OPITZ CONSULTING

Digitale Service Manufaktur

Wohin mit dem Monolithen?

Optionen der Modernisierung für eine evolutionäre Applikationswelt

Whitepaper

© OPITZ CONSULTING Deutschland GmbH 2020

Wohin mit dem Monolithen?

Optionen der Modernisierung für eine evolutionäre Applikationswelt

Über den Autor

Rolf Scheuch ist geschäftsführender Gesellschafter und Chief Technical Officer bei OPITZ CONSULTING. Daneben ist er als Managementcoach aktiv, mit Schwerpunkten in der Entwicklung einer geschäftszielorientierten IT-Strategie und der organisatorischen Implementierung von Initiativen im Umfeld der digitalen Transformation. Ferner engagiert sich Scheuch als Speaker auf Fachkonferenzen und als Autor in Fachpublikationen.

Kontakt

Rolf Scheuch
Chief Technical Officer
OPITZ CONSULTING GmbH
Kirchstr. 6
51647 Gummersbach
rolf.scheuch@opitz-consulting.com

Impressum

OPITZ CONSULTING Deutschland GmbH
Kirchstr. 6
51647 Gummersbach
+49 (0)2261 6001-0
info@opitz-consulting.com

Disclaimer

Text und Abbildungen wurden sorgfältig entworfen. Die OPITZ CONSULTING Deutschland GmbH ist für den Inhalt nicht juristisch verantwortlich und übernimmt keine Haftung für mögliche Fehler und ihre Konsequenzen. Alle Rechte, z. B. an den genannten Prozessen, Show Cases, Implementierungsbeispielen und Quellcode, liegen bei der OPITZ CONSULTING Deutschland GmbH. Alle genannten Warenzeichen sind Eigentum ihrer jeweiligen Besitzer.

Inhalt

1 Einführung 3

2 Modernisierung im Spannungsfeld 5

3 Unabhängige Produktstrategie als Ziel 7

4 Handlungsoptionen 8
4.1 Status quo als legitime Option 8
4.2 Zeit gewinnen durch Veränderung 9
Re-Hosting/Lift & Shift 9
Portierung/Migration 9
Enablement durch Kapselung/Wrapping 10
4.3 Ablösung 10
Einführung von Standardsoftware 10
Re-Engineering durch ein Re-Write 11
Monolith Crunching 12

5 Fazit 13
Kleine Checkliste zur Modernisierung von monolithischen Anwendungslandschaften 13

© OPITZ CONSULTING Deutschland GmbH 2020

Vorwort

Das Thema der Modernisierung von Legacy-Systemen bzw. monolithischen Anwendungen beschäftigt momentan die deutschen Unternehmen und IT-Abteilungen wie kein anderes, da viele dieser Altsysteme eine *strukturelle Zukunftsunfähigkeit* der Unternehmen bewirken. Diese integrierten und umfänglichen Systeme erweisen sich meist als änderungsresistent, daher besteht hier ein ernstzunehmender Handlungsbedarf. Hinzu kommt eine verspürte höhere Marktdynamik durch Globalisierung und Digitalisierung, die zu der Notwendigkeit führt, IT-Systeme als Differentiator zu verwenden.

Das vorliegende Whitepaper erläutert die grundlegende Problematik der Modernisierung von Altsystemen und gibt wertvolle Hinweise durch ein Aufzeigen der unterschiedlichen Handlungsoptionen bei der Transformation (sprich Modernisierung) dieser meist monolithischen Altsysteme.

Wir gehen auf das Refactoring des Monolithen, sprich die Weiterentwicklung unter Beibehaltung des Status quo als legitime Option ein und erläutern ebenso den anderen Pol, die Ablösung durch Einführung von Standardsoftware. Ferner beschreiben wir mögliche alternative weiche Übergangsszenarien, wie ein Re-Hosting, etwa durch ein Lift & Shift in eine Cloud-Umgebung, eine Portierung der Code-Basis oder eine erweiterte Nutzbarkeit des Monolithen durch Wrapping. Diese Ansätze verändern den Monolithen, aber greifen diesen nicht im Kern an. Diesen nur verändernden Szenarien stellen wir Ansätze der Ablösung gegenüber. Zum einen kann dies eine komplette Neuentwicklung (Re-Write), die erwähnte Einführung von Standardsoftware oder ein Monolith Crunching sein.

Rolf Scheuch

1 Einführung

Die technischen wie organisatorischen Herausforderungen des digitalen Wandels beschäftigen die deutschen Unternehmen momentan wie kein anderes Thema. Zunehmender Wettbewerbsdruck erfordert die Anpassung von Geschäftsmodellen und unterstützenden Geschäftsprozessen in immer kürzer werdenden Zyklen. Gleichzeitig wird durch die Globalisierung und die digitale Vernetzung der Unternehmen die Interaktion mit externen Geschäftspartnern und Kunden immer entscheidender für den Unternehmenserfolg. Diese umfassende Vernetzung benötigt das Internet of Everything[1] und umfasst Menschen, Applikationen sowie Maschinen und Endgeräte (bzw. Dinge). Erst die Konvergenz der technologischen Trendthemen Big Data, Machine Learning, schwache Künstliche Intelligenz[2], Cloud-Computing, Integration of Everything sowie auch der neuen Möglichkeiten der Mensch-Maschine-Interaktion (MMI) ermöglichen die Verfolgung und Implementierung neuer digitaler Geschäftsmodelle bzw. Ansätze von Industrie 4.0.

Als Folge der umfassenden Vernetzung im Zuge der Digitalisierung wird der verlässliche Informationsaustauch mit qualitativ hochwertigen und geschützten Daten auf einer umfassenden Plattform entscheidend. Diese Herausforderungen setzen die bislang gewachsenen Systemlandschaften unter Druck und erfordern eine grundlegende Veränderung und Transformation der unterliegenden Architekturkonzepte (► Abb. 1).

► Wir benötigen zukünftig evolutionäre Anwendungsarchitekturen[3], die durch das Architekturprinzip *Design for Change* eine stetige und gezielte Veränderung der Architektur ermöglichen.

Abb. 1: Zielbild ist eine evolutionäre Architektur

Analog zur Plattformstrategie beim Auto besteht die angestrebte Systemlandschaft aus Modulen, die in sich Innovation, Austausch und Veränderung erfahren können, ohne das Gesamtsystem zu kompromittieren, und dabei eine transparente, beherrschbare Kopplung der Systemkomponenten untereinander ermöglichen.

Eine Vielzahl deutscher Unternehmen ist in der Wartung und Stabilisierung monolithischer Systemwelten gefangen. Daneben bleibt keine Kraft, um die neuen Herausforderungen anzugehen. Lünendonk spricht in seiner Studie „Software-Modernisierung" von einer *strukturellen Zukunftsunfähigkeit* bei vielen Altsystemen: *„Trotz ihres aktuellen Nutzens für die Unternehmen ist Alt-Software strukturell zukunftsunfähig. Das liegt an ihren Charakteristika und den daraus erwachsenden Risiken für das eigentliche Geschäft."*[4]. Eine 2014 bei deutschen CIOs durchgeführte Studie belegt diese Aussage mit ihren Umfrageergebnissen.

Diese strukturelle Zukunftsunfähigkeit hinsichtlich der Weiterentwicklung und Evolution der Applikationslandschaft wird somit zu einem Risiko bei der Optimierung der Business Capabilities oder der Verfolgung neuer digitaler Geschäftsmodelle. Diese Systemlandschaften basieren oft auf umfänglichen, aufwendig zu integrierenden Applikations-Suiten, auch Monolithen genannt, und erweisen sich als veränderungsresistent. Hier muss ein Umdenken hinsichtlich der zukünftigen Systemlandschaften und Applikations-architekturen stattfinden. Nicht mehr *Big is beautiful*, sondern mit dem Denkmuster eines *Design for Change*[5] mit dem Zielbild einer evolutionären Architektur müssen wir uns zukünftig in der VUCA-Welt[6] behaupten.

Bei vielen unserer Mandanten sehen wir eine kritische Distanz zur aktuellen Situation der IT und dem Wunsch, neue Architekturkonzepte zu verfolgen, um die notwendige Geschwindigkeit zur Implementierung neuer Geschäftsideen oder Optimierungen bei bestehenden Geschäftsprozessen, etwa im Rahmen der Industrie-4.0-Initiative, zu erreichen. Hintergrund sind Geschäftstreiber, die zwar nicht neu sind, jedoch durch den Wettbewerbsdruck der Digitalisierung eine Veränderung hinsichtlich der geforderten Umsetzungsgeschwindigkeit und der Flexibilität in der Ausgestaltung erfahren.

Folgende grundlegenden und oft gleichzeitig eintretenden Geschäftstreiber stellen die Leitlinien für den Umbau und die Modernisierung der gewachsenen Systemlandschaften dar:

- Erhöhung der operativen Exzellenz: Der Wunsch, durch die neuen Konzepte von Industrie 4.0, Internet of Things, der Digitalisierung allgemein oder des Cloud-Computings die Prozess- bzw. die Stückkosten zu senken sowie Prozessinnovationen in kurzen Zyklen zu erzielen. Weiterhin soll durch Automatisierung die Nutzung von selbststeuernden Einheiten in der Produktion ermöglicht werden. Hier spielen die Ansätze des *Predictive Everything* und des Machine Learning eine wesentliche Rolle.
- Steigerung der Adaptionsfähigkeit: Die Möglichkeit, die Interaktionskosten mit Geschäftspartnern und Kunden durch digitale Lösungen und ein Ecosystem-of-value[7] zu senken sowie über eine organisatorische Ambidextrie schneller Chancen und Business Moments[8] ergreifen zu können. Hier erhofft man sich eine Erhöhung der Umsetzungsgeschwindigkeit durch Einsatz von Big Data, Cloud-Computing und den Aufbau einer Systemlandschaft als Plattform für sich selbst und Geschäftspartner. Aus Sicht der Organisation versprechen Lean-Start-up-Ansätze und eine agile IT eine höhere Beweglichkeit seitens der IT.
- Differenzierung vom Wettbewerb: Die Notwendigkeit steigt, durch eine stärkere kundenzentrische Sicht mittels einer Outside-In-Betrachtung, etwa über eine Customer Journey oder Initiativen zu Verbesserung der Customer-Experience, neue Service- und Produktinnovationen marktgetrieben und zeitnah zu implementieren. Hierbei verspricht eine bessere und an konkretem Kundenverhalten ausgerichtete User-Experience eine höhere Kundenloyalität. Durch den Einsatz von Machine Learning und Algorithmen zur Datenanalyse wird ein besseres Verständnis für die aktuellen Kundenbedürfnisse erreicht.
- Neue Märkte und Umsatzsicherung: Die Chance, über neue digitale Geschäftsmodelle bzw. Digitalisierung der bestehenden Geschäftsmodelle den bestehenden Marktzugang zu sichern oder sogar neue Kundengruppen zu erschließen. Hierbei verspricht die Digitalisierung eine Veränderung der Wertschöpfung durch digitale Produkte, die Möglichkeit, neue Produkte durch Informationen zu erfinden, etwa über Data-as-a-Product-Initiativen[9], um Wettbewerbsvorteile zu erzielen. Nicht das beste Produkt wird gewinnen – das WIE bezogen auf das konkrete Geschäftsmodell wird entscheidend. Technologische Durchbrüche revolutionieren die Wirtschaft, Disruption entsteht, alte Märkte kollabieren, neue tun sich auf – es entsteht Handlungsbedarf.

© OPITZ CONSULTING Deutschland GmbH 2020

2 Modernisierung im Spannungsfeld

Wir möchten im Weiteren nicht von Altsystemen, Legacy-Systemen etc. sprechen, sondern die Begriffe Monolith oder monolithische Applikationslandschaft verwenden. In Anlehnung an M. Fowler verstehen wir unter einem Monolithen *eine Enterprise-Applikation, die eine eigenständige logisch ausführbare Einheit darstellt und somit als Ganzes deployed werden muss.*[10]

Sämtliche notwendige Geschäftslogik liegt in der Applikation vor, inklusive einer Wiederverwendung von definierten Diensten. Dies sind die typischen Denkmuster der integrierten Gesamtsysteme mit den geschilderten Schwachstellen im Zeitablauf. Daher werden wir von Monolithen sprechen und hierunter auch monolithische Systemlandschaften oder Legacy-Systeme oder Altsysteme verstehen, die als Ganzes deployed werden müssen.

► Indikator: Ein klares Indiz für eine Erstarrung des Monolithen ist die Anzahl an Release-Wechseln mit neuer Funktionalität pro Jahr. Je weniger Release-Wechsel, desto *starrer* das Altsystem.

Die Veränderung eines Monolithen zu einer Ansammlung von eigenständigen Sub-Systemen in einer zusammenhängenden Release-Einheit könnte bereits ein erster Schritt sein, um flexible und wartbare Lösungen als Alternative für die monolithische Systemlandschaft zu schaffen. Jedoch sollte in einer zukünftigen evolutionären Applikationsarchitektur mehr drin sein: nämlich unabhängige Release-Zyklen pro Sub-System und somit eine unabhängige Produktstrategie pro Sub-System. Diese Sichtweise werden wir in diesem Whitepaper noch ausführlicher erläutern.

Wir möchten für eine zukunftssichere Applikationslandschaft jedoch einen Schritt weitergehen und legen dafür Wert auf Flexibilität und Agilität, eine rasche Reaktionsfähigkeit auf sich verändernde Anforderungen und eine kostengünstige Wartung. Das sind Ziele, die ein monolithischer Ansatz nicht erreichen kann. Sie sind nur durch eine evolutionäre Applikationsarchitektur mit Sub-Systemen zu erreichen, die voneinander unabhängige Produktstrategien verfolgen dürfen.

Ford, Parsons und Kua definieren in ihrem Buch Building Evolutionary Architectures, O'Reilly Media, 2017 eine evolutionäre Architektur wie folgt: „*an evolutionary architecture supports incremental, guided change as a first principle across multi dimensions.*" Bei der Definition haben sie sich an der biologischen Evolutionstheorie orientiert, nach der nicht der Schnellste, Größte oder Beste das Rennen gewinnt, sondern derjenige, der sich permanent, nachhaltig an die Veränderung anpassen kann.

Generell lassen sich aus den vielfältigen Herausforderungen von Monolithen vier grundlegende Treiber für eine Modernisierung erkennen:

- Sinkender Nutzen bei steigenden Kosten für Wartung, Weiterentwicklung und Betrieb, der in der Regel durch einen steigenden Technical Debt[11] verursacht wird.
- Fehlende Agilität, lange Entwicklungszeiten und schwache Time-to-Market neuer Ideen bremsen neue Geschäftsmodelle aus und verursachen einen Anwendungsstau.
- Vertrauensverlust durch eine sinkende Datenqualität, die in der Regel durch inkonsistente und veränderte Nutzung der Datenfelder entsteht, die aus *Work-Arounds* mit einem Technical Debt resultieren[12].
- Wissensabwanderung durch Überalterung der Know-how-Träger und Desinteresse der Jungen an der eingesetzten Technologie lähmen eine Weiterentwicklung.

Im Folgenden diskutieren wir die oben erwähnten Treiber etwas näher und erläutern, wie in ► Abb. 2 aufgeführt, das Spannungsfeld bei einer Modernisierung.

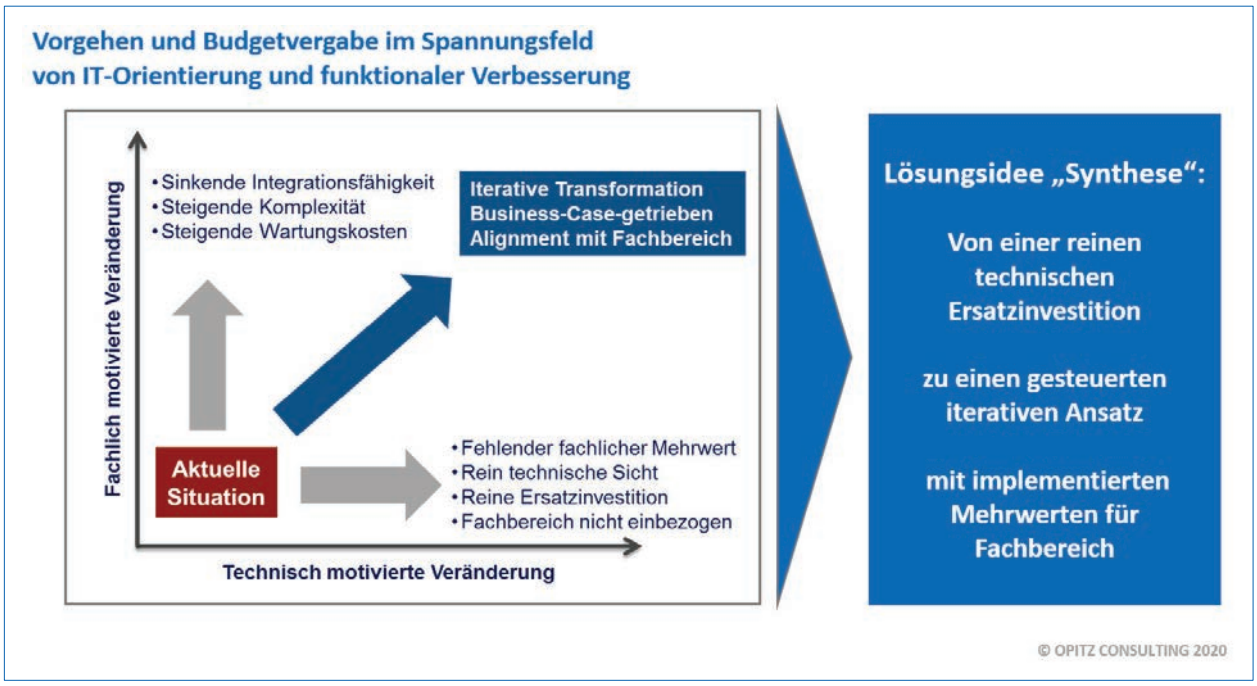

Abb. 2: Transformation im Spannungsfeld von IT-Orientierung und funktionaler Verbesserung

Sinkender Nutzen bei steigenden Kosten: Der TCO[13] für Wartung, Weiterentwicklung und Betrieb steigt kontinuierlich im Zeitablauf an. Die Implementierung gesetzlicher Regularien wird zum Wartungs-Alptraum und frisst die Entwicklungskapazitäten auf, sodass Neues nicht implementiert werden kann. In der Folge sinkt der fachliche Nutzen durch den Monolithen für den Fachbereich permanent. So gehen nicht nur Kapazitäten verloren, die nicht rechtzeitig aufgebaut werden können, sondern auch das Wissen bezüglich der Altsysteme sinkt. Dies bedingt in der Folge steigende Wartungskosten bzw. lange Liegezeiten.

Fehlende Agilität bremst Geschäftsmodelle: Aufgrund der schwierigen und aufwendigen Wartung sinkt die Chance, durch eine Weiterentwicklung der Altsysteme neue bzw. sich verändernde Geschäftsmodelle unterstützen zu können. In der Folge entstehen aus der Not heraus eigenständige Lösungen (etwa über SaaS-Lösungen), die wiederum notwendige Integrationsleistungen nach sich ziehen. Diese führen wiederum zu einer erhöhten Komplexität der Altsysteme und behindern sukzessive auch die neuen Lösungen. Ein Teufelskreis beginnt!

Vertrauensverlust durch sinkende Datenqualität: Sinkende Datenqualität hebelt eines der wichtigsten Argumente der Verfechter von Altsystemen aus. Waren in der Vergangenheit Datenqualität und eine gesicherte Abwicklung von Transaktionen und Prozessen entscheidende Argumente für die bewährten Altsysteme, schleifen sich im Zeitablauf die oben erwähnten Work-Arounds ein, was dazu führt, dass sich die Datenkonsistenz reduziert. Mehr und mehr Work-Arounds werden geschaffen, Felder re-definiert und Feldinhalte kontext-sensitiv ausgewertet. Infolgedessen sinkt im Zeitablauf die Datenqualität, und das Vertrauen in den Monolithen als Quelle der Wahrheit schwindet.

Wissensabwanderung: Im Zeitablauf verschlechtert sich der Dokumentationsgrad des Monolithen kontinuierlich und mit dem Ausscheiden langjähriger Mitarbeiter geht das nötige Wissen zur Wartung verloren. Junge Mitarbeiter sehen die Beschäftigung mit der veralteten Technologie als Abstellgleis für ihre Karriere und wechseln lieber das Unternehmen, als zu einem „Software-Archäologen“ zu werden.

Hieraus ergibt sich unmittelbar ein Spannungsfeld zweier unterschiedlicher Pole, wie auch in ► Abb. 2 erläutert: die Sicht der reinen IT-Optimierung auf der einen Seite und der Wunsch nach fachlicher Verbesserung auf der anderen Seite als konkurrierende Strategien bei der Modernisierung des Monolithen.

- Die rein technisch motivierte Veränderung ist ein typisches Vorgehen bei einer Burning Platform[14]. Der Monolith wird auf eine neue Infrastruktur migriert oder auch die Programmiersprache in eine neue (moderne) Sprache konvertiert. Diese rein technische Sichtweise betrachtet die Transformation als eine reine Ersatzinvestition und birgt keinen neuen fachlichen Mehrwert. Daher wird der

© OPITZ CONSULTING Deutschland GmbH 2020

Fachbereich auch im Vorfeld bei Entscheidung und Durchführung nicht eingebunden.

- Das andere Extrem liegt in einer rein fachlich motivierten Veränderung. Der Fachbereich hat die Budgethoheit und sieht Modernisierung rein als Erweiterung fachlicher Anforderungen. Damit sinkt die Integrationsfähigkeit und steigt die Komplexität, wodurch die Wartungskosten steigen – und sich somit der Preis pro Function-Point[15] (Funktionalität) stetig erhöht. Ein Teufelskreis beginnt!

Die Lösungsidee ist eine Synthese beider Sichtweisen in Form eines iterativen Vorgehens, um sukzessive, wie auch in ► Abb. 2 dargestellt, und in sinnvollen Inkrementen das Zielbild einer evolutionären Architektur zu verfolgen. Wichtig ist dabei, dass dies gleichzeitig in enger Abstimmung mit dem Fachbereich erfolgt, um neue fachliche Anforderungen zu ermöglichen und auch zu implementieren.

► Die Modernisierung des Monolithen verstehen wir als eine mehrjährige IT-Transformation, die auch die Arbeits-weisen und die Organisation der Fachbereiche berühren wird.

3 Unabhängige Produktstrategie als Ziel

Wie so oft ist das Neue auch bei diese Thema Fluch und Segen zugleich. Mit der Aufspaltung des Monolithen in eine Anzahl von weitestgehend autarken Sub-Systemen gehen auch die Möglichkeit und der Segen einher, endlich die Applikationsstrategie zu verändern und individueller an den Sub-Systemen und deren Business Capabilities auszurichten. War der Monolith früher eine Applikation mit definierten (meist seltenen) Release-Zyklen, so hat nun jedes Sub-System über sein Produktmanagement eine eigene Strategie und verfolgt eine eigenständige Release-Politik. Diese Sichtweisen müssen über eine Governance zur Deckung gebracht werden, ohne die Eigenständigkeit und Einzigartigkeit der Sub-Systeme zu beschneiden und die dadurch gewonnenen Vorteile zu verlieren.

Die Analysten von Gartner haben 2012 die Pace-Layered Application Strategy als Methodik *„for categorizing, selecting, managing and governing applications to support business change, differentiation and innovation"*[16] eingeführt. Kernidee war, die Möglichkeit zur Unterscheidung von Applikationsstrategien in verschiedenen Applikationsklassen zu verbessern. Hier unterscheidet Gartner die folgenden drei Klassen, wie auch aus ► Abb. 3 ersichtlich:

- Systems of Record: Systeme, die grundlegende und standardisierte Geschäftsprozesse abbilden, etwa über Standardsoftwaresysteme. In diese Kategorie fallen auch Monolithen. Die Evolution dieser Systeme verläuft langsam und geordnet mit meist wenigen Release-Zyklen. Dies ist, nach Gartner, auch richtig so, da sich die Geschäftsprozesse lediglich geringfügig oder nur unter dem Druck neuer gesetzlicher Regelungen ändern. Es gilt: Stabilität vor Geschwindigkeit!
- Systems of Differentiation: Systeme, die die Einzigartigkeit des Unternehmens in seiner Wettbewerbssituation ausprägen. Diese Systeme unterliegen einem konstanten Wandel, um die Einzigartigkeit des Unternehmens zu unterstützen. Dies bedeutet insbesondere eine ganzheitliche Integrations- und API-Strategie, um robuste Services der Systems of Record zu gewährleisten oder ein generelleres API Enablement zur Anbindung von Systemen Dritter zu ermöglichen.
- Systems of Innovation: Systeme, die neue innovative Geschäftsmodelle oder Experimente unterstützen sollen. Hier sind meist Lean-Start-up-Ansätze nötig, um die Ideen der Fachbereiche durch IT-Lösungen auszuprägen. Die neuen technischen Möglichkeiten von Big Data und Digitalisierung sind gute Beispiele für solche neuartigen Systeme, die aber in der Regel robuste Backend-Prozesse verwenden.

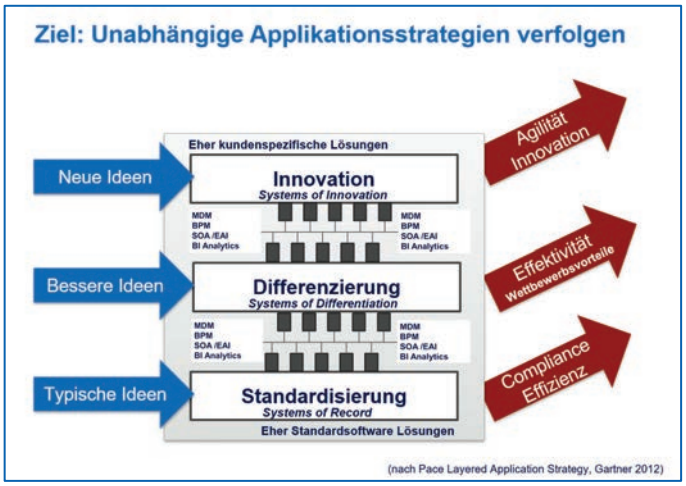

Abb. 3: Unabhängige Applikationsstrategien und Geschwindigkeiten verfolgen

Durch die Aufspaltung des Monolithen in einzelne Sub-Systeme ist es nun möglich, pro Sub-System die passende Applikationsstrategie zu verfolgen. Waren alle Sub-Systeme in einem Monolithen gefangen, so bestimmte die ausführbare Einheit mit ihrer Release-Strategie die Geschwindigkeit aller logischen Sub-Systeme im Monolithen: Der Langsamste setzte sich durch. Hier unterstützt etwa eine Microservices-Architektur eine differenziertere Applikationsstrategie und trägt somit zu einer besseren, weil schnelleren Bereitstellung neuer Funktionalität und einer besseren Unterstützung der Wertschöpfung in den unterstützten Geschäftsprozessen bei.

► Tipp: Eine genauere Betrachtung der Business Capabilities[17] in Ihrem Unternehmen sowie eine Bewertung über eine Heat-Map zeigen auf, wo Ihre Fähigkeiten marktüblich

(standardisiert), differenzierend oder innovativ sind. Hieraus folgt, dass die Sub-Systeme, die diese Business Capabilities unterstützen, bereits Kandidaten für eine unterschiedliche Applikationsstrategie sind und auch als Sub-Systeme oder sogar als unabhängige digitale Produkte gesehen werden sollten!

4 Handlungsoptionen

Entscheidet sich das Unternehmen zu handeln, so sind einige alternative Lösungsansätze zur Modernisierung sinnvoll und lassen sich grob, wie in ► Abb. 4 dargestellt, in sieben Handlungsoptionen einteilen. In der Praxis zeigt sich jedoch, dass in der Regel Mischformen verfolgt werden.

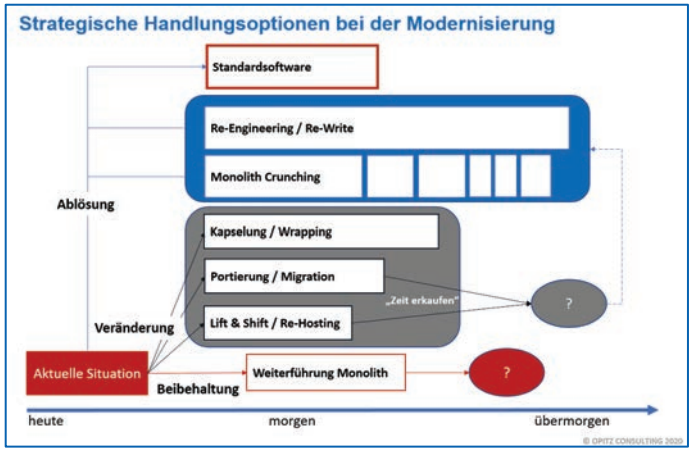

Abb. 4: Grundlegende strategische Optionen der Modernisierung

Die sieben Handlungsoptionen lassen sich über ein Clustering in drei grundlegende Ansätze einteilen, die wir im Folgenden detailliert erläutern:

- Beibehaltung: Den Status quo der aktuellen Systeme mit dem Application Lifecycle Management beibehalten und Total Cost of Ownership (TCO) wie auch Time-to-Market, etwa durch ein Refactoring oder DevOps-Ansätze, optimieren
- Veränderung: Den Monolithen im Rahmen eines Projekts verändern bzw. modernisieren, um seinen Lebenszyklus zu verlängern, neue Chancen zu ermöglichen und dadurch Zeit zu erkaufen
- Ablösung: Den Monolithen im Rahmen eines Transformationsprogramms ablösen, um eine evolutionäre Architektur als Zielbild zu verfolgen oder um Standardsoftware einzuführen

Bei der Diskussion der unterschiedlichen Handlungsoptionen werden wir diese Optionen anhand von qualitativen Kriterien bewerten.

Folgende Kriterien können Sie als Checkliste zur qualitativen Einordnung möglicher Szenarien in Ihrem Hause heranziehen:

- Transformationskosten: Es werden die gesamten Transformationskosten geschätzt und bewertet. Hierbei gehen neben den internen und externen Kosten für die Implementierung auch die Managementkosten wie auch der Aufwand für das Veränderungsmanagement bei Organisation und Personal in die Schätzung ein. (Bewertung: 1 – sehr hohe Kosten bis hin zu 5 – niedrige Kosten)
- Dauer der Transformation: Es wird die Dauer der Transformation bis zum erfolgreichen produktiven Betrieb der neuen Lösung geschätzt. (Bewertung: 1 – Lange Zeitdauer für Inbetriebnahme bis zu 5 – schnelle Produktionsübernahme)
- TCO (Total Costs of Ownership): Hier werden die Total Costs of Ownership geschätzt. Diese beinhalten den Betrieb, die Wartung und die Weiterentwicklung der Lösung wie auch die Kosten für die neue und gegebenenfalls bestehende Infrastruktur. (Bewertung: 1 – sehr hohe Kosten bis hin zu 5 – niedrige Kosten)
- Fachlicher Mehrwert: An dieser Stelle wird der Mehrwert für die Fachbereiche geschätzt. Die Kerngröße der Schätzung ist die Veränderung der Prozesseffizienz, die durch eine Senkung der Prozesskosten ausgedrückt werden soll, sowie die Möglichkeit, neue Geschäftsmodelle bzw. Business Capabilities aufzubauen. Ferner fließen auch die Datenqualität bzw. die Nutzbarkeit der Daten in diese Kategorie ein. (Bewertung: 1 – kein Mehrwert bis zu 5 – sehr hoher Mehrwert für die Organisation)
- Evolutionsfähigkeit: Dieses Kriterium ist eher strategisch zu sehen und beschreibt die erwartete Verbesserung der Agilität, der Adaptionsfähigkeit und der Veränderungsfähigkeit bei Anpassungen der IT-Lösung an sich verändernde Geschäftsprozesse und Anforderungen oder zur Unterstützung neuer Geschäftsmodelle. (Bewertung: 1 – fehlende Agilität bis zu 5 – hohe Agilität)

Bei der Bewertung bleiben wir mit dem Ranking über fünf Klassen bewusst sehr grobgranular, um die unterschiedlichen Qualitäten und damit Ziele der Handlungsoptionen klarer zum Ausdruck zu bringen.

4.1 Status quo als legitime Option

Obwohl diese Option in ► Abb. 4 etwas lapidar als *Weiterführung des Monolithen* bezeichnet wurde, kann dies eine bewusste Entscheidung sein, bei der sich Unternehmen bewusst gegen die Kosten und Risiken einer Umstellung entscheiden und ihre Kraft in die Verbesserung der Wartung und Weiterentwicklung setzen. Hierdurch sparen die Unternehmen zwar die Kosten für eine Ersatzinvestition ein und vermeiden Unruhe bei einer Umstellung.

© OPITZ CONSULTING Deutschland GmbH 2020

Jedoch ist bei dieser Handlungsoption eine Professionalisierung der Wartung zwingend nötig, um auch zukünftig Vorgaben, etwa im gesetzlichen Bereich, schneller implementieren oder dringend notwendige Anpassungen vornehmen zu können.

Hierzu gehören insbesondere Maßnahmen zur Verbesserung der Transparenz des Codes des Monolithen durch moderne Werkzeuge zur Analyse (*Software-Archäologie*) mit einem anschließenden Refactoring und Ansätze zur Testautomatisierung (etwa durch Nutzung der Log-Files der User-Interaktion). Generell verschiebt man hierdurch das Problem der strukturellen Zukunftsunfähigkeit in die Zukunft. Dies kann valide sein, falls finanzielle oder kapazitative Mittel zur Modernisierung fehlen oder die berechtigte Erwartung besteht, dass in absehbarer Zeit Standardsysteme den nötigen Funktionsumfang besitzen werden. Das Ergebnis der qualitativen Bewertung wird bildlich in -> Abb. 5 dargestellt.

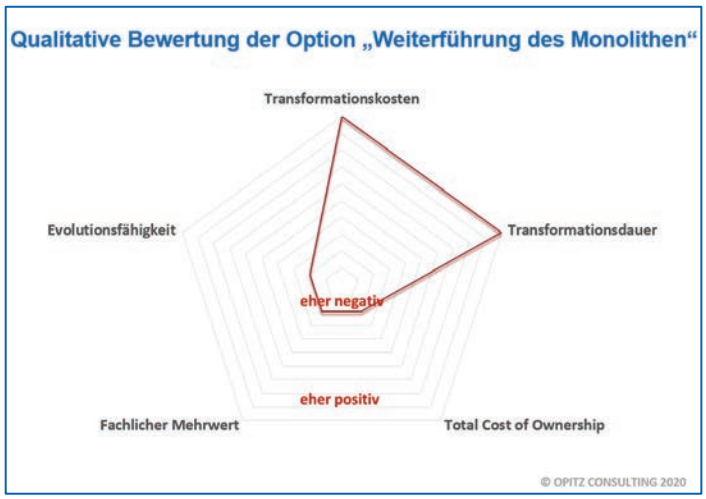

Abb. 5: Qualitative Bewertung der Option „Weiterführung des Monolithen“

Die Beibehaltung des Monolithen wird keine neuen Kosten für die Transformation erzeugen und es fällt keine Dauer für eine Umstellung an, jedoch bleiben die Schwächen bestehen: hohe und meist steigende Betriebs- und Wartungskosten (TCO) bei fehlenden Effekten für das Unternehmen und weiterhin das Problem einer fehlenden Veränderungsfähigkeit der Anwendung.

4.2 Zeit gewinnen durch Veränderung

Eine Möglichkeit mit einem überschaubaren Risiko bietet sich durch die gezielte Veränderung über eine Portierung der Legacy-Applikation an. Dieses Szenario erlaubt unterschiedliche Optionen. Obwohl wir ausführlich begründet haben, warum die IT-Transformation als Programm zu sehen ist, können die Optionen zur Veränderung häufig als Projekt mit einem klaren Anforderungskatalog gesehen werden.

Re-Hosting/Lift & Shift

Die bestehende Anwendung wird in einem Re-Hosting automatisiert auf eine andere, günstigere Infrastruktur-Plattform migriert. Hierzu zählt auch der sogenannte Lift & Shift-Ansatz zur Portierung des Monolithen ohne jegliche Code-Portierung auf eine Cloud-Infrastruktur. Dieser Ansatz verändert die Funktionalität nicht und wird auch zu keiner nennenswerten Verbesserung der Zukunftsfähigkeit oder Wartung führen. Der Ansatz fokussiert allein auf die Betriebskosten der Laufzeitumgebung und ist der typische Ansatz, um im Rahmen der Burning Platform Zeit zu gewinnen oder die Kosten im Betrieb zu reduzieren. Im Mittelpunkt steht die Ablösung von kostspieligen veralteten Infrastrukturen, etwa Mainframes, die unter Umständen sogar im Laufe der Zeit nur noch wenige Applikationen bedienen, aber noch immer immense Betriebskosten verschlingen.

Portierung/Migration

Eine weitere Option besteht in der Migration der Code-Basis, etwa durch eine automatisierte Migration des Codes selbst auf eine zukunftssichere Plattform oder die Vergabe als Gewerk an eine Off-Shoring-Einheit. Werkzeuge für eine automatisierte Migration von COBOL auf eine Java-Code-Basis existieren bereits. Bei der Portierung wird der bestehende Monolith auf eine andere, günstigere Infrastruktur-Plattform migriert. Die Erfahrung zeigt leider, dass der generierte Code nur in den seltensten Fällen wartbar ist und daher keine positiven Impulse bei der Implementierung neuer fachlicher Anforderungen möglich sind. Im Gegenteil: Durch die Unwartbarkeit der generierten Code-Basis wird die Implementierung neuer Funktionalitäten eher erschwert. Das Ergebnis der Bewertung wird bildlich in ►Abb. 6 dargestellt.

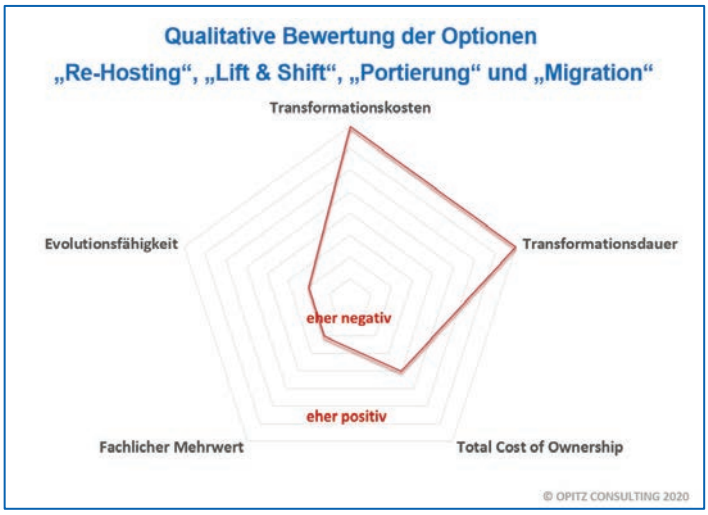

Abb. 6: Qualitative Bewertung der Optionen „Re-Hosting / Lift & Shift / Portierung / Migration“

Beim Re-Hosting werden, nach der Produktivsetzung der portierten Lösung, die Betriebskosten (und damit die TCO)

sinken, sobald die alte Infrastruktur abgeschaltet wird. Leider werden sich, außer Performance oder Stabilität, durch die neue Infrastruktur keine Mehrwerte für den Fachbereich einstellen und Verbesserungen sind nur mit erhöhtem Aufwand möglich. Die Evolutionsfähigkeit wird nicht verbessert.

- Anmerkung: Bei der Option „Weiterführung des Monolithen" und „Re-Hosting", „Lift & Shift", „Portierung" und „Migration" haben wir Transformationskosten und -dauer gleich hoch geschätzt. Dies ist natürlich rechnerisch nicht ganz korrekt, doch wir wollten in der Grafik verdeutlichen, dass Kosten und Dauer der Transformation im Verhältnis zu den nun folgenden Optionen nur einen Bruchteil des Aufwands darstellen.

Enablement durch Kapselung/Wrapping

Ein weiterer, aus unserer Sicht sehr vielversprechender Ansatz besteht in einer besonderen Form der Applikationsintegration: der Kapselung bzw. dem Wrapping oder dem API-Enablement der Funktionen oder Bildschirme des Altsystems, damit die bewährte und ausreichende Funktionalität des Monolithen in neuen Anwendungen weiterverwendet werden kann.

Das Schlagwort der *Service-fizierung* von Monolithen beschreibt diesen Ansatz recht gut. Obwohl ein Investitionsschutz bis zu einem gewissen Grad möglich ist, bleibt das grundlegende Problem der Zukunftssicherheit auf dem Tisch. Ende des letzten Jahrhunderts wurde dies häufig mit der Technik des Screen-Scraping bzw. heute auch über Robotic Process Automation (RPA)[18] erreicht, indem die Bildschirmnutzung im Hintergrund, ähnlich wie bei einem Macro-Ansatz, emuliert wurde. Somit war es möglich, neue Prozessabläufe schnell durch IT zu unterstützen. Allerdings zeigte sich dabei, dass der RPA-Ansatz zwar neue Optionen eröffnete, Wartung und Weiterentwicklung in der Folge jedoch eher noch erschwert wurden.

Interessanter erscheint da der Ansatz, die Geschäftslogik des Monolithen über Services nachzubilden und die Funktionalität den Umgebungssystemen über APIs anzubieten. Die Kopplung erfolgt nun über Services, und die meist robusten Dienste des Monolithen können nun für Neues genutzt werden. Ferner besteht die Option, sukzessive Oberflächen aus dem Monolithen durch andere technologische Ansätze zu ersetzen. Der Monolith entwickelt sich *rückwärts* zur Batch-Maschine und einem zentralen Data Store, ohne die robusten, bewährten Geschäftslogiken aufgeben zu müssen. Dies wäre eine besondere Form des Monolith Crunching. Die Evolutionsfähigkeit ist durch die starke Kopplung mit den meist komplexen, übergreifenden Datenmodellen nur bedingt gegeben und eher als ein Übergangsszenario zu sehen, um zügig Neues zu implementieren. Das Ergebnis der Bewertung wird bildlich in ►Abb. 7 dargestellt.

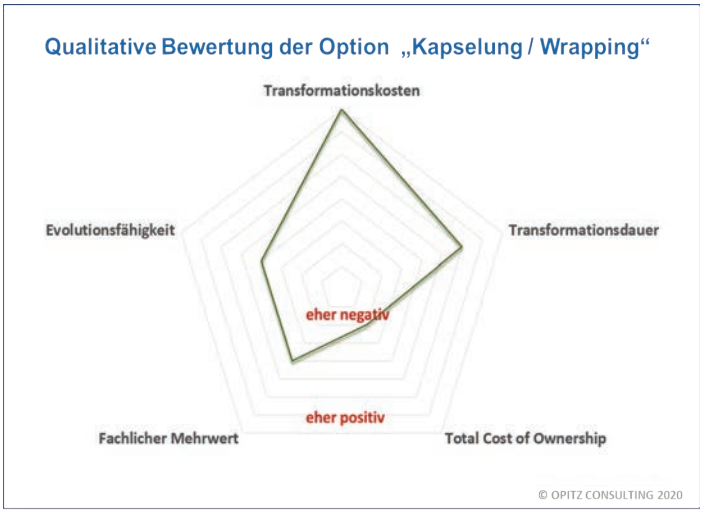

Abb. 7: Qualitative Bewertung der Option „Kapselung / Wrapping"

Beim Wrapping oder auch der *Service-fizierung* des Monolithen bleiben die ursprünglichen Betriebs- und Wartungskosten bestehen oder werden sogar noch etwas höher durch die neue Komplexität. Das Wrapping kann erste Effekte bei der Prozesseffizienz sowie bei der Implementierung neuer Funktionalitäten bewirken. Der Umfang dieser Effekte muss sich gegen die Transformationskosten und die gestiegenen Betriebs- und Wartungskosten rechnen.

- Anmerkung: In ►Abb. 4 wurden diese Ansätze plakativ als *Sicherung des morgen* durch eine *erkaufte Zeit* dargestellt. Das eigentliche Problem bleibt jedoch bestehen: fehlende Zukunftsfähigkeit des migrierten oder gekapselten Monolithen bezogen auf die Funktionalität. Die treibende Motivation für eine reine Veränderung der Laufzeitumgebung liegt in einem dringenden Handlungsbedarf bei sogenannten Burning Platforms, da der weitere Betrieb bzw. die Weiterentwicklung nicht mehr gesichert werden können. Tatsächlich kann dies eine erste Maßnahme sein, um Zeit zu gewinnen und finanzielle Mittel freizusetzen. Oft sind die erkaufte Zeit und die eingesparten Betriebskosten ein guter erster Schritt für eine gezielte Ablösung des Monolithen.

4.3 Ablösung

Die Ablösung des Monolithen ist in der Regel der einzige Weg, um eine Zukunftsfähigkeit mit der notwendigen funktionalen Innovation für die Endanwender zu erreichen und eine Evolutionsfähigkeit der Anwendungslandschaft zu erzielen. Bleibt man im Denkmuster der Ablösung des Monolithen durch ein neues System, sind in der Regel drei Optionen denkbar.

Einführung von Standardsoftware

Stellt das Unternehmen im Rahmen einer Untersuchung fest, dass alle wesentlichen Geschäftsprozesse zur Wertschöpfung

© OPITZ CONSULTING Deutschland GmbH 2020

durch den Einsatz einer Standardsoftware mit einem geringen Maß an Anpassungen (Customizing/Parametrisierung) unterstützt werden, bietet sich die Einführung der Standardlösung an. Ob diese Lösung nun on-prem bzw. gehostet eingeführt wird oder als SaaS-Lösung, ist dabei sekundär. Die Vorteile einer Lösung, deren Wartung durch einen spezialisierten externen Anbieter erfolgt, sind allgemein aus der Diskussion des *Make or Buy*[19] bekannt. Zudem wird sich eine Cloud-Lösung als SaaS auf die Gesamtsystemlandschaft auswirken. Dies gilt insbesondere in der Anfangsphase, wenn das Center of Gravity[20] noch on-prem liegt und eine SaaS-Lösung verwendet wird, sodass im Ergebnis eine hybride Applikations- und Integrationsarchitektur[21] entsteht.

Gleichwohl erfordert diese Lösung eine Transformation der Fertigungstiefe der eigenen IT. Es werden weniger Mitarbeiter im Bereich Entwicklung gebraucht, es wäre daher sinnvoll, viele Entwickler zu Beratern oder Business Analysten weiterzubilden – keine leichte Aufgabe. Ferner liegt ein nicht zu unterschätzendes Risiko in den versteckten Funktionalitäten des Monolithen, da es sich in der Praxis häufig zeigt, dass der Monolith nicht ausreichend dokumentiert ist und auch eine Analyse diese versteckten Funktionalitäten nicht entdeckt. Oft hat im Laufe der Jahre das Wissen über den Monolithen das Unternehmen verlassen, und so müssen die implementierten Regelwerke mühsam neu entdeckt und als Anforderung an die Standardsoftware aufgenommen werden. Weiterhin verführen die umfangreichen Möglichkeiten des Customizings der Standardsoftware zu einer De-facto-Eigenentwicklung, die teilweise nicht mehr über die Releases und Patches der Standardsoftware aktualisiert werden kann. In diesem Fall hat man seinen Vorteil durch den Standard verloren und ein neues Risiko geschaffen. Das Ergebnis der Bewertung wird bildlich in ►Abb. 8 dargestellt.

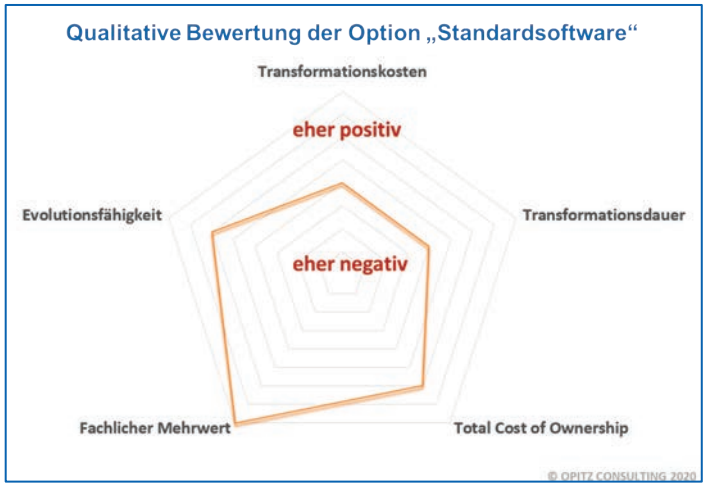

Abb. 8: Qualitative Bewertung der Option „Standardsoftware“

Generell gilt: Mit dem Übergang der Optionen von der Veränderung zur Ablösung dreht sich das Bild: Umstellungs- und Einführungsdauer sowie Transformationskosten steigen deutlich an, dem kann jedoch der gesteigerte Nutzen der Ablösung entgegengehalten werden.

Re-Engineering durch ein Re-Write

Eine kostenintensive, zeitaufwendige und risikoreiche Option ist die Neuentwicklung des Monolithen. Ob dies in Form eines Gewerks, das seinerseits mit diversen Nachteilen verbunden ist, oder mit eher iterativen oder agilen Ansätzen erfolgt, ist sekundär. Entscheidend ist die strategische Entscheidung, die Einzigartigkeit des Geschäftsmodells bzw. der Geschäftsprozesse zur Wertschöpfung durch eine individuelle Lösung auszuprägen und auf diese Weise einen Wettbewerbsvorteil durch Differenzierung zu sichern, um in der Folge sukzessive die neue Plattform auszubauen. Hierbei muss der Fachbereich mit seinen Kenntnissen zu Geschäftsprozessen und Geschäftsregeln intensiv eingebunden werden. Das kann zu einer hohen Beanspruchung der Fachbereiche oft über mehrere Jahre führen. Insbesondere werden bei der Neuentwicklung meist neue Funktionalitäten implementiert, damit die neue Lösung fachliche Mehrwerte aufweist. Dies ist in ►Abb. 9 aufgeführt.

Die Kriterien zur Bewertung der Neuentwicklung decken sich mit denen zur Bewertung eines Re-Engineering. Der Unterschied liegt meist in der Roadmap mit einer anderen Einführungsstrategie.

Bei der Neuentwicklung erfolgt die Einführung meist im sogenannten *Big-Bang-Ansatz*, sodass sich alle Vorteile bezüglich der Senkung der Betriebskosten und der Chance, neue Funktionalitäten zu implementieren, erst ab dem Roll-out bemerkbar machen. Dies trifft auch zu, falls die Neuentwicklung mittels agiler Projektansätze erfolgt. Beim eher iterativen Re-Engineering-Ansatz erfolgt eine stufenweise Einführung der Lösung, was jedoch mit der Herausforderung einhergeht, auch den Monolithen für eine bestimmte Zeit parallel zu betreiben. Dieser Punkt ist seitens der Komplexität der Replikationsmechanismen, Daten-integration und -qualität nicht zu unterschätzen. Dies gilt gleichermaßen bei der Option „Monolith Crunching“.

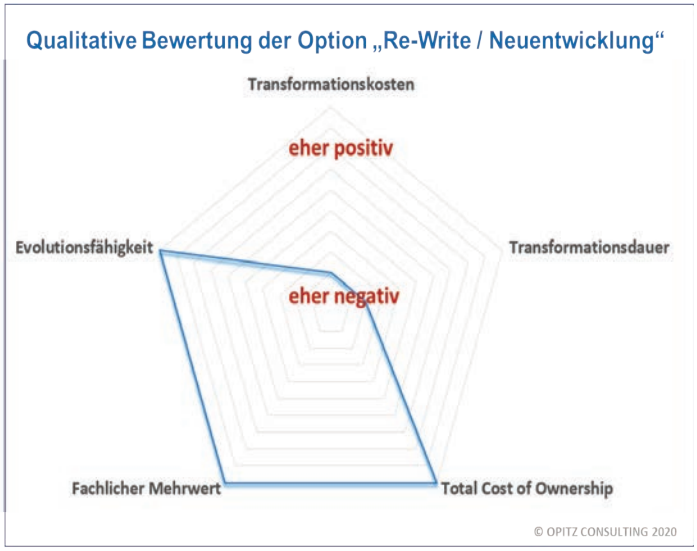

Abb. 9: Qualitative Bewertung der Option „Re-Write / Neuentwicklung“

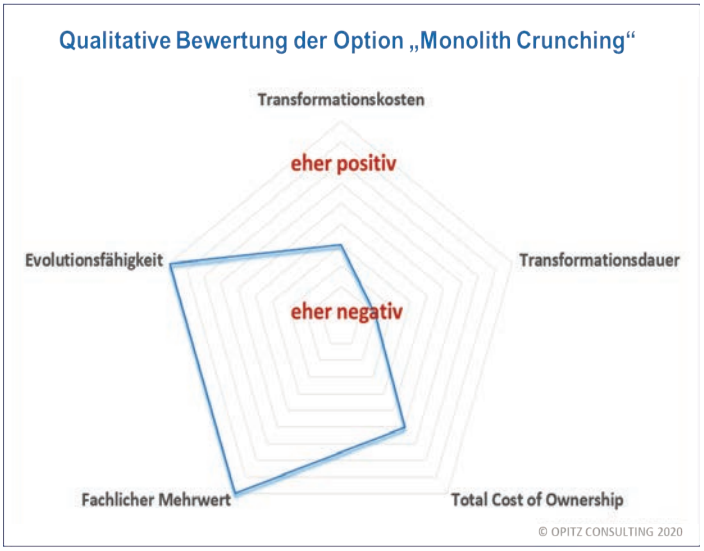

Abb. 10: Qualitative Bewertung der Option „Monolith Crunching“

Monolith Crunching

Einen Gegenentwurf zur vollständigen Neuentwicklung stellt das iterative Re-Engineering durch ein Monolith Crunching dar – ebenfalls ein iterativer Ansatz zur Ablösung. Nach einer Phase der Analyse des Reverse-Engineerings, in dem das Verständnis und die Transparenz über die technischen Strukturen des Monolithen hergestellt werden, erfolgt im Sinne eines Forward-Engineerings die sukzessive Umstellung (oder auch Neuentwicklung) der Funktionalitäten des Monolithen auf die neue Plattform. Die Herausforderung liegt hierbei in der Synchronisierung und Risikominimierung des produktiven Betriebs beider Systemwelten: zum einen die im Zeitablauf mit jeder Iteration abnehmende Funktionalität des Monolithen, andererseits die gleichzeitig wachsende Funktionalität der neuen Plattform. In der Fachwelt gibt es hierfür den recht einprägsamen Begriff der *Legacy Application Strangulation*[22], um eine sukzessive Ablösung des Monolithen zu beschreiben. Das Ergebnis der Bewertung wird bildlich in ►Abb. 10 dargestellt.

Die Bewertung dieser Option zeigt deutliche Vorteile bei der Optimierung und Abdeckung der Geschäftsprozesse sowie bei der Agilität der neuen Lösung. Im Vergleich zur Neuentwicklung, bei der der Prozess iterativ durchlaufen wird, liegt der TCO beim Monolith Crunching etwas höher und fällt somit in der qualitativen Bewertung etwas schwächer aus. Warum kann Monolith Crunching trotzdem Sinn machen? Zum einen ist das Risiko der einzelnen Iterationen deutlich niedriger als bei einem „Big Bang“ und zum anderen werden neue Komponenten früher als bisher produktiv nutzbar sein.

Zentraler Ausgangspunkt für das Monolith Crunching ist die Analyse des Monolithen durch Dekomposition und Aufzeigen der Kopplung der fachlichen wie auch technischen Module. Insbesondere nutzen Monolithen meist integrierte Datenmodelle, sodass eine Analyse der *referenziellen Verwobenheit* der Datenobjekte unerlässlich ist. Diese ausführliche Zerlegung ist notwendig, da nach der Dekomposition der kreative Akt der Identifikation von Bounded Context[23] ansteht. Dadurch erhofft man sich eine mögliche Aufspaltung des Monolithen in disjunkte Sub-Systeme, die eigenständige Strategien und Release-Zyklen besitzen. Verbindet man diese technische Sichtweise mit den fachlichen Zielen der Fachbereiche, kann man versuchen, Kandidaten dafür zu identifizieren.

Die Praxis zeigt, dass die Kopplung im Monolithen oft zu inhärent ist, sodass man auf eine Neuentwicklung des gesamten Systems ausweichen muss. Meist stellt sich heraus, dass der Monolith zwar mandantenfähig ist, aber im Hintergrund versucht, die unterschiedlichen Anforderungen in gleichartigen Datenstrukturen abzubilden. Dieses Dilemma lässt sich leider nur im Einzelfall analysieren.

© OPITZ CONSULTING Deutschland GmbH 2020

Somit lassen sich mit den beiden Ansätzen des Re-Write und des Monolith Crunching auch Überlegungen zu einer Zerlegung des Monolithen in Sub-Systeme, seien es Micro-Services, Serverless Functions oder Self-Contained Systems, vornehmen, um die angesprochenen getrennten Produktstrategien wie auch unabhängige Release-Zyklen zu verfolgen.

5 Fazit

In diesem Whitepaper haben wir unterschiedliche Handlungsoptionen bei einer Modernisierung meist monolithischer Anwendungslandschaften einander gegenübergestellt und diskutiert. Ferner konnten wir qualitative Kriterien festhalten, um die unterschiedlichen Optionen für sich zu bewerten. Hierbei wurden neben den technischen Unterschieden auch die unterschiedlichen kaufmännischen Ziele und Geschäftsstrategien der Lösungsansätze deutlich.

Steht der zukünftige Nutzen für den Fachbereich im Mittelpunkt und wird das qualitative Kriterium der Evolutionsfähigkeit (A*gilität/Adaptivität)* der zukünftigen Lösung sehr hoch bewertet, so erscheint die *Ablösung* des Monolithen oft der einzig sinnvolle Ansatz zu sein. Hierbei handelt es sich um eine mehrjährige IT-lastige Geschäftstransformation und das Vorhaben muss als ein Programm aufgefasst und geplant werden.

Sind die Betriebs- und Wartungskosten das entscheidende Kriterium, so liegt es nahe, eine *Veränderung* des Altsystems durch Portierung, Re-Hosting oder Wrapping zu verfolgen. In diesem Fall ist der Fachbereich nicht nennenswert involviert und die IT führt diese Veränderung als klassisches IT-Projekt, meist als Gewerk, eigenständig durch.

Die Ergebnisse der qualitativen Bewertung aller Handlungsoptionen sind in ►Abb. 11 als Heat-Map aufgeführt.

Ausgangspunkt der Bewertung ist die Frage: Wie wird die Veränderung resultierend aus der untersuchten Handlungsoption anhand der genannten Kriterien im Vergleich zur Option „Nicht tun / Freeze“ (Benchmark) eingeschätzt? Auch bei der Option „Nicht tun / Freeze“ wird man Wartungs- und Betriebstätigkeiten haben, aber keine Weiterentwicklungen oder neue fachliche Funktionalitäten zu implementieren versuchen. Typische Ausnahme bei diesem Ansatz wären regulatorische Anforderungen, die zwingend implementiert werden müssen. Von dieser grundlegenden (und extrem passiven) Option aus betrachtet, schätzen wir nun die Veränderung zur Option „Nicht tun / Freeze“ ein.

Eine Analyse der Heat-Map zeigt deutlich, wie auch in ►Abb. 12 grafisch als Portfolio aufbereitet, dass man das Spannungsfeld der Modernisierung in den Handlungsoptionen erkennen kann. Will man eigentlich nur das Nötigste tun, wird man sich eher für die Fortführung des Status quo, das heißt die *Option „Nicht tun / Freeze“* entscheiden. Hierbei wird man jedoch versuchen, über ein Refactoring die Technical Debts etwas zu verringern, dringende neue Funktionalitäten einzubauen oder eine Kostenreduzierung durch eine Senkung der Betriebskosten mitzunehmen. Dies ist übrigens ein typisches Szenario für ein Lift & Shift in die Cloud.

Liegt der Fokus auf einer Verfolgung von Chancen bzw. einer Differenzierung der eigenen Position im Markt durch IT-Lösungen, so sind die Optionen Einführung von Standardsoftware, Neuentwicklung bzw. Monolith Crunching der bisherigen monolithischen Anwendungslandschaft vorteilhafter.

Das Wrapping der bestehenden Lösung, um die existierenden robusten Prozesse und Funktionalitäten in *einem neuen Gewand* zu verwenden, ist eine interessante Mischform der Überlegungen, um Neues voranzutreiben, aber die Einführung einer evolutionären Architektur etwas herauszuzögern und dadurch mehr Zeit für die Analyse der Zerlegung zu haben.

Zum Abschluss möchten wir Ihnen noch eine kleine Checkliste mit auf den Weg geben, damit Sie in einem Self-Assessment bereits einige offene Punkte klären und strategische Fragestellungen für sich beantworten können.

Kleine Checkliste zur Modernisierung monolithischer Anwendungslandschaften

- Führen Sie ein Assessment durch, um die Motivation und die Ziele der Ablösung des Monolithen zu klären. Nutzen Sie die dargestellten Treiber als Kriterien.
- Betrachten und planen Sie die IT-Transformation des Monolithen als mehrjähriges Programm.
- Analysieren Sie die beschriebenen Optionen vor dem Hintergrund Ihrer Ziele und nutzen Sie die vorgestellten Kriterien, um eine qualitative Bewertung vorzunehmen.
- Führen Sie eine Dekomposition des Monolithen durch und beziehen Sie die Kopplung der Datenobjekte, Transaktionen und Integrationsflüsse in Ihre Überlegungen ein.
- Entwickeln Sie auf der Basis Ihrer Business Capabilities ein Modell der möglichen Sub-Systeme inklusive deren individueller Applikationsstrategie und vergleichen Sie dies mit der Dekomposition des Monolithen. Können Sie bereits einen passenden Bounded Context identifizieren?

© OPITZ CONSULTING Deutschland GmbH 2020

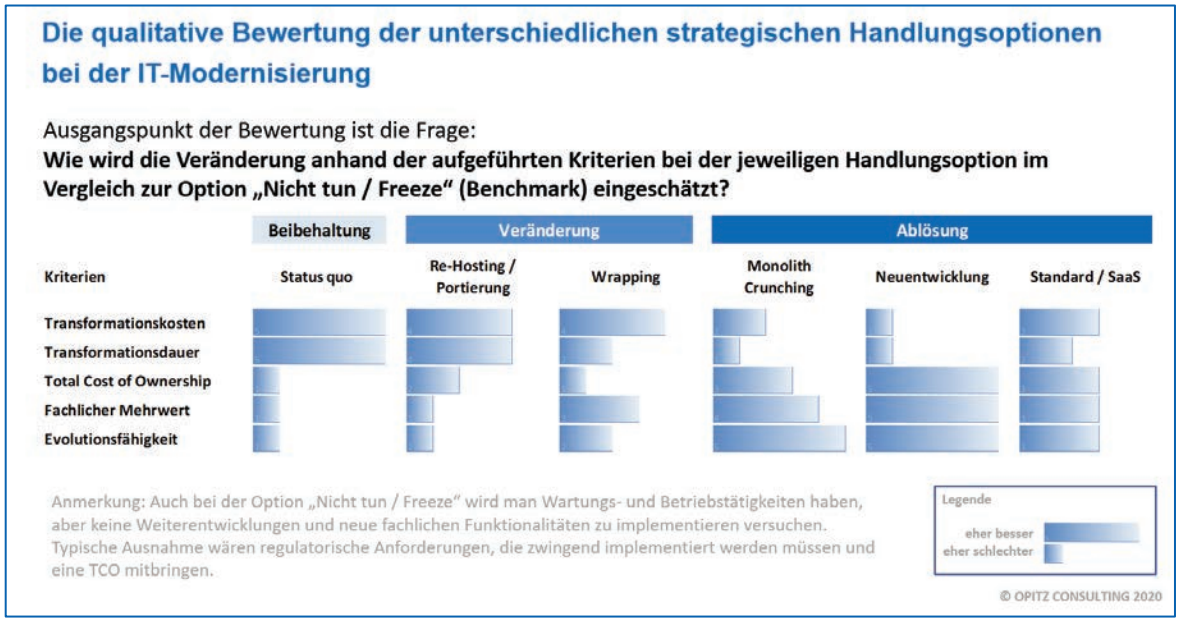

Abb. 11: Heat-Map der Handlungsoptionen

Abb. 12: Übersicht über die Handlungsoptionen und ihre qualitative Einordnung im Portfolio

© OPITZ CONSULTING Deutschland GmbH 2020

[1] https://www.i-scoop.eu/internet-of-things-guide/internet-of-everything/
[2] https://de.wikipedia.org/wiki/K%C3%BCnstliche_Intelligenz
[3] Neal Ford et al., Building Evolutionary Architectures, O'Reilly Media Inc., USA, 2017
[4] Lünendonk® White Paper, Software-Modernisierung, Lünendonk, 2015
[5] https://www.opitz-consulting.com/portfolio/digitalisierung/e-book-dynamikrobuste-architekturen-der-digitalisierung.html
[6] https://www.vuca-welt.de/
[7] https://en.wikipedia.org/wiki/Ecosystem_valuation
[8] https://www.forbes.com/sites/gartnergroup/2017/08/08/why-business-moments-are-different-in-a-digital-world/#bf05b495314f
[9] https://www.juiceanalytics.com/writing/turning-data-into-product
[10] Sam Newman, Monolith to Microservices – Evolutionary Patterns to Transform Your Monolith, O'Reilly UK Ltd., 2019
[11] https://de.wikipedia.org/wiki/Technische_Schulden
[12] Nutzt man einen Monolithen für unterschiedliche Sparten oder Geschäftsbereiche, so werden sich oft die Anforderungen und Regeln an Datenattribute unterschiedlich entwickeln. Man hilft sich meist mit Fallunterscheidungen, die, etwa aus einem Data-Warehouse-Kontext heraus, dann zu aufwendigen ETL-Strecken führen.
[13] TCO – Total Cost of Ownership, siehe https://de.wikipedia.org/wiki/Total_Cost_of_Ownership
[14] http://johnhelmerconsulting.com/how-burning-is-your-platform/
[15] https://de.wikipedia.org/wiki/Function-Point-Verfahren
[16] https://www.gartner.com/binaries/content/assets/events/keywords/applications/apn30/pace-layered-applications-research-report.pdf
[17] https://www.leanix.net/en/business-capability
[18] https://de.wikipedia.org/wiki/Robotergesteuerte_Prozessautomatisierung
[19] http://www.infolab.de/wp-content/uploads/2012/06/2010-06_ERP-Management_Pro_u_Contra_Individualsoftware_S49.pdf
[20] http://estafet.com/centre-of-gravity/
[21] Red Hat, Introduction to hybrid cloud strategy, 2019
[22] Paul Hammant, Legacy Application Strangulation: Case Studies, http://paulhammant.com/2013/07/14/legacy-application-strangulation-case-studies/(Status 4/2015)
[23] https://de.wikipedia.org/wiki/Domain-driven_Design