# Legacy System Modernization im Zeitalter von AI

Warum erfolgreiche Modernisierung heute weniger eine technische Migration als eine organisatorische Transformation ist

TEQABLE AG, Reto Maduz · Version 1.0 · Juli 2026

## Abstract

Legacy System Modernization gehört zu den strategisch wichtigsten und zugleich am häufigsten unterschätzten Transformationsvorhaben grosser Unternehmen. Über Jahrzehnte galt das zentrale Problem als ein technisches: Niemand verstand die historisch gewachsenen Systeme noch vollständig. Fehlende Dokumentation, implizites Expertenwissen und über Generationen gewachsene Abhängigkeiten machten Modernisierung zu einem langwierigen, riskanten Reverse-Engineering-Unterfangen.

Generative AI verändert diese Ausgangslage grundlegend. Sie macht in Stunden sichtbar, wofür Unternehmen früher Jahre brauchten. Doch genau dadurch verschiebt sich der eigentliche Engpass: Nicht mehr das Verstehen der Systeme ist die Hürde, sondern das Treffen von Entscheidungen über sie. Dieses Paper verfolgt anhand einer durchgängigen Fallstudie eine bewusst zugespitzte These: Legacy System Modernization ist die kontinuierliche Entkopplung geschäftlicher Fähigkeiten von ihren historischen technischen Implementierungen. Modernisierung ist damit weniger eine technische Migration als eine dauerhafte organisatorische Fähigkeit. Und sie ist keine Sonderdisziplin: Modernisierung ist ein Anwendungsfall normaler Software-Organisationsführung. Eine gewöhnliche, aber anspruchsvolle Challenge, die sauber abgewickelt werden muss und dafür expliziten Fokus braucht.

## 1. Einleitung

Die meisten Whitepaper über Legacy-Modernisierung beginnen mit Definitionen. Dieses Paper hier beginnt mit einer Geschichte. Das ist Absicht. Wer Modernisierungsprogramme aus der Nähe erlebt hat, weiss, dass die entscheidenden Momente selten in Architekturdiagrammen stattfinden, sondern in Sitzungszimmern, in denen niemand eine Entscheidung treffen will.

Der Begriff «Legacy» suggeriert ein technisches Problem: alte Programmiersprachen, veraltete Plattformen, monolithische Architekturen. Diese Sichtweise ist verständlich, aber sie führt in die Irre. Die verwendete Programmiersprache ist nicht das Problem. Die Plattform, auf der ein System läuft, ist nicht das Problem. Selbst monolithische Architekturen sind nicht das eigentliche Problem. Legacy entsteht dort, wo geschäftliche Entscheidungen über Jahre untrennbar mit ihrer technischen Implementierung verschmelzen, bis niemand mehr sagen kann, welche Regel ein bewusster Entscheid war und welche ein historischer Zufall.

Aus dieser Beobachtung folgt die These dieses Papiers: Legacy System Modernization ist die kontinuierliche Entkopplung geschäftlicher Fähigkeiten von ihren historischen technischen Implementierungen. Sie zieht sich als roter Faden durch jeden folgenden Abschnitt.

Daraus folgt eine zweite, unbequemere Feststellung: Legacy System Modernization ist kein eigenes Fachgebiet. Sie ist immer ein Thema der Software-Organisation und behandelt dieselben Fragen nach Verantwortung, Priorisierung, Entscheidungsfähigkeit und Messbarkeit, nur unter dem Druck eines Systems, das niemand mehr vollständig versteht. Wer Modernisierung zum Sonderprojekt erklärt, löst sie von genau jenen Führungsmechanismen, die sie tragen müssten. Umgekehrt gilt: «ganz normal» heisst nicht «nebenbei». Auch eine normale Challenge braucht expliziten Fokus, sonst verliert sie jedes Quartal gegen das Tagesgeschäft.

Die folgenden Seiten erzählen zunächst die Geschichte eines realistischen, wenn auch verdichteten, Modernisierungsprogramms eines Energieversorgers. Erst danach abstrahieren wir daraus ein Reifegradmodell, das Legacy Decision Maturity Model (LDMM), und leiten konkrete Erkenntnisse für die Unternehmensführung ab. Die Theorie erklärt die Geschichte, nicht umgekehrt; so, wie es gute Fallstudien seit jeher tun.

## 2. Die Fallstudie

Die folgende Fallstudie ist verdichtet, aber nicht erfunden. Sie führt Muster zusammen, wie sie in zahlreichen Modernisierungsprogrammen unterschiedlicher Branchen, von Energieversorgern über Industrieunternehmen bis zu öffentlichen Verwaltungen und Dienstleistern, zu beobachten sind. Die Zahlen sind plausibel gewählt, die Konflikte sind real. Wer ähnliche Programme begleitet hat, wird vieles wiedererkennen. Die Fallstudie spielt bewusst in einem konkreten Umfeld, ihre Aussage ist nicht daran gebunden. Ob die Ausgangslage eine Eigenentwicklung, eine Standardsoftware, eine SaaS-Plattform oder ein Cloud-Service ist, ändert am Vorgehen nichts.

### 2.1 Ein System, das seit jeher funktioniert

Beginnen wir mit dem System. Der Energieversorger, stellvertretend für viele seiner Art, betreibt sein Abrechnungs- und Kundensystem auf einer über Jahrzehnte gewachsenen, plattformgebundenen Eigenentwicklung. Eingeführt wurde es Ende der Achtzigerjahre, als der Strommarkt noch ein Monopol und «Smart Metering» ein unbekanntes Wort war. Anwendungslogik, Datenhaltung und Bedienoberflächen sind eng miteinander verzahnt und an eine einzige Plattform gebunden; Kundendienst und Ableseerfassung arbeiten bis heute mit den Masken der ersten, zweiten und dritten Generation. Jede Nacht verarbeitet ein Batchlauf zuverlässig Ablesewerte, Verbrauchsabrechnungen und Netznutzungsentgelte.

Und genau das ist der Punkt: Das System funktioniert. Es ist schnell, stabil und seit Jahren ohne nennenswerten Ausfall. In drei Jahrzehnten ist es zur unsichtbaren Infrastruktur geworden, so selbstverständlich wie Strom aus der Steckdose und ebenso wenig beachtet.

Doch die Entwickler, die es gebaut haben, sind längst pensioniert. Geblieben ist eine Handvoll Spezialisten, die das System nicht entworfen, aber jahrelang gepflegt haben. Ihre Dokumentation besteht weniger aus Diagrammen als aus Erfahrung. Steht eine Änderung an, wissen sie meist, wo man sie vornimmt. Nur selten lässt sich aber begründen, warum eine bestimmte Regel überhaupt noch existiert. Das Wissen über das System ist nicht verloren, aber es ist implizit geworden. Und implizites Wissen lässt sich weder prüfen noch übergeben.

### 2.2 Der Auslöser

Lange war das kein Problem. Ein System, das funktioniert, verlangt keine Begründung. Der Auslöser kam denn auch nicht aus der IT, sondern gleich aus mehreren Richtungen von aussen miteinander.

Der Markt forderte Echtzeit: Viertelstundenwerte aus Smart Metern, dynamische Tarife, ein Kundenportal mit tagesaktuellem Verbrauch. Was Kundinnen und Kunden von digitalen Energiedienstleistern gewohnt waren, erwarteten sie nun auch von ihrem Versorger. Das System aber war für nächtliche Stapelverarbeitung gebaut, nicht für den Sekundentakt.

Die Regulierung forderte Tempo: Neue Vorgaben mussten innert Monaten umgesetzt werden, nicht innert Jahren. Jede Anpassung am Kernsystem dauerte jedoch umso länger, je weniger Menschen es noch wirklich verstanden.

Und schlussendlich kündigte nun auch noch der letzte Entwickler, der die Tarif- und Abrechnungslogik von Grund auf kannte, seine vorzeitige Pensionierung an. Damit stand im Verwaltungsrat erstmals eine Frage im Raum, die niemand mit Sicherheit beantworten konnte: Sind wir überhaupt noch in der Lage, dieses System zu verändern? Es war keine technische Frage. Es war eine Frage über die Handlungsfähigkeit des Unternehmens.

### 2.3 Die erste Bestandsaufnahme

Das Programm begann nicht mit einer Migration, sondern mit einer Bestandsaufnahme. Erstmals kam dabei Generative AI zum Einsatz. Nicht um Code zu schreiben, sondern um den vorhandenen Code zu verstehen.

Das AI-System identifizierte innerhalb weniger Stunden über 18'000 direkte und indirekte Abhängigkeiten zwischen rund 1'300 Programmen, 480 Datenbanktabellen sowie mehreren hundert Bildschirmtransaktionen. Die daraus erzeugte Heat Map zeigte erstmals jene Komponenten, welche gleichzeitig hohe Änderungsfrequenzen und aussergewöhnlich starke Kopplungen aufwiesen. Überraschenderweise gehörten dazu nicht die erwarteten Kernmodule der Verbrauchsabrechnung, sondern mehrere historisch gewachsene Programme der Tarif- und Abrechnungsverwaltung.

Was Generationen von Entwicklern als implizites Wissen mit sich getragen hatten, lag nun als Karte auf dem Tisch. Mehr noch: Die Analyse legte Geschäftsregeln offen, die niemand mehr bewusst verantwortete. In der Tarifberechnung etwa fand sich eine Sonderbehandlung für einen Tarif, den das Unternehmen zwölf Jahre zuvor eingestellt hatte Die Regel lief seither unverändert weiter, weil niemand wusste, ob man sie gefahrlos entfernen durfte.

Genau hier wurde die These unseres Papers greifbar: Im Code steckten nicht nur Programme, sondern eingefrorene Entscheidungen. Jede Zeile war irgendwann einmal eine Antwort auf eine geschäftliche Frage gewesen. Mit der Zeit war die Frage vergessen worden, die Antwort lief weiter. AI machte diese vergessenen Entscheidungen zum ersten Mal sichtbar.

### 2.4 Transparenz schafft keine Entscheidungen

Die Heat Map war beeindruckend. Sie veränderte zunächst - nichts. Denn Transparenz ist nicht dasselbe wie Entscheidung. Das AI-System hatte gezeigt, was ist. Es konnte nicht sagen, was sein soll. Und kaum lag die Karte auf dem Tisch, begann die eigentliche Schwierigkeit jedes Modernisierungsprogramms: die Organisation.

Der CFO verlangte einen Business Case. Verständlich, doch ein belastbarer Business Case setzt voraus, dass man Aufwand und Nutzen beziffern kann. Beides war zu Beginn unbekannt. Die Forderung nach Sicherheit kollidierte mit der Natur eines Vorhabens, das gerade erst Sicherheit schaffen sollte.

Compliance blockierte. Solange nicht nachgewiesen war, dass jede regulatorisch relevante Regel im Zielbild erhalten bliebe, wollte man keiner Veränderung zustimmen. Eine nachvollziehbare Haltung, die das Programm faktisch zum Stillstand bringen konnte.

Die Fachbereiche widersprachen sich. Was für das eine Team eine überflüssige Altlast war, war für ein anderes eine geschäftskritische Ausnahme. Die Heat Map zeigte die Kopplungen; sie konnte nicht unterscheiden, welche davon Geschäft und welche historischer Zufall waren.

Die Entwickler wollten neu bauen. Für viele Ingenieure ist der Rewrite die ehrlichste Lösung: einmal sauber von vorne. Doch ein Rewrite hätte bedeutet, dreissig Jahre eingefrorener Entscheidungen neu zu treffen, ohne zu wissen, welche davon überhaupt noch gewollt waren.

Der Betrieb lehnte Microservices ab. Wer ein System mit sehr hoher Verfügbarkeit betreibt, sieht in einem verteilten System aus Dutzenden Services nicht zuerst den Fortschritt, sondern das Risiko. Auch das war keine Sturheit, sondern Erfahrung. Die Revision wiederum forderte lückenlose Nachvollziehbarkeit jedes Schritts, und senkte damit das Tempo zusätzlich. Und schliesslich wurde, wie so oft, das Budget gekürzt, noch bevor die erste Komponente migriert war.

Keiner dieser Einwände war falsch. Im Gegenteil: Jeder einzelne war rational, gut begründet und aus seiner Perspektive richtig. Und genau das ist der Kern. Legacy entsteht nicht trotz solcher Konflikte, sondern durch sie. Eine Organisation, die sich nicht auf eine gemeinsame Entscheidung einigt, friert den Status quo ein, und der Status quo ist der Code. Jede vertagte Entscheidung wird zu einer weiteren Zeile, die irgendwann niemand mehr begründen kann.

Die Transparenz hatte den Engpass also nur verschoben. Vorher fehlte das Wissen über das System. Nun fehlte die Fähigkeit, auf Basis dieses Wissens zu entscheiden. AI hatte das technische Problem weitgehend gelöst und damit das organisatorische Problem in voller Schärfe freigelegt.

### 2.5 Von Systemen zu Geschäftsfähigkeiten

Der Durchbruch kam nicht durch bessere Technologie, sondern durch eine andere Frage. Solange das Programm über Systeme sprach, über Programme, Datenbanktabellen und Bildschirmmasken, blieb es in der Zuständigkeit der IT. Und die IT konnte die offenen Fragen nicht beantworten, weil es keine technischen Fragen waren.

Der Wendepunkt war ein Perspektivwechsel: weg von Systemen, hin zu Geschäftsfähigkeiten. Nicht mehr «Was tut dieses Programm?», sondern «Welche geschäftliche Fähigkeit bildet es ab und wollen wir sie heute noch so?».

Das Unternehmen begann, seine Architektur entlang von Business Capabilities zu

beschreiben: Kunden identifizieren, Zählpunkt verwalten, Messwerte verarbeiten, Tarif berechnen, Rechnung stellen. Jede Fähigkeit erhielt eine fachliche Ownership, eine Person aus dem Business, nicht aus der IT, die verantwortete, wie diese Fähigkeit künftig aussehen sollte.

Damit verschoben sich die Konflikte aus obigem Abschnitt 2.4 an den richtigen Ort. Ob die Sonderregel der Tarifverwaltung bleiben sollte, war keine Migrationsfrage mehr, sondern eine Business-Entscheidung mit einem klaren Verantwortlichen. AI lieferte die Faktenbasis, Abhängigkeiten, betroffene Regeln, Risiken. Die Entscheidung traf das Business.

Genau das ist die Entkopplung, von der unsere These spricht: Die geschäftliche Fähigkeit «Tarif berechnen» wurde gedanklich von ihrer dreissig Jahre alten Implementierung getrennt. Erst diese Trennung machte es möglich, die Implementierung zu verändern, ohne das Business zu gefährden. Das Business kann weiterentwickelt werden, ohne von der Implementierung gefangen zu bleiben.

### 2.6 Wenn die Realität das Projekt verändert

Kein Programm überlebt den Kontakt mit der Realität unverändert. Der ursprüngliche Plan sah vor, die Businessprozesse nacheinander im Muster der schrittweisen Ablösung aus dem Monolithen zu lösen und das alte System so lange weiterlaufen zu lassen, bis die letzte Business-Funktionalität übergeben ist.

In der Praxis kam es anders. Eine kurzfristige regulatorische Vorgabe zwang das Team, eine eigentlich später geplante Business-Funktionalität vorzuziehen. Eine vermeintlich einfache Komponente, die Adressverwaltung, erwies sich als so tief mit Zählpunktzuordnung, Mahnwesen und regulatorischem Reporting verflochten, dass ihre Ablösung Monate länger dauerte als veranschlagt. Und während ein Teil der Mannschaft migrierte, musste der andere weiterhin den laufenden Betrieb sichern.

Die wichtigste Lektion betraf die Reihenfolge. Anfangs hatte das Team nach technischer Abhängigkeit priorisiert: zuerst das, was am wenigsten gekoppelt war. Es lernte, stattdessen nach Entscheidungsreife zu priorisieren: zuerst das, wofür es einen klaren fachlichen Eigner, eine klare Zielvorstellung und eine entscheidungsfähige Governance gab. Technisch einfache Fähigkeiten ohne Eigner blieben liegen; geschäftlich wichtige Fähigkeiten mit klarem Owner kamen voran, auch wenn sie technisch anspruchsvoller waren.

Auch der Einsatz von AI veränderte sich. Aus der einmaligen Bestandsaufnahme wurde ein dauerhafter Begleiter. Bei jeder Änderung prüfte das AI-System erneut die Abhängigkeiten, schlug Testfälle vor und markierte Regeln, deren Verhalten sich unbemerkt zu ändern drohte. Transparenz war kein Projektergebnis mehr, sondern ein laufender Dienst.

### 2.7 Drei Jahre später

Drei Jahre später ist das Altsystem nicht verschwunden. Und das ist kein Scheitern, sondern der Punkt. Das Ziel war nie, eine Technologie zu eliminieren. Das Ziel war, wieder entscheidungsfähig zu werden.

Einige Fähigkeiten laufen heute auf neuen Plattformen: Smart-Meter-Datenverarbeitung, Kundenportal, Tarifkonfiguration. Der Kern der Verbrauchsabrechnung läuft weiterhin auf dem Altsystem, ist nun aber hinter klar definierten Schnittstellen entkoppelt und, zum ersten Mal seit Jahrzehnten, vollständig als Sammlung bewusster Entscheidungen dokumentiert. Niemand muss ihn überstürzt ersetzen. Man kann ihn ersetzen, wenn es geschäftlich sinnvoll ist. Das ist ein entscheidender Unterschied.

Verändert hat sich weniger die Technologie als die Organisation. Es gibt fachliche Owner für jede Business-Funktionalität. Es gibt eine ständige Modernisierungs-Governance, die Entscheidungen trifft, statt sie zu vertagen. Und es gibt AI als dauerhafte Fähigkeit, die Transparenz nicht einmalig herstellt, sondern erhält.

Die messbare Wirkung zeigt sich nicht in eingesparten Codezeilen, sondern in der Geschwindigkeit: Änderungszyklen, die früher Monate dauerten, dauern heute Wochen. Die eingangs gestellte Frage des Verwaltungsrats, ob wir noch in der Lage seien, dieses System zu verändern, kann das Unternehmen heute mit Ja beantworten. Nicht, weil das alte System weg ist, sondern weil sie gelernt hat, ihre Fähigkeiten von ihrer Technik zu trennen.

## 3. Was wir daraus lernen

Aus dieser Geschichte lassen sich acht Erkenntnisse abstrahieren. Keine davon ist für sich genommen neu, aber die Fallstudie zeigt, warum sie zusammengehören und in welcher Reihenfolge sie wirken.

Die ersten sechs knüpfen unmittelbar an die Arbeit von Cartwright, Horn und Lewis an, deren Patterns of Legacy Displacement (martinfowler.com) seit Jahren dokumentieren, dass Technologie höchstens die Hälfte des Legacy-Problems ausmacht und erfolgreiche Ablösung von klaren Zielbildern, der Zerlegung in Geschäftsfähigkeiten, inkrementeller Lieferung und kulturellem Wandel abhängt. Unser Paper teilt diese Grundlage und erweitert sie in zwei Punkten: erstens um die Rolle generativer AI, die den jahrzehntealten Engpass des Verstehens weitgehend auflöst und ihn zur Entscheidung verschiebt; zweitens um das im nächsten Kapitel beschriebene Reifegradmodell, das diese Erkenntnisse nicht als Mustersammlung, sondern als Stufen organisatorischer Entscheidungsreife ordnet.

Darin liegt die eigentliche Verschiebung: Jene Autoren beschreiben, wie man ein Altsystem ablöst. Unser Paper hier fragt, wer darüber entscheidet.

### 3.1 Legacy entsteht nicht durch Technologie

Der Hauptschuldige war nie die Technologie. Legacy entstand dort, wo geschäftliche Entscheidungen über Jahre untrennbar mit ihrer Implementierung verschmolzen. Eine Technologie wird nicht alt, weil sie veraltet, sondern weil niemand mehr begründen kann, warum sie tut, was sie tut. Modernisierung beginnt deshalb nicht mit der Wahl einer neuen Plattform, sondern mit dem Wiederfreilegen alter Entscheidungen.

### 3.2 AI verändert den Engpass

Jahrzehntelang war das Verstehen der Systeme der Engpass. Generative AI hat ihn weitgehend aufgelöst: Was früher Jahre an Reverse Engineering kostete, gelingt heute in Stunden. Doch ein gelöster Engpass legt den nächsten frei. Nach der Analyse kommt die Entscheidung. Diese ist kein technisches, sondern ein organisatorisches Problem. Wer AI nur als Beschleuniger der Analyse einsetzt, verschiebt seinen Stau lediglich an die nächste Engstelle.

### 3.3 Geschäftslogik ist wertvoller als Quellcode

Der eigentliche Wert eines Altsystems liegt nicht in seinem Code, sondern in den darin eingefrorenen Business Rules. Das ist oft Wissen, das nirgends sonst dokumentiert ist. Ein Rewrite, der den Code ersetzt, aber die Logik nicht versteht, vernichtet diesen Wert. AI macht die Logik erstmals lesbar und damit übertragbar. Nicht der Code ist zu retten, sondern die Entscheidungen, die er enthält.

### 3.4 Transparenz ist Voraussetzung, nicht Lösung

Die Heat Map veränderte zunächst nichts. Transparenz ist notwendig, aber sie ist nicht hinreichend. Sie zeigt, was ist, nicht, was sein soll. Programme, die Transparenz mit Fortschritt verwechseln, produzieren beeindruckende Analysen und treffen trotzdem keine Entscheidungen. Sichtbarkeit ist der Anfang der Arbeit, nicht ihr Ende.

### 3.5 Governance wird zur Schlüsselkompetenz

Sobald AI die technische Hürde senkt, entscheidet die Fähigkeit der Organisation, zu entscheiden. Klare fachliche Ownership je Geschäftsfähigkeit, eine entscheidungsfähige Governance und der Mut, Konflikte auszutragen statt zu vertagen, werden zum eigentlichen Wettbewerbsfaktor. Jede vertagte Entscheidung schreibt sich als neue Altlast in den Code.

### 3.6 Modernisierung ist eine kontinuierliche Fähigkeit

Der Energieversorger hat sein System nicht ein letztes Mal modernisiert. Er hat gelernt, es fortlaufend zu modernisieren. Aus dem einmaligen Projekt wurde eine dauerhafte Organisationsfähigkeit: Fähigkeiten beschreiben, Entscheidungen verantworten, Implementierungen entkoppeln, Transparenz erhalten. Modernisierung ist kein Zustand, den man erreicht, sondern eine Praxis, die man pflegt.

### 3.7 Priorisierung ist nicht delegierbar

Modernisierungsprogramme scheitern selten am ersten Schritt, sondern am Umfang. Wer jahrelang auf die Gelegenheit gewartet hat, das System anzufassen, will sie nutzen: Funktionalität X wäre auch noch wichtig, Bereich Y sowieso, und Z ist ohnehin überfällig. Genau dieses Anlagern tötet Modernisierungsvorhaben von Beginn weg. Nicht weil die Wünsche unberechtigt wären,

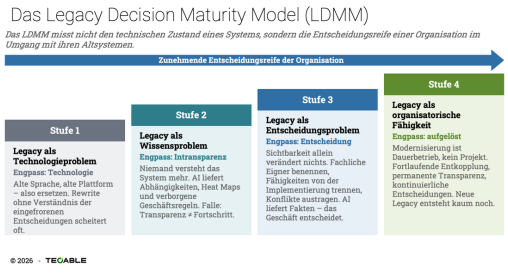
sondern weil ein Vorhaben ohne definiertes Minimalziel keinen Endpunkt hat. AI kann Abhängigkeiten berechnen und Aufwände schätzen; sie kann nicht entscheiden, was weggelassen wird. Die Priorisierung auf das wirklich Notwendige bleibt die am wenigsten delegierbare Führungsaufgabe des ganzen Programms.

### 3.8 Modernisierung ist ein Thema der Software-Organisation

Nichts in dieser Geschichte ist legacy-spezifisch. Fachliche Ownership, entscheidungsfähige Governance, Scope-Disziplin, Transparenz als laufender Dienst, Messbarkeit von Delivery und Teamgesundheit: das sind die Grundfragen jeder Software-Organisation. Legacy verschärft sie, erzeugt sie aber nicht. Daraus folgt beides: Modernisierung braucht keine eigene Methodenwelt, sondern eine Organisation, die ihr Handwerk beherrscht. Und dennoch braucht es den expliziten Fokus auf die Modernisierung, weil sie sonst gegen jede kurzfristig sichtbarere Anforderung verliert.

## 4. Das Reifegradmodell: Legacy Decision Maturity Model (LDMM)

Rückblickend durchläuft die Fallstudie vier Stufen der Reife, die wir im Legacy Decision Maturity Model (LDMM) zusammenfassen. Es beschreibt bewusst nicht den technischen Zustand eines Systems, sondern die Entscheidungsreife einer Organisation im Umgang mit ihren Altsystemen.




### Stufe 1 - Legacy als Technologieproblem

Auf der ersten Stufe gilt Legacy als rein technisches Problem: alte Sprache, alte Plattform, also ersetzen. Der vermeintliche Engpass ist die Technologie, die vermeintliche Lösung Migration oder Rewrite. AI spielt kaum eine Rolle oder wird als blosses Werkzeug zur Code-Konvertierung missverstanden. Governance erschöpft sich in Projektsteuerung. Organisationen auf dieser Stufe scheitern häufig, weil sie Code ersetzen, ohne die darin enthaltenen Entscheidungen zu verstehen.

### Stufe 2 - Legacy als Wissensproblem

Auf der zweiten Stufe erkennt die Organisation, dass das eigentliche Problem fehlendes Wissen ist: Niemand versteht das System mehr vollständig. Der Engpass ist Intransparenz. Hier entfaltet AI ihren ersten grossen Nutzen mit Abhängigkeitsanalysen, Heat Maps und dem Offenlegen verborgener Business Rules. Die Organisation gewinnt Sichtbarkeit. Die typische Falle dieser Stufe: Man verwechselt Transparenz mit Fortschritt und bleibt in der Analyse stecken.

### Stufe 3 - Legacy als Entscheidungsproblem

Auf der dritten Stufe wird klar, dass Sichtbarkeit allein nichts verändert. Der Engpass verschiebt sich vom Wissen zur Entscheidung. Die Organisation muss fachliche Owner benennen, Geschäftsfähigkeiten von ihrer Implementierung trennen und die Konflikte zwischen CFO, Compliance, Business und Betrieb tatsächlich austragen. AI liefert die Faktenbasis, das Business trägt die Entscheidung. Wer diese Stufe erreicht, modernisiert nicht mehr Systeme, sondern Entscheidungen.

### Stufe 4 - Legacy als organisatorische Fähigkeit

Auf der vierten Stufe ist Modernisierung kein Projekt mehr, sondern eine dauerhafte Fähigkeit. Die Organisation entkoppelt fortlaufend Business-Funktionalitäten von ihren technischen Implementierungen, hält Transparenz über AI permanent aufrecht und entscheidet kontinuierlich statt einmalig. Legacy im alten Sinn entsteht hier kaum noch, weil Entscheidungen bewusst getroffen und dokumentiert werden. Diese Stufe ist kein Endzustand, sondern eine Betriebsweise.

| Stufe | Sicht auf Legacy | Engpass | Rolle von AI |
|---|---|---|---|
| 1 | Technologie-problem | Technologie (vermeintlich) | gering - Code-Konvertierung |
| 2 | Wissens-problem | Intransparenz | Analyse, Transparenz herstellen |
| 3 | Entscheidungs-problem | Entscheidungs-fähigkeit | Faktenbasis für Entscheide |
| 4 | Organisatorische Fähigkeit | Kontinuität | dauerhafter Transparenz-dienst |

## 5. Der Entscheidungsraum: Make or Buy und fünf Strategien

Bevor eine Organisation über Modernisierungsstrategien spricht, muss sie eine vorgelagerte Frage beantworten: Wollen wir das eigene System weiterentwickeln oder ein Standardprodukt beschaffen? Dieser Make-or-Buy-Entscheid hat mit Legacy System Modernization nur indirekt zu tun, bestimmt aber, ob überhaupt eine Modernisierungsfrage vorliegt. Wer ihn überspringt, führt eine Architekturdebatte über ein System, das das Unternehmen unter Umständen gar nicht selbst betreiben will.

Beantwortbar wird die Frage erst mit der Faktenbasis aus der Bestandsaufnahme: Wie hoch sind die Gesamtkosten über die Lebensdauer, wie viel der tatsächlich benötigten Geschäftsregeln deckt ein Standardprodukt ab, und was kosten die Sonderfälle, die es nicht abdeckt? Genau hier liegt der zweite grosse Nutzen von AI: Sie liefert innert Wochen eine belastbare Liste dessen, was das Altsystem wirklich tut, und macht damit den Make-or-Buy-Entscheid erstmals faktenbasiert statt weltanschaulich.

Die zweite vorgelagerte Frage lautet: Sind wir überhaupt noch in der Lage, modular vorzugehen? «Teile und herrsche» ist der zentrale Ansatz jeder Ablösung. Ein Big Bang ist nur in den wenigsten Fällen der richtige Weg. Fehlt die Modularisierbarkeit, kippt die Diskussion in der Regel zurück auf Make or Buy.

Erst danach steht die Strategiewahl an. Fünf Optionen sind gebräuchlich, geordnet von der am wenigsten zur am stärksten transformativen:

Rehost (Lift & Shift) verlagert das System auf eine neue Plattform, ohne fachlich etwas zu verändern. Schnell, günstig, risikoarm und ohne Wirkung auf die eingefrorenen Entscheidungen. Wer nur rehostet, betreibt sein Legacy in einer moderneren Umgebung weiter.

Refactor (Strangler Fig) setzt eine API-Fassade vor das Altsystem, stellt neue Komponenten daneben und löst es schrittweise ab. Das ist die Standardstrategie, wenn Modularisierung überhaupt möglich ist. Ihr Preis ist der Parallelbetrieb: zwei Systeme, doppelte Betriebskosten, bis die letzte Business-Funktionalität übergeben ist.

Re-architect (Event-Driven / Microservices) ersetzt direkte Aufrufe durch Ereignisse über einen Message Broker. Das ergibt den höchsten strukturellen Gewinn, aber bedeutet auch höchste Anforderungen an Betriebs- und Governance-Reife.

Replace - Big Bang Rewrite baut neu und schaltet in einem Cutover um. Diese Option verlangt, dreissig Jahre eingefrorener Entscheidungen in einem Zug neu zu treffen, meist ohne zu wissen, welche davon noch gewollt sind.

Replace - Buy führt den Make-or-Buy-Entscheid als Modernisierungsstrategie aus: Standardprodukt statt Eigenentwicklung, mit allen Folgefragen zu Sonderfällen, Schnittstellen und Datenmigration.

Entscheidend ist weniger, welche dieser Strategien «die richtige» ist, als dass die Wahl bewusst als Geschäftsentscheidung getroffen und danach gehalten wird. Ein Wechsel mitten im Programm vernichtet den Fortschritt beider Wege. Ebenso vorab zu entscheiden ist die unscheinbarste und folgenreichste Frage der ganzen Transformation: Wo werden neue Anforderungen während der Umstellung gebaut: Im Altsystem oder im neuen System? Bleibt diese Frage offen, beantwortet sie jedes Team einzeln und jedes Mal anders.

## 6. Was man messen muss: DORA, SPACE und der Modernisierungsfortschritt

Wer Modernisierung als dauerhafte Fähigkeit führt, braucht dauerhafte Messgrössen. Ein Programmstatus in Prozent genügt dafür nicht, denn er misst Aktivität und nicht Wirkung. In der Praxis haben sich vier Messebenen bewährt, von denen die ersten beiden ein Paar bilden: DORA beschreibt, was das Umfeld von einem Team sieht, SPACE, wie das Team seine Arbeit selbst erlebt. Diese Unterscheidung zwischen Teamaussensicht und Teaminnensicht stammt aus der Praxis eines unserer Kunden und ist der schnellste Weg, das Gespräch über Kennzahlen aus der Rechtfertigungsschleife zu holen.

### 6.1 Teamaussensicht: Delivery-Performance (DORA)

Die vier DORA-Kennzahlen messen, was ein Umfeld tatsächlich wahrnimmt.

- Deployment Frequency: Wie oft wird geliefert?
- Lead Time for Changes: Wie lange braucht eine Änderung von der Idee bis in die Produktion?
- Change Failure Rate: Wie oft richtet eine Änderung Schaden an?
- Time to Restore Service: Wie schnell herrscht nach einem Vorfall wieder Normalbetrieb?

Für Modernisierungsvorhaben sind diese DORA Metriken doppelt nützlich. Erstens operationalisieren sie mit der Lead Time die Forderung, Erfolg an der Änderungsgeschwindigkeit zu messen. Zweitens verhindern sie den klassischen Selbstbetrug, denn ein Programm, das migrierte Programme zählt, aber Lead Time und Change Failure Rate nicht verbessert, hat Code verschoben und keine Handlungsfähigkeit gewonnen.

### 6.2 Teaminnensicht: Developer Experience (SPACE)

DORA sagt nichts darüber, ob eine Leistung tragfähig ist. Das leistet SPACE mit fünf Dimensionen: Satisfaction & Wellbeing, Performance, Activity, Communication & Collaboration sowie Efficiency & Flow. Ergänzen wir diese SPACE Metriken um Cognitive Load aus Team Topologies, erhalten wir sehr aussagekräftige Grössen für den Legacy-Kontext. Ein Team, das ein System pflegt, es aber nur zur Hälfte versteht, arbeitet dauerhaft an der Grenze seiner kognitiven Belastung.

Der typische Legacy-Verlauf ist deshalb: Die Aussensicht bleibt eine Weile passabel, die Innensicht kippt zuerst. Wer nur DORA misst, erkennt die Erosion erst, wenn sie in den Lieferzahlen ankommt. Meist wir die Erosion dann zusätzlich noch durch Kündigungen verstärkt, was zu zusätzlichem Wissensverlust führt. Umgekehrt ist eine intakte Innensicht der einzige belastbare Frühindikator dafür, dass ein Vorhaben von mehreren Quartalen auch durchgängig tragfähig ist.

### 6.3 Modernisierungsfortschritt und Geschäftswert

Beide Sichten beschreiben die Organisation, nicht das Vorhaben. Dafür braucht es zwei weitere Ebenen. Der Modernisierungsfortschritt misst den Anteil migrierter Funktionalität, die Anzahl aktiver Schnittstellen zwischen Alt und Neu, Testabdeckung und Code-Qualität sowie den Cloud-Native-Reifegrad. Der Geschäfts- und Betriebswert misst TCO-Reduktion, Time-to-Market, SLA-Einhaltung und Verfügbarkeit sowie Security Findings und Patch-Lag.

Besondere Aufmerksamkeit verdient die Anzahl aktiver Alt-/Neu-Schnittstellen. Sie ist die ehrlichste Kennzahl des Parallelbetriebs: Steigt sie über mehrere Quartale, ohne je zu sinken, ist keine Ablösung im Gang, sondern eine zweite Landschaft im Entstehen.

### 6.4 Warum erst die Kombination trägt

Einzeln ist jede dieser Grössen manipulierbar. Deployment Frequency lässt sich durch kleinere Deployments steigern, migrierte Funktionalität durch das Umkodieren von Trivialem, Zufriedenheit durch Entlastung vom Schwierigen. Erst gemeinsam betrachtet ergeben die vier Ebenen ein belastbares Bild und taugen zur Führung statt nur zur Berichterstattung.

Im Reifegradmodell markiert das den Unterschied zwischen Stufe 3 und Stufe 4. Auf Stufe 3 entscheidet eine Organisation bewusst; auf Stufe 4 sieht sie fortlaufend, was ihre Entscheidungen bewirken. Aussen- und Innensicht dauerhaft nebeneinander zu erheben ist deshalb keine Reporting-Übung, sondern die Voraussetzung dafür, dass Modernisierung zur Betriebsweise wird.

## 7. Konsequenzen für die Unternehmensführung

Wenn der Engpass von der Technik zur Entscheidung wandert, ändern sich die Aufgaben der Führung. Aus der Fallstudie und dem LDMM folgen sieben konkrete Konsequenzen.

**Modernisierung ist Chefsache:** Solange das Thema als IT-Projekt geführt wird, scheitert es an Fragen, welche die IT nicht beantworten kann. Die Verantwortung für Geschäftsfähigkeiten gehört ins Business und auf die Führungsebene.

**Investieren Sie in Entscheidungsfähigkeit, nicht nur in Werkzeuge:** AI senkt die Analysekosten dramatisch. Den Unterschied macht künftig, wie schnell und wie konsequent eine Organisation auf Basis dieser Analysen entscheidet. Governance ist die neue Knappheit.

**Setzen Sie AI dauerhaft ein, nicht als einmaliges Audit:** Transparenz, die nur einmal hergestellt wird, veraltet mit der nächsten Änderung. Wer AI als laufenden Dienst in den Entwicklungs- und Betriebsprozess einbettet, verhindert, dass neue Legacy entsteht.

**Geben Sie jeder Business-Funktionalität einen Owner:** Eine Business-Funktionalität ohne fachlichen Verantwortlichen lässt sich weder bewusst weiterentwickeln noch von ihrer Implementierung entkoppeln. Ownership ist die organisatorische Voraussetzung jeder Entkopplung.

**Messen Sie Erfolg an der Änderungsgeschwindigkeit:** Nicht die Zahl migrierter Programme zeigt den Fortschritt, sondern die Zeit, die eine geschäftlich gewünschte Veränderung von der Idee bis zur Produktion benötigt. Diese Kennzahl misst die wahre Modernisierungsreife. Damit wir aber nicht Tempo auf Kredit kaufen, erheben wir auch noch die Team Innensicht.

Beantworten Sie zuerst die Zielfrage und definieren Sie ein Minimalziel. «Was ist überhaupt das Ziel?» ist die einzige Frage, die vor allen anderen geklärt sein muss. Die grösste Gefahr für ein Modernisierungsvorhaben ist danach nicht der Widerstand, sondern die Zustimmung: Jeder Bereich möchte seine lange aufgeschobene Anforderung im gleichen Zug erledigt haben. Legen Sie zu Beginn fest, welcher kleinste Umfang den eigentlichen Zweck erfüllt, und behandeln Sie jede Erweiterung als eigenen Entscheid mit eigener Begründung. Ein Frühwarnsystem aus wenigen, regelmässig gelesenen Kennzahlen ist günstiger als jede spätere Rettungsaktion.

Führen Sie die Diskussion in der Sprache Ihrer Unternehmenskultur. Ob ein Vorhaben top-down verordnet, im Konsens erarbeitet oder über Zwischenziele legitimiert wird, prägt seinen Verlauf so stark wie jede Architekturentscheidung. Der Weg zum Management-Support bleibt derselbe: von der technischen Analyse zur Fachlichkeit, von der Fachlichkeit zu den Geschäftsprozessen, und erst dann in die Führungsdiskussion. Dort zählen keine Details, sondern Optionen mit ihren Konsequenzen. AI ist dabei weniger Analysewerkzeug als Argumentationshilfe: Sie liefert die Antworten auf jene gezielten Fragen, mit denen gut vorbereitete Führungskräfte in solche Sitzungen kommen.

## 8. Die Simulation: Entscheiden unter Konsequenzen

Um unsere These erfahrbar und damit prüfbar zu machen, haben wir sie in eine Management-Simulation überführt. Ihr Name ist Programm: «Software Organization Simulation» und nicht «Legacy Simulation». Die Spieler übernehmen die Rolle eines CIO, CTO oder Head of Engineering und steuern über zehn bis zwölf Quartale eine Software-Organisation. Die Modernisierungsherausforderung ist nicht das Spiel, sondern ein Ereignis darin. Genau so verhält es sich in der Realität.

Software Organization Simulation: https://managility.ch

Drei Mechanismen bilden die Kernaussagen unseres Papers im Spiel ab.

- Erstens ist die Herausforderung (einer Legacy System Modernization) von Beginn an sichtbar und lässt sich maximal zweimal verschieben. Jede Verschiebung erhöht den Schweregrad und verwandelt einen abfederbaren Wissensverlust in einen harten, unvermeidbaren. Vorlaufzeit ist die einzige Ressource, die sich nicht nachbeschaffen lässt.
- Zweitens ist die Strategiewahl aus Kapitel 5 zu Beginn bindend: Wer wechseln möchte, hat vorher zu wenig entschieden.
- Drittens sind Delivery-Performance und Zufriedenheit keine Anzeigen, sondern Multiplikatoren auf die verfügbare Kapazität. Ein Team mit erodierter Innensicht liefert messbar weniger, genau wie in Kapitel 6 beschrieben.

Gewonnen wird in der Simulation nicht durch eine abgeschlossene Migration, sondern durch fünf von sieben gleichzeitig erfüllten Bedingungen: Geschäftswert, technische Schuld, Delivery-Performance, Wissen, Architektur, Kundenzufriedenheit und die bewältigte Herausforderung. Keine davon ist rein technisch, keine lässt sich isoliert optimieren, und die volle Punktzahl ist nicht vorgesehen. Damit leistet die Simulation, was ein Whitepaper nicht kann: Sie macht die Kosten des Vertagens im Zeitverlauf spürbar. Es gibt keine richtigen Züge sondern nur Konsequenzen.

## 9. Fazit - Die eigentliche Modernisierung

Legacy-Systeme werden häufig mit alter Technologie verwechselt. Unser Paper argumentiert bewusst anders. Legacy entsteht nicht durch eine bestimmte Programmiersprache. Legacy entsteht nicht durch eine bestimmte Plattform. Legacy entsteht auch nicht durch monolithische Architekturen. Legacy entsteht dort, wo geschäftliche Entscheidungen über Jahre oder Jahrzehnte untrennbar mit ihrer technischen Implementierung verschmelzen.

Generative AI verändert diese Ausgangslage grundlegend. Nicht, weil AI Altsysteme automatisch modernisieren könnte, sondern weil AI erstmals sichtbar macht, welche Entscheidungen ein Unternehmen über die Jahre tatsächlich getroffen hat.

Damit verschiebt sich die eigentliche Herausforderung: von der Analyse zur Entscheidung, von der Technologie zum Geschäftsmodell, von der Softwareentwicklung zur Unternehmensführung.

Und sie verschiebt sich an einen Ort, an dem sie ohnehin hingehört: in die Führung der Software-Organisation. Legacy System Modernization ist kein Spezialgebiet neben der Softwareentwicklung, sondern deren Normalfall unter erschwerten Bedingungen. Wer sie zum exotischen Grossvorhaben erklärt, gibt ihr eine Sonderrolle, die sie nicht braucht, und entzieht ihr gleichzeitig den Fokus, den sie braucht.

Legacy System Modernization ist deshalb kein Projekt. Es ist die Fähigkeit einer Organisation, ihre geschäftlichen Fähigkeiten kontinuierlich von ihren historischen technischen Implementierungen zu entkoppeln und bewusst weiterzuentwickeln. Genau diese Fähigkeit wird künftig über die Innovationsgeschwindigkeit grosser Unternehmen entscheiden.

**Quellen und weiterführende Literatur**

*Die in diesem Paper geschilderte Fallstudie ist ein verdichtetes, illustratives Szenario; die genannten Kennzahlen sind plausibel gewählt und stammen nicht aus einem einzelnen realen Projekt. Die folgenden Quellen belegen die zugrunde liegenden Konzepte sowie dokumentierte Praxisergebnisse AI-gestützter Modernisierung. Die Erkenntnisse zu Make-or-Buy, Priorisierung und Unternehmenskultur stammen zusätzlich aus einem Fachworkshop mit Kundenvertretern (Juni 2026); die Unterscheidung von Teamaussen- und Teaminnensicht über DORA und SPACE geht auf den Input eines unserer Kunden zurück.*

**Grundlagen und Konzepte**

Fowler, M. (2004): Strangler Fig Application. martinfowler.com/bliki/StranglerFigApplication.html - schrittweise Ablösung von Altsystemen statt «Big-Bang»-Rewrite.

Cunningham, W. (1992): The WyCash Portfolio Management System. OOPSLA '92 Experience Report - Ursprung des Begriffs «Technical Debt».

Feathers, M. (2004): Working Effectively with Legacy Code. Prentice Hall.

Evans, E. (2003): Domain-Driven Design. Tackling Complexity in the Heart of Software. Addison-Wesley.

Newman, S. (2019): Monolith to Microservices. O'Reilly.

Conway, M. E. (1968): How Do Committees Invent? Datamation - «Conway's Law» zum Zusammenhang von Organisation und Systemarchitektur.

Skelton, M.; Pais, M. (2019): Team Topologies. IT Revolution - fachliche Ownership und Team-zentrierte Verantwortung.

Business Architecture Guild (laufend): A Guide to the Business Architecture Body of Knowledge (BIZBOK Guide) - Business-Capability-Modellierung.

Ford, N.; Parsons, R.; Kua, P. (2017): Building Evolutionary Architectures. O'Reilly - Architecture Fitness Functions als automatisierte, kontinuierliche Prüfgrössen, die architektonische Eigenschaften messbar erhalten; vgl. Thoughtworks Technology Radar, «Architectural fitness functions».

Kazman, R.; Klein, M.; Clements, P. (2000): ATAM - Method for Architecture Evaluation. SEI/CMU, CMU/SEI-2000-TR-004 - strukturierte, stakeholder-getragene Bewertung von Architektur-Entscheidungen und ihrer geschäftlichen Zielkonflikte.

Bennatan, E. M. (2006): Catastrophe Disentanglement. Getting Software Projects Back on Track. Addison-Wesley - zehnstufiger Prozess für entgleiste Vorhaben; hier relevant für Minimalziele und Frühwarnsystem.

**AI-gestützte Legacy-Modernisierung**

QuantumBlack, AI by McKinsey (2025): Overcoming legacy tech through agentic AI (LegacyX) - Praxisprojekt: in fünf Wochen über 4'000 Geschäftsregeln aus Legacy-Code extrahiert; 70 % der Geschäftslogik in acht Dateien konzentriert.

McKinsey & Company (2025): McKinsey's LegacyX - Rejuvenating legacy infrastructure with agentic AI. mckinsey.com

McKinsey & Company (2026): The AI revolution in software development / Rewiring the business for value - «modernization becomes business as usual».

CIO.com (2025): Using AI to modernize mainframes - Turning legacy tech into a strategic advantage.

Legacy Modernization with AI - Mainframe modernization (2025). arXiv:2512.05375.

Thoughtworks Technology Radar (2024-2025): «Using GenAI to understand legacy codebases» und «GenAI for forward engineering» - AI-gestützte Erschliessung verborgener Geschäftsregeln und schrittweise Modernisierung von Altsystemen; ergänzend Fowler, M. et al.: Legacy Modernization meets GenAI. martinfowler.com/articles/legacy-modernization-gen-ai.html

**Metriken und Delivery-Performance**

Forsgren, N.; Humble, J.; Kim, G. (2018): Accelerate. The Science of Lean Software and DevOps. IT Revolution - empirische Grundlage der vier DORA-Kennzahlen.

Forsgren, N.; Storey, M.-A.; Maddila, C.; Zimmermann, T.; Houck, B.; Butler, J. (2021): The SPACE of Developer Productivity. ACM Queue 19(1) - fünfdimensionales Rahmenwerk für Developer Experience.

DORA / Google Cloud (laufend): State of DevOps Report. dora.dev - jährliche Benchmarks zu Deployment Frequency, Lead Time, Change Failure Rate und Time to Restore Service.

Skelton, M.; Pais, M. (2019): Team Topologies. IT Revolution - Cognitive Load als Belastungsgrösse der Teaminnensicht (vgl. Grundlagen).