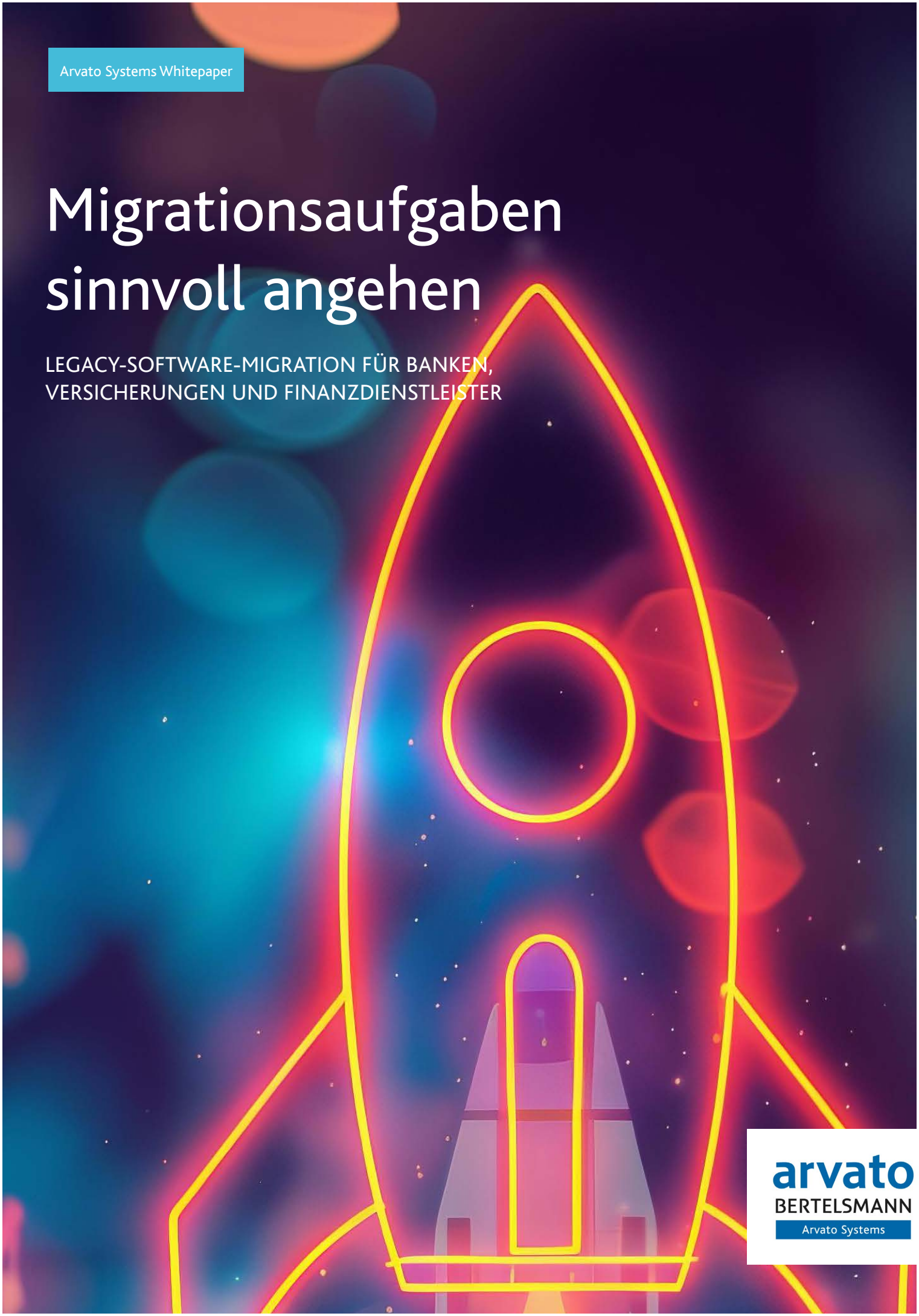
Arvato Systems Whitepaper

Migrationsaufgaben sinnvoll angehen

LEGACY-SOFTWARE-MIGRATION FÜR BANKEN, VERSICHERUNGEN UND FINANZDIENSTLEISTER



arvato
BERTELSMANN
Arvato Systems

Einleitung

Früher war es für Banken und Versicherungen einmal Branchenstandard, ihre COBOL-Applikationen auf IBM-Mainframes zu betreiben. Und heute? Heute sind die veralteten, unflexiblen und höchst wartungsaufwendigen Legacy-Systeme der wohl größte technische Bremsklotz für die Weiterentwicklung von Finanzinstituten – und ein ernstes Problem im Wettbewerb. Aufgeschobene Migrationsprojekte kosten letztlich Marktanteile, Entwicklungschancen und Zukunftsfähigkeit. Den Verantwortlichen in von ihren Altsystemen geplagten Banken und Versicherungen ist dies durchaus bewusst. So hat beispielsweise Expleo im Rahmen seines „Global Business Transformation Index 2023" 200 Führungskräfte und Experten aus der Finanzindustrie danach gefragt, welche Faktoren den größten Veränderungsdruck auf ihr Unternehmen ausübten. Nach neuen Kundenerwartungen (48 Prozent) und Kostenreduzierung (46 Prozent) wurden Legacy-Systeme und veraltete Technologie mit 42 Prozent bereits am dritthäufigsten genannt (Quelle: https://expleo.com/global/de/zukunft-des-bankings/#digital-transformation-initiatives). Zudem darf man davon ausgehen, dass sich mit unflexiblen und wartungsintensiven Altsystemen weder moderne Kundenwünsche erfüllen noch Kosten reduzieren lassen.

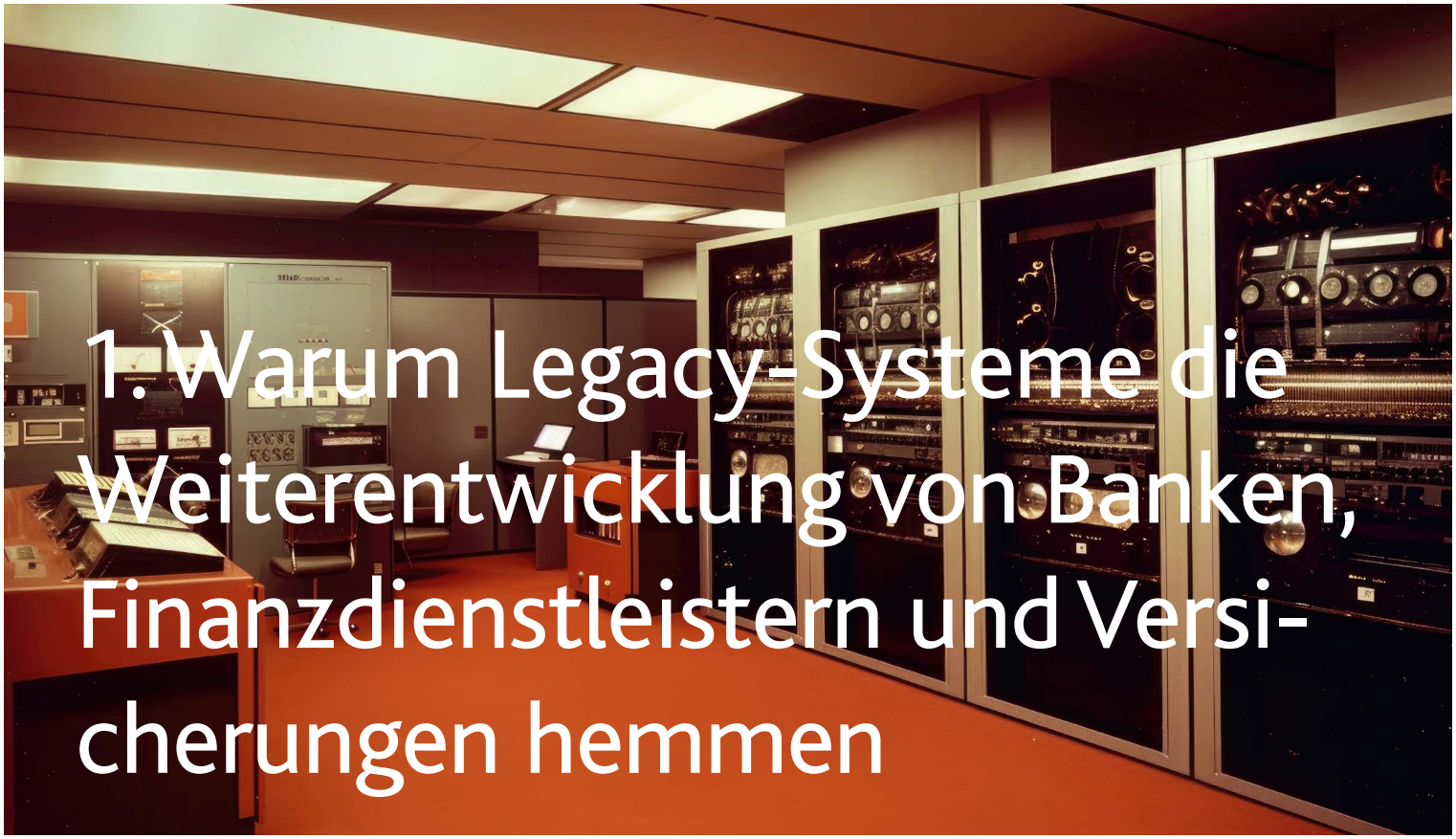
Es ist unverzichtbar, sich endlich mit dem Modernisierungsstau in der Finanz- und Versicherungsbranche auseinanderzusetzen und die Migration der veralteten Legacy-Software in Angriff zu nehmen – auf methodisch sinnvolle und durchdachte Weise.

Dieses Whitepaper möchte Ihnen zeigen

- welche Herausforderungen Unternehmen behindern, wenn sie an Legacy-Systemen festhalten,
- welche strategischen Fragestellungen Sie vor Ihrem Migrationsprojekt beantworten sollten und
- wie solch eine Migration alter Softwarekomponenten beispielhaft aussehen kann.

Um eines schon vorwegzunehmen: Ein Migrationsansatz, der auf Code-Automation basiert, bringt erfahrungsgemäß oft eher fragwürdige Ergebnisse hervor. Sehr viel zielführender ist ein individueller Ansatz, der zunächst Ihren Bedarf analysiert und Ihre individuellen Anforderungen, Funktionen und Prozesse einbezieht. Nur so haben Sie die Gewissheit, dass die Applikation Ihrem fachlichen Bedarf auch dann noch gerecht wird, wenn sie die Welt des COBOL-Codes verlassen hat. Schließlich geht es Ihnen bei Ihrem Migrationsprojekt darum, Zukunftsfähigkeit und -sicherheit herzustellen.

Unser Whitepaper gibt Ihnen einen ersten Einblick, welche Fragen bei der Ablösung Ihrer Legacy-Systeme relevant sein können. Wenn Sie für Ihr Migrationsvorhaben unser Know-how persönlich in Anspruch nehmen möchten, sprechen Sie uns bitte einfach an.
Wir wünschen Ihnen eine aufschlussreiche Lektüre.
Ihr Team von Arvato Systems



1. Warum Legacy-Systeme die Weiterentwicklung von Banken, Finanzdienstleistern und Versicherungen hemmen

1.1 Aussterbende Programmiersprachen und schwindende Ressourcen

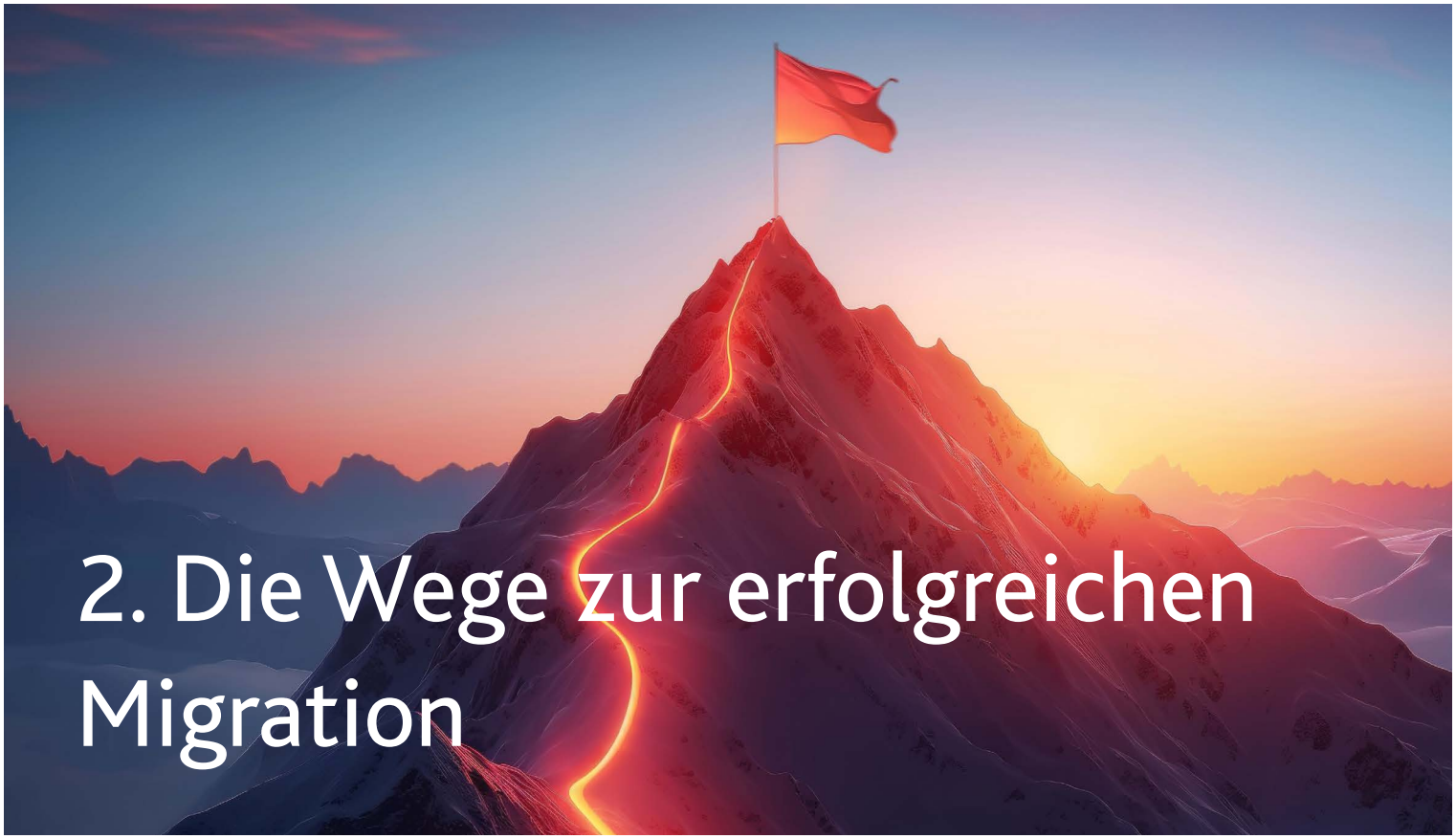
Es ist schon einfach das Alter von Legacy-Systemen, das dazu führt, dass sie Banken, Finanzdienstleister und Versicherungen vor oft gravierende Herausforderungen stellen. Die erste Folge des hohen Alters der Systeme: Es gibt in den Finanzinstituten immer weniger Spezialisten, die sich noch mit ihnen auskennen. Früher war es in der Branche praktisch der Standard, in COBOL programmierte Applikationen auf IBM-Mainframe-Rechnern zu betreiben. Schon weil viele Programmierer inzwischen das Rentenalter erreicht haben, stehen heute aber immer weniger Experten zur Verfügung, die sich mit den Programmiersprachen, die Anfang der neunziger Jahre benutzt wurden, noch auskennen. Man darf nicht vergessen: Viele der damals gängigen Programmiersprachen sind sogar noch älter und stammen aus den siebziger Jahren.

1.2 Teure Wartung und teure Lizenzen – ohne Flexibilität

Das Problem der aussterbenden Programmiersprachen in den Legacy-Systemen behindert auch etwaige Änderungswünsche an den Applikationen. Neue Business-Features umzusetzen, wird darum typischerweise schwierig oder teuer – wenn die technischen Limitierungen der veralteten Applikation es nicht sogar völlig unmöglich machen, neue Funktionen zu integrieren. Gleichzeitig wird die Wartung der Systeme immer aufwendiger. Auch dies ist angesichts des Ressourcenmangels eine Herausforderung. Ebenso ist der Betrieb inzwischen mit hohen Kosten verbunden. Dabei ist es schon aus Perspektive der Lizenzkosten ein Nachteil, bei seinem Legacy-System von einem monopolistischen Anbieter abhängig zu sein. Die veralteten Applikationen weiterhin zu nutzen, bedeutet in nahezu jedem Fall, auf die Flexibilität zu verzichten, die eine neue, modernere Lösung eröffnen würde. Das Problem ist auch noch nicht behoben, wenn beispielsweise die COBOL-Applikation bereits von dem alten Mainframe in eine Unix-Umgebung umgezogen ist. So hat man zwar die Lizenzkosten für den Mainframe gespart, aber der Mangel an Flexibilität besteht nach wie vor.

1.3 Betriebsrisiken sowie Compliance- und Audit-Probleme

Nicht nur die Wartung wird immer komplizierter – Legacy-Systeme haben schon wegen ihres hohen Alters ein inhärent höheres Ausfallrisiko. Das macht es noch wichtiger, nach unerwarteten Unterbrechungen schnell den Normalbetrieb wiederherstellen zu können. So spielt auch bei Audits der IT-Infrastruktur eines Finanzinstituts die Betriebskontinuität eine große Rolle. Altsysteme können vor diesem Hintergrund besonders problematisch sein, auch wegen des Spezialistenmangels. Zudem verändern und verschärfen sich Compliance-Anforderungen kontinuierlich. Auch hier kann das Festhalten an Legacy-Systemen dazu führen, dass es immer komplizierter wird, den einschlägigen regulatorischen Anforderungen und Berichts- und Meldepflichten in vollem Umfang zu genügen. Oft machen Altsysteme es unnötig schwierig, alle benötigten Daten für regelmäßige Berichte, Analysen oder interne wie externe Audits zu extrahieren.



2. Die Wege zur erfolgreichen Migration

2.1 Strategische Klarheit schaffen: Wie sieht das Zielbild aus?

Bevor es an die Ablösung und Migration von Altsystemen gehen kann, sollten Banken, Finanzinstitute und Versicherer sehr gut durchdenken, welchen Zielzustand sie erreichen möchten. Wie soll das konkrete Ergebnis aussehen? Welche Funktionen und Prozesse muss das System auch nach seiner Migration noch abbilden? Soll es vielleicht sogar völlig neue Funktionen unterstützen? Letztlich ist die Frage also: Welche strategische Relevanz hat die zu migrierende Applikation für das zukünftige Business? Die konkreten Ansprüche hängen natürlich auch davon ab, zu welcher Klasse von Lösungen das Altsystem gehört. Ist es beispielsweise

- ein Abrechnungssystem
- ein Buchhaltungssystem
- eine Buchungsdatenverwaltung
- eine Kontodatenverwaltung
- ein Kundenmanagement oder
- ein Zahlungssystem?

Hier ist ebenfalls zu klären, zu welcher Zielbetriebsumgebung das Altsystem migriert werden soll. Gegebenenfalls existieren sogar bereits klare Vorgaben zur Zielumgebung, weil das bestehende Rechenzentrum der Bank, der Versicherung oder des Zahlungsdienstleisters dies so vorgibt. Generell zählen zu den Optionen für die Betriebsformen beispielsweise

- On-Premises (klassischer Betrieb auf einem Server im eigenen Rechenzentrum oder in dem eines Hosters)
- Virtual Private Cloud (VPC – als isolierter, eigener Bereich in einer mandantenfähigen Public Cloud für geschäftskritische Applikationen)
- Public Cloud ohne cloud-native Applikationen (für eher geschäftsunkritische, über das öffentliche Internet bereitgestellte Applikationen)
- Public Cloud mit modernen cloud-nativen Applikationen (um die volle Skalierbarkeit, Flexibilität, Elastizität und Resilienz der Cloud zu nutzen).

2.2 Drei grundsätzliche Ansätze für eine Migration

Prinzipiell stehen einer Bank, einem Finanzdienstleister oder einem Versicherer für die Ablösung eines Legacy-Systems drei Migrationsansätze offen:

- die Einführung einer neuen Standardsoftware
- die Übersetzung des alten Programmiercodes in eine moderneren wie etwa Java
- die Entwicklung einer neuen, auf den Bedarf zugeschnittenen Individualsoftware.

Weg 1: Die erprobte Standardsoftware

Diese drei Wege unterscheiden sich sowohl nach dem Grad, zu dem ein Unternehmen auch nach der Migration noch seine individuellen Anforderungen und Prozesse abbildet, und zum anderen nach dem mit der Migration verbundenen Projektrisiko. So ist die Gefahr, dass das Projekt „Einführung einer neuen Standardsoftware“ aus dem Ruder läuft, vergleichsweise gering. Die Kehrseite: Die Standardsoftware macht es für die Bank, den Finanzdienstleister oder die Versicherung erforderlich, die eigenen, etablierten Prozesse weitgehend an die im Standard verfügbaren Funktionalitäten anzupassen.

Weg 2: Die passgenaue Neuentwicklung

Oft sind Legacy-Systeme aber im Laufe der Zeit gewachsen, haben individuelle Funktionen und Prozesse bekommen, die dem Bedarf der Bank, des Zahlungsdienstleisters oder der Versicherung sehr genau entsprechen. All diese Individualität in einer Standardsoftware abbilden zu wollen, ist meist unmöglich. Gegenüber der Standardsoftware markiert eine neuentwickelte Individualsoftware das andere Extrem. Hier können die Entwickler dafür sorgen, dass die neue Applikation dem Bedarf optimal

entspricht und zugleich flexibel und offen genug ist, auch neue und zukünftige Anforderungen zu erfüllen. So kann es sinnvoll sein, sich bei der Neuentwicklung der Applikation gleich für eine cloud-native Software zu entscheiden. Die Kehrseite dieses Migrationswegs: Es existiert ein gewisses Risiko, dass das Projekt einer kompletten Neuentwicklung in beträchtliche Schieflage gerät oder komplett scheitert. Beispielsweise wird mitunter die Komplexität einer Neuentwicklung unterschätzt, den Entwicklern fehlt es gegebenenfalls an dem erforderlichen fachlichen Know-how, oder man geht das Projekt der Neuentwicklung mit einer falschen Priorisierung der umzusetzenden Funktionen an. Wegen des inhärent höheren Risikos einer völligen Neuentwicklung ist es unverzichtbar, sich genau zu überlegen, welche strategische Relevanz das neue System hat. Das heißt: wie genau es dem aktuellen und zukünftigen Bedarf des Unternehmens entsprechen muss. Denn wenn die Applikation nach der Migration viele neue Business Features haben soll, lässt sich die Individualentwicklung gar nicht umgehen. Die strategische Notwendigkeit kann ein guter Grund dafür sein, das Mehr an Aufwand und Risiko in Kauf zu nehmen. Angesichts aktueller und zukünftiger Herausforderungen ist die Migration per Neuentwicklung mitunter die einzig sinnvolle Option.

Weg 3: Alten Code in moderne Programmiersprachen übersetzen

Zwischen diesen beiden Extremen – Standardsoftware versus Neuentwicklung – existiert auch ein beliebter Mittelweg: Die Übersetzung des alten Codes in eine neue, modernere und flexiblere Programmiersprache wie etwa Java oder .NET. Grundsätzlich kann es so vergleichsweise leicht gelingen, die speziellen, individuell gewachsenen Funktionen und Prozesse aus der COBOL-Welt in das Java-Universum zu übertragen. Die Legacy-Applikation wird in der neuen Programmiersprache nachgebaut und dann per Lift and Shift in die Public Cloud gebracht. Ebenso ist es denkbar, nur das Frontend der übersetzten Applikation in der Public Cloud bereitzustellen, das Backend dagegen in eine Virtual Private Cloud zu migrieren. Aber Vorsicht: Was sich so gut wie nie empfiehlt, ist, für die Übertragung in die neue Programmiersprache Code-Automation-Lösungen nutzen zu wollen. Im folgenden Abschnitt 2.3 beleuchten wir die Gründe dafür.

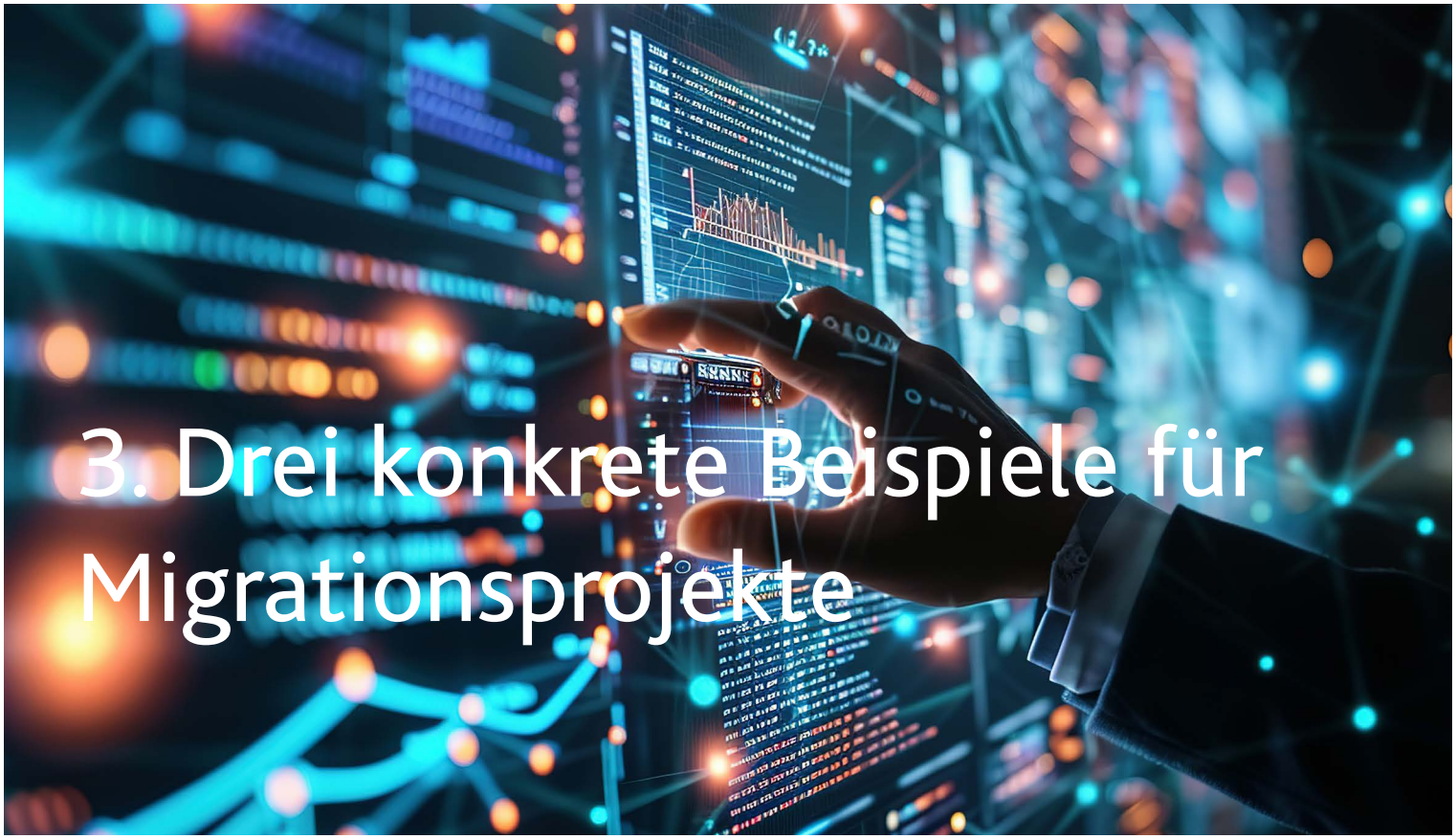
2.3 Softwareentwicklung schlägt Code-Automation

Nie vergessen sollte man bei einer Migration, dass sie nicht nur eine technische, sondern auch eine hochrelevante fachliche Dimension hat. Gerade im Kontext des Migrations-Mittelwegs der Code-Übersetzung besteht eine gewisse Gefahr, dies aus dem Blick zu verlieren. Der Grund: die Verlockung der Code-Automation-Lösungen. Code-Automation verspricht, den Code der Legacy-Applikation komplett und vollautomatisch in die moderne Programmiersprache zu übertragen. Nur ist genau dies das Problem einer maschinellen Übersetzung: Sie findet vollautomatisch statt. Der Code-Automat kann nicht abwägen, welche Code-Bestandteile noch sinnvoll sind und welche nicht, darum überträgt er stets den kompletten Programmcode. Aber der Code alter Legacy-Systeme ist typischerweise komplex. Diese Komplexität ist der Verwendung alter Programmiersprachen wie COBOL geschuldet und war damals unvermeidbar. Per Code-Automation überträgt man diese

unnötige Komplexität aber eins zu eins in die neue Programmiersprache. Eine negative Folge ist, dass sich die Wartung der neuen, auf Java basierenden Applikation dann ähnlich schwierig gestaltet wie die der alten. Der andere negative Aspekt: Die Anpassbarkeit der neuen Applikation verschlechtert sich dramatisch.

2.4 Erfolg dank technischem und fachlichem Know-how

Eine automatisierte Übertragung des Legacy-Codes mag kurzfristig günstiger erscheinen, aber auf mittlere Sicht ist damit wenig gewonnen. Es ist wichtig, dies zu verstehen: Auch wenn man sich für den Mittelweg bei Individualisierung und Risiko entscheidet – für die Migration per Code-Übertragung –, ist ein gewisses Maß an Softwareentwicklungs-Know-how dennoch unentbehrlich. Damit z.B. in Java auf Basis des alten Legacy-Systems eine sinnvolle und zukunftsfähige Applikation entstehen kann, braucht es das Wissen und die Eingriffe menschlicher Experten. Java-Entwickler werden schnell erkennen, wo es möglich und sinnvoll ist, die alte COBOL-Komplexität zu reduzieren. Zudem profitiert auch eine Migration per Code-Übertragung von fachlichem Know-how, von der fachlichen Perspektive der Programmierer auf die individuellen Anforderungen und Bedürfnisse des Unternehmens. Mit dem Status quo der Legacy-Anwendung auch deren Probleme und deren mangelnde Flexibilität gegenüber neuen geschäftlichen Anforderungen fortzuschreiben, würde das Ziel der Migration ad absurdum führen.



3. Drei konkrete Beispiele für Migrationsprojekte

3.1 Zahlungsverkehrssystem: vom Legacy-System zur Standardsoftware

Auf eine Standardsoftware zu migrieren, bedeutet immer, dass die Bank, der Zahlungsdienstleister oder die Versicherung sich von gewissen, vielleicht auch historisch gewachsenen Funktionen in der eigenen Legacy-Applikation verabschiedet. Eine Standardlösung macht es darum erforderlich, auch die eigenen etablierten Prozesse bis zu einem gewissen Grad an die Möglichkeiten anzupassen, die die neue Software nun einmal zur Verfügung stellt. Allerdings muss dies für die Organisation nicht sonderlich schmerzlich sein. Eine Standardsoftware, die ihren Namen verdient, ist vermutlich ausgereift und leistungsfähig genug, die überwiegende Zahl der gängigen Anwendungsszenarien abzubilden. Im Kontext der Banking- und Financial Services-Branche zählen beispielsweise Zahlungsverkehrssysteme zu dieser Softwarekategorie mit brauchbaren Standardlösungen, die sich in der Branche auch schon recht weit verbreitet haben. Dennoch gilt es natürlich, sich die Funktionalitäten, die die Lösungen der diversen Anbieter offerieren, genau anzusehen – der Zahlungsverkehr einer Bank ist oft alles andere als trivial. Sollte die Möglichkeit, die eigenen Prozesse der Standardapplikation anzupassen, nicht bestehen, kann es lohnen, den Aufwand individueller Anpassungen an der neuen Standardsoftware in Kauf zu nehmen.

3.2 Meldewesen: durch Migration vom Alt-System zur individuellen Java-Lösung

Manche Applikationen sind so komplex und mitunter auch hochindividuell, dass eine reine Migration in Gestalt einer Code-Übertragung den Königsweg dafür darstellt, sich von seinem Legacy-System zu verabschieden. Den alten COBOL-Code von kompetenten Softwareentwicklern in Java, .NET oder anderen Programmiersprachen nachbauen zu lassen, vermeidet den Aufwand und das Risiko einer völligen Neuentwicklung, eröffnet aber

dennoch die Möglichkeit, den neuen Code schlanker, wartbarer und anpassbarer zu gestalten, als dies mit einem bloßen Code-Automation-Ansatz möglich wäre. So hat etwa die Bank, der Finanzdienstleister oder die Versicherung die Gewissheit, dass alle relevanten individuellen Funktionalitäten auch in der neuen, auf der modernen Programmiersprache basierenden Applikation nach wie vor enthalten sind. Ein typisches Beispiel für diesen Migrationsfall per Code-Übertragung ist das Meldewesen einer Bank. Bekanntlich bestehen umfassende Anzeige- und Meldepflichten gegenüber der Bundesanstalt für Finanzdienstleistungsaufsicht (BaFin), der Deutschen Bundesbank sowie der Europäischen Zentralbank – ob es etwa um Millionenkredite, die eigene Liquidität oder Wertpapiertransaktionen geht. Hier ist es entscheidend, dass das Meldewesen auch nach der Migration noch mit derselben Zuverlässigkeit funktioniert wie zuvor. Eine Code-Übertragung durch bspw. Java-Spezialisten, die auch COBOL verstehen, wird dies leisten.

3.3 B2C-E-Commerce-Zahlungssystem: die völlig neue Individualsoftware

Altsysteme sind oft weit davon entfernt, Daten in Echtzeit verarbeiten zu können. Sie bedienen sich stattdessen eines Batch-Ansatzes. Das heißt, sie verarbeiten zu regelmäßigen Zeiten eine größere Zahl von Vorgängen auf einmal, in einem größeren Batch. In vielen Anwendungsfällen ist dies heute aber nicht mehr zeitgemäß. Zahlungsdienstleister etwa benötigen oft eine Realtime-Verarbeitung, mit einem System, das 24/7 verfügbar ist. Will ein Zahlungsdienstleister beispielsweise für eine B2C-Handelsplattform ein E-Commerce-Zahlungssystem anbieten, wäre das angesichts der Ansprüche an eine rund um die Uhr verfügbare Realtime-Verarbeitung von keinem Legacy-System zu leisten. Gleichzeitig muss ein B2C-Zahlungssystem problemlos skalierbar sein und auch jederzeit Lastspitzen bewältigen können. Hier ist eine Migration per neuer Individualentwicklung oft der einzige Weg, das Geschäftsmodell des Unternehmens auch in Zukunft zu unterstützen und die strategischen Ziele umzusetzen. Das grundsätzlich höhere Risiko einer völligen Neuentwicklung wird vor diesem Hintergrund nachrangig.

4. Zusammenfassung: Migrieren, aber richtig!

„Dieses Legacy-System muss endlich weg. Damit haben wir nichts als Ärger." Das ist ein verständlicher Wunsch. Aber eine erfolgversprechende und tragfähige Migrationsstrategie hat man damit noch nicht. Welcher Migrationsweg der richtige ist, um sich endlich von dem unflexiblen, ressourcenfressenden und risikobehafteten Altsystem verabschieden zu können, ist immer auch eine strategische Frage. Darum ist die initiale Analyse der Ist- und der angestrebten Ziel-Situation so wichtig. Welche Funktionen des Legacy-Systems sind auch in Zukunft unverzichtbar, welche zusätzlichen Funktionen sind erforderlich, und wie flexibel soll das neue System nach der Migration werden? Erst wenn diese Fragen vor dem Hintergrund der strategischen Ausrichtung der Bank, des Zahlungsdienstleisters oder der Versicherung geklärt sind, kann es daran gehen, den individuell optimalen Weg für die technische Softwaremigration zu bestimmen. Ob neue Standard-Software, reine Migration per Code-Übertragung oder völlig neue Individualentwicklung: Welche Option sinnvoll ist, hängt immer von Ihrem konkreten Bedarf ab. Denn mit Ihrer Migration verfolgen Sie ein strategisches Ziel. Und Ihr Migrationsprojekt soll die Voraussetzung schaffen, dieses Ziel tatsächlich zu erreichen.

Legacy-Systeme abzulösen, ist sehr oft eine gute Idee. Aber entscheidend ist, wie Sie dies tun. Bei Arvato Systems haben wir das Know-how und die Erfahrung, Sie in Sachen Legacy-Migration eingehend zu beraten und umfassend zu unterstützen. Nutzen Sie unsere Kompetenz.

5. Unternehmensprofil, Impressum, Kontakt

5.1 Über Arvato Systems

Als international agierender IT-Spezialist unterstützt Arvato Systems namhafte Unternehmen bei der Digitalen Transformation. Rund 3.400 Mitarbeitende an weltweit über 25 Standorten stehen für hohes technisches Verständnis, Branchen-Know-how und einen klaren Fokus auf Kundenbedürfnisse.

Als Team entwickeln wir innovative IT-Lösungen, bringen unsere Kunden in die Cloud, integrieren digitale Prozesse und übernehmen den Betrieb sowie die Betreuung von IT-Systemen. Zudem können wir im Verbund der zum Bertelsmann-Konzern gehörenden Arvato Group ganze Wertschöpfungsketten abbilden.

Durch unser starkes strategisches Partner-Netzwerk mit internationalen Top-Playern wie AWS, Google, Microsoft oder SAP stärken wir unser Know-how kontinuierlich und setzen auf modernste Technologie.

We Empower Digital Leaders.

Weitere Fragen?
Nehmen Sie Kontakt mit uns auf:

Arvato Systems GmbH
Jan-Peter Gordon
Experte für Software Entwicklung für Banken & Versicherungen
Reinhard-Mohn-Straße 18
33333 Gütersloh
Tel.: +49 5241 80 40731
Banking_Insurance@Bertelsmann.de

arvato
BERTELSMANN
Arvato Systems