodukt nicht den Erwartungen entspricht. Aber dieses Risiko gehört letztlich untrennbar zur Entwicklung komplexer Software.

DIALOG: Die von Ihnen skizzierten Eigenschaften der Softwareentwicklung und der Softwareorganisation beschreiben eine sehr besondere Landschaft. Wie groß sind die Herausforderungen, diese Welt in die klassische Produktionswelt zu integrieren?

VS: Um komplexe Industrieprodukte zu realisieren, die Hardware- und Softwareelemente beinhalten, müssen Entwicklerteams disziplin- und länderübergreifend an einer gemeinsamen Vision arbeiten. Hier kommen Menschen aus diametral entgegengesetzten Welten zusammen: Dabei tun sich Softwareentwickler schwer mit Prozessen, wo es keine Spielräume gibt, die normiert, hochsicherheitsrelevant und massenfertigungstauglich sind, die man nicht agil managen kann. Klassische Ingenieure haben wiederum oft keinen Zugang zur Offenheit und Kreativität, die in der Softwareentwicklung unabdingbar sind, sowie zu dem Phänomen, dass Entwicklung und Produktion eins sind und dass man sehr kurzzyklisch immer wieder neue Elemente aus bestehenden Modulen bauen und in Plattformen integrieren und laufend verbessern kann, während sie schon im Einsatz sind. Zwischen diesen Welten muss man Brücken bauen können. Teilweise muss man die Softwareorganisation vom klassischen Betrieb auch entkoppeln und Schnittstellen definieren, wenn die Unterschiede zu groß sind. Wie schwierig diese Aufgabe ist, sehen wir täglich.

DIALOG: Diese Brücken zu bauen wird sicher nicht dadurch einfacher, dass die Welt zunehmend „software-defined“ ist und somit die organisatorische Position der Softwareentwicklung immer stärker wird, oder?

VS: „Software-defined“ bedeutet nicht, dass es nur noch um Software geht – auch in Zukunft nicht. Aber man muss dennoch mit der Tatsache umgehen können, dass die mechanische Optimierung und der daraus resultierende Mehrwert in sehr vielen Bereichen an ihre Grenzen kommen. Wenn man bei einer Maschine bei 99,9% physikalisch möglicher Präzision ist, geht es nicht mehr darum, das noch zu verbessern. Dann wird die Möglichkeit, diese Maschine mit anderen zu vernetzen, oder die darauf laufenden Programme innerhalb von Sekunden zu ändern, differenzierend und innovationstreibend. Dann entstehen neue Nutzungsszenarien, neue Märkte. Diesen Paradigmenwechsel kann man nicht ignorieren.

„IN ÖKOSYSTEMEN ZU DENKEN WIRD IMMER WICHTIGER.“

Ein Gespräch mit Dr. Elmar Hubner, Geschäftsführer der ROI Management Consulting GmbH, Wien, über Software Excellence, Projekterfahrungen und Lessons Learned.

DIALOG: Herr Dr. Hubner, was sind die wichtigsten Fragen, die man sich als Industrieunternehmen stellen sollte, wenn man sich erstmals mit dem Thema Softwareentwicklung auseinandersetzt?

EH: Tatsächlich haben viele Industrieunternehmen heute bereits Erfahrung mit der Entwicklung von Software, etwa bei der Steuerung von elektronischen Geräten. Neu ist allerdings die wachsende Anzahl smarter Produkte, die untereinander vernetzt sind, ans Internet angebunden werden und Daten austauschen – sowie die dahinterliegenden Geschäftsmodelle, mit denen sich viele Firmen bisher nicht auseinandersetzen mussten. Zunehmend ist Hardware nicht mehr der bestimmende Faktor. Der technische Lifecycle eines Geräts ändert sich, mit Auswirkungen auf das Engineering und die Organisation. Hier betreten aktuell viele Unternehmen Neuland. Sie müssen das Thema Software neu denken und erstmals eine Softwarestrategie für ihr Unternehmen entwickeln.

Eine zentrale Frage lautet dabei: Wo will ich hin und welche Kompetenzen brauche ich dafür? Was kann inhouse entwickelt werden? Was sollte man extern vergeben? Bei diesen Make-or-Buy-Entscheidungen geht es einerseits um Kosten und Produktivität, denn Entwicklungsressourcen, speziell im Softwarebereich, sind begrenzt und hart umkämpft. Andererseits ist zu überlegen, was im Hinblick auf eine nachhaltige Softwarearchitektur und Sicherheitsaspekte im Haus verbleiben muss und was zugekauft werden kann. Grundsätzlich ist es wichtig, sich bei der Entwicklung auf wettbewerbskritische Kompetenzen zu konzentrieren und – dort, wo es möglich ist – auf vorhandene Lösungen zurückzugreifen. Man muss z.B. kein eigenes Bezahlsystem oder ein User Access Management entwickeln, dafür gibt es passende Standardmodule. Doch auch diese Komponenten können Fehler haben und darauf muss man sich vorbereiten.

Die zweite große Frage ist, wie man die Entwicklung angeht. Hier geht es um Effizienz und die passenden Methoden der Softwareentwicklung. Eine große Rolle spielen dabei die Schnittstellen bzw. Integrationen von Soft- und Hardware-Artefakten innerhalb der Organisation. Dort liegen typische Problemquellen, da beide Entwicklungsströme i.d.R. einer anderen Logik folgen – doch das könnte sich künftig auch ändern.

DIALOG: Was sind die ersten Schritte auf dem Weg von einer reinen Hardware- zu einer Software-getriebenen Company?

EH: Wenn man die Klarheit über die Softwarestrategie hat, muss man den Reifegrad des Unternehmens in Bezug auf die Softwareentwicklungsmethoden und die Organisation kennen. Viele Unternehmen merken erst im Laufe der Zeit, dass es bei Softwareentwicklung um viel mehr geht als die reine Applikation. In unseren Projekten stellen wir häufig fest, dass wichtige technische Kompetenzen gar nicht vorhanden sind, z.B. hinsichtlich Cloud-Technologien, Cyber Security oder der Funktionsweise von „Software as a Service“. Zudem sollte man sich bewusst machen, dass eine Software kontinuierliche Wartung benötigt und potenziell weiterentwickelt wird. Produkte erhalten regelmäßig neue Funktionalitäten, sie müssen auf neuen Plattformen laufen, benötigen neue Kompatibilität. Dieser Prozess ist nie abgeschlossen. Man benötigt eine Produktstrategie und ein Produktmanagement. Zudem braucht man ein Release Management und entsprechend qualifizierte Menschen, u.a. für Wartung, Dokumentation und Software/IT Operations. Man muss also die gesamte Organisation richtig aufstellen, um Software schnell entwickeln und effizient betreiben zu können. Die große Herausforderung liegt dabei oft weniger in den technischen Fragen, als vielmehr darin, aus der heutigen Organisation in die neue Struktur zu kommen. Da geht es dann um einen Transformationsprozess.

DIALOG: Wie kann denn die Organisation dazu beitragen, dass Software-Teams sinnvoll und produktiv arbeiten können?

EH: Die Organisation sollte der Entwicklung modularer, skalierbarer Softwarestrukturen dienen. Es ist entscheidend, ein ordentliches Produktmanagement auch für die Software zu etablieren. Dies beinhaltet die Anforderungen an die Software und die Architektur des Systems. Damit man ein System warten, updaten und externe Ressourcen einsetzen kann, muss man modular denken. Mitunter findet man in Unternehmen über Jahre gewachsene Software-Monolithen vor, die weder aktualisierbar noch skalierbar sind, weil alle Funktionalitäten zusammenhängen. Da kann es vorkommen, dass Hunderte Mitarbeiter nonstop an der Wartung der Software arbeiten, aber keinen echten Output generieren, weil sie permanent Altlasten – die sogenannte Technical Debt – bereinigen. Wenn man einen Punkt überschreitet, kommt man da kaum noch heraus. Solche ineffizienten Strukturen muss man erkennen, rechtzeitig auseinandernehmen und durch modulare Strukturen

ersetzen. Modulare Architekturen sind auch die Voraussetzung für die Skalierbarkeit einer Organisation. Genauso wird dadurch die Einbindung externer Lösungen und damit die Nutzung von weiteren Innovationsquellen möglich. Doch eine nachhaltige Architektur erfordert eine Menge Vorarbeit, geistige Leistung und kontinuierliche Pflege.

DIALOG: Immer wieder kommt es vor, dass Software beim Release nicht wie geplant funktioniert oder dass Systeme nach einem Update ausfallen. Wo liegen die Gründe dafür?

EH: Dass eine Software beim Launch versagt, kann ganz banale Gründe haben. Vielleicht wurden in der Planung die späteren Zugriffszahlen unterschätzt. Möglicherweise wurde nicht hinreichend getestet. Man darf auch die Variabilität und Komplexität der Produkte im Feld nicht unterschätzen. Zudem läuft Software nicht nur mit Code, sondern mit vielen weiteren Daten, die mitunter nicht berücksichtigt werden. Das Testen von Software ist in der Tat eine große Herausforderung. Automobilhersteller z.B. verlassen sich niemals nur auf reine Softwaretests, sondern prüfen die Software vor einem Release immer im Kontext des realen Fahrzeugs. Aus den unzähligen möglichen Fahr- und Bediensituationen, die sich unter realen Bedingungen ergeben, entstehen zu viele Fehlerquellen. Da kann man sich nicht nur auf eine Simulation verlassen. Zur Qualitätssicherung setzen manche Unternehmen, z.B. in regulierten Branchen, auf das Prinzip des Pair Programming bzw. Pair Fixing, bei dem zwei Entwickler nach dem Vier-Augen Prinzip gemeinsam an einem Code arbeiten. Es gibt jedenfalls Wege, die Risiken zu minimieren.

DIALOG: Welche Synergien oder Skaleneffekte lassen sich denn in der Softwareentwicklung nutzen?

EH: In vielen Fällen können Entwicklerteams auf umfangreiche Funktionsbibliotheken zurückgreifen. Und zunehmend wird der Entwicklungsprozess durch Automatisierung unterstützt. So lässt sich Code in den verschiedenen Phasen des Software-Delivery-Prozesses mittlerweile automatisiert auf formale Fehler oder mögliche Sicherheitslücken überprüfen.

DIALOG: Hardware- und Softwareentwicklung folgen teils unterschiedlichen Prozessen und Methoden. Wenn ein Produkt wie z.B. ein Fahrzeug zu einem bestimmten Zeitpunkt serienreif sein und in die Produktion gehen soll, müssen alle auf diesen Punkt hinarbeiten. Wie bekommt man das harmonisiert?

EH: Das funktioniert über Synchronisationspunkte, die im Entwicklungsprozess in regelmäßigen Abständen positioniert werden. Bei großen Projekten können das im Zeitablauf mehrere Hundert sein. Um bei dem Beispiel zu bleiben: Wenn Sie einen Motor zusammen mit der Abgasnachbehandlung testen wollen und deren Softwareversion nicht mit der des Motors kompatibel ist, kann es sein, dass der Motor beim Teststart nicht anspringt. Insbesondere die Synchronisation der Hardware- und Softwareentwicklung zu managen ist eine hohe Kunst und eine große Herausforderung für die Planung. Wenn die Software- und Hardware-Teams jeweils auf ihrer Seite verharren, funktioniert es nicht. Beide Seiten müssen sich jeweils in die andere Welt hineindenken. Gelingt die Integration, lässt sich großes Potenzial heben.

DIALOG: Brauchen wir also zunehmend Menschen, die mit der Software- und Hardware-Welt vertraut sind und beide Bereiche wirklich verstehen?

EH: Ja, absolut. Ein integriertes Software Engineering und eine skalierbare Systemarchitektur aufzubauen sind sehr anspruchsvolle Aufgaben. Um übergreifende Systemkompetenz zu entwickeln, braucht man nicht nur die entsprechenden Softwarekenntnisse, sondern auch das individuelle Domänen-Know-how der unternehmensspezifischen Prozesse und Produkte. Die Leute, die sich in beiden Welten bewegen können, kann man nicht einfach schnell einkaufen.

DIALOG: Verschmelzen Hardware und Software zunehmend oder verschiebt sich eher die Rangordnung zwischen den beiden Bereichen?

EH: Das hängt vom Unternehmen und vom Produkt ab – und davon, über welche Merkmale sich das Unternehmen im Markt differenzieren möchte. Am Ende entscheidet darüber die Sicht des Kunden bzw. des Nutzers. Bei manchen Produkten, etwa bei bestimmten Automodellen, standen die mechanische Exzellenz und Aspekte wie das Fahrverhalten im Vordergrund.

Zukünftig werden jedoch softwarebasierte Funktionalitäten, wie z.B. Assistenzsysteme, den Kundennutzen bestimmen. Bei Produkten wie etwa Smartphones leitet sich der Kundennutzen mittlerweile nahezu vollständig aus der Software ab. Klar ist, dass die Bedeutung von Software bei allen technischen Produkten weiter zunehmen wird.

Ein Beispiel ist die sogenannte Funktionsintegration: Wo man früher für zehn Funktionen zehn verschiedene mechanische Knöpfe hatte, kann man heute mit einer Softwareapplikation über einen Touchscreen alle Funktionen bedienen. Damit steigt die Bedeutung der Software, aber auch ihre Komplexität.

DIALOG: Wenn wir den größeren Kontext betrachten, welche Herausforderungen ergeben sich durch Entwicklungen wie das Internet of Things an die wachsenden Software-Ökosysteme?

EH: Die Interoperabilität zwischen Plattformen und in diesem Zusammenhang auch die Fähigkeit, Produkte in Ökosysteme einzubinden, sind entscheidende Punkte. Überhaupt in Ökosystemen zu denken wird immer wichtiger. Wenn Sie heute einen Fernseher kaufen, können Sie diesen selbstverständlich über Android und iOS ansteuern. Vor der gleichen Herausforderung stehen Industrieunternehmen. Es geht dabei nicht nur um die Softwareentwicklung, mitunter kommen von der Softwareseite ganz neue Wettbewerber ins Spiel. Auch hier sind die entscheidenden Fragen: Welche Kompetenzen brauche ich?

Will ich diese intern verankern oder am Markt einkaufen? Diese Ressourcen zu entwickeln, die entsprechende Architektur aufzubauen und die Schnittstellen zu managen, das sind Schlüssel zu zukünftiger Exzellenz in einer immer stärker Software-geprägten Welt.

DIALOG: Kann es unter Qualitäts- und Effizienzgesichtspunkten auch sinnvoll sein, Softwareentwicklung ganz abzugeben? Könnte z.B. ein Unternehmen, dessen Kernkompetenz im mechanischen Bereich liegt, Software nicht komplett extern einkaufen?

EH: Auch die mechanischen Produkte funktionieren heute nicht mehr ohne Software und die Fertigungsprozesse werden immer stärker digitalisiert. Kunden wollen Produkte, die Probleme lösen und funktionieren.

In diesem Zusammenhang spielt Software eine immer größere Rolle. In Branchen, in denen es auf der Hardwareseite wenig Innovation gibt, kann sogar das Umschwenken auf reine Software eine Option sein. Die Software und die Frage, wie man ihre Entwicklung managt, werden zunehmend zum Domänenwissen, zum Differenzierungsmerkmal und zum Wettbewerbsvorteil. Darum ist es wichtig, bei der Entwicklung die Zügel in der Hand zu behalten. Wenn man die Architektur skalierbar und modular aufgebaut hat, kann man problemlos Module extern vergeben oder zukaufen. Aber die Kunden und die Markterkenntnis, die Softwarearchitektur, die Beschreibung der Funktionen und die Spezifikationen, die muss das Unternehmen selbst im Griff haben.

BUILDING INDUSTRIAL FUTURE TOGETHER

ROI-EFESO Management Consulting zählt zu den führenden internationalen Operations-Beratungen. Wir verbinden Fach-, Digital- und Transformationskompetenz, um innovative End-2-End-Konzepte für die Gestaltung von Supply Chain und Operations zu entwickeln und weltweit zu realisieren. Dabei setzen wir insbesondere auf die Potenziale datengetriebener Ansätze, um Komplexität zu beherrschen, Effizienz zu steigern und neue Betriebs- und Geschäftsmodelle zu verwirklichen.

Mit unserer Arbeit helfen wir der produzierenden Industrie, ihren Beitrag zu einer lebenswerten und zukunftssicheren Welt zu leisten. Dabei behalten wir ambitionierte Nachhaltigkeitsziele im Blick und unterstützen Menschen, die diesen Wandel vom Boardroom bis zum Shopfloor bewältigen müssen.

Als Teil der international agierenden EFESO-Gruppe verfügt ROI-EFESO über eine starke Präsenz in den wichtigsten Industrieregionen der Welt. Die Arbeit von ROI-EFESO erhält für ihre Ergebnisqualität, Effizienz und den Innovationsgrad regelmäßig renommierte Auszeichnungen. Seit 2013 zeichnet ROI-EFESO die besten Praxislösungen im Kontext der industriellen Digitalisierung mit dem internationalen „Industrie 4.0 Award“ aus, der zu den wichtigsten Benchmarks für die digitale Transformation in der Industrie zählt.

www.roi-efeso.com