Sonderdruck
aus IT Spektrum 04/2022

Ausgabe 04|2022

Deutschland € 12,90 Österreich € 13,90 Schweiz sfr 22,20

4 194164 112900 04

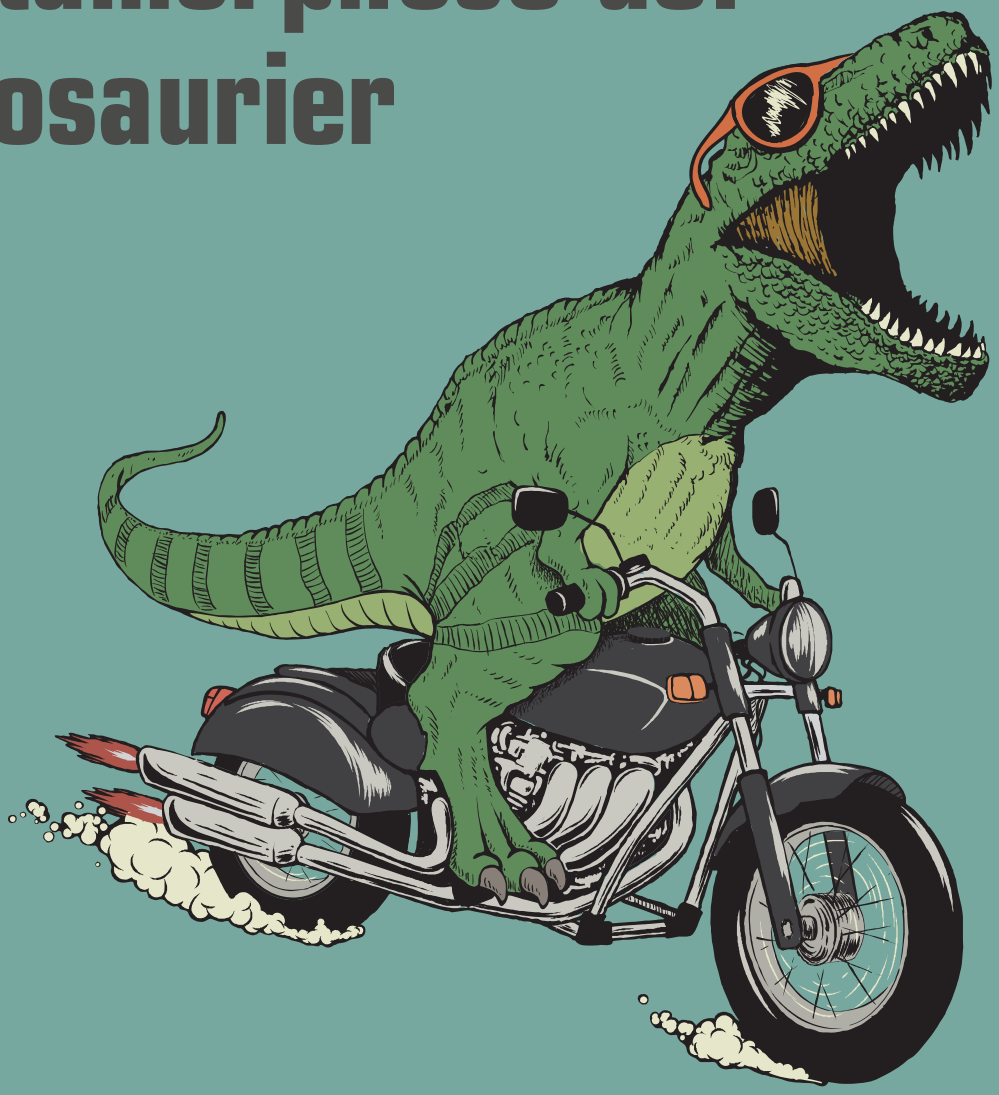
www.ITSpektrum.

IT Spektrum

vormals OBJEKTspektrum

Digitaler Wandel & Software-Architektur für Profis

Metamorphose der Dinosaurier



Metamorphose der Dinosaurier

EasiRun

Step-by-Step

den Mainframe anpacken, externes Wissen einfließen lassen, technologische Werkzeuge nutzen, damit einem die Puste nicht ausgeht.

- Komplexität aufbrechen
- Ressourcen schonen
- Risiken minimieren

G 6540F

Mainframe-Modernisierung – Best Practice im Überblick

Metamorphose der Dinosaurier

Viele Unternehmen, die heute noch mainframebasierte Kernanwendungen betreiben, sehen sich einem zunehmenden Modernisierungsdruck ausgesetzt. Nicht handeln ist keine Option, da die Anforderungen an die IT marktgetrieben stetig wachsen. Die Modernisierung mainframezentrierter Anwendungssysteme ist eine komplexe Aufgabe. Nur die systematische Herangehensweise und eine kluge Modernisierungsstrategie ermöglichen es, diese Herausforderung erfolgversprechend anzugehen. Der Artikel gibt einen Überblick.

Ausgangssituation

Wer heute im Geschäft erfolgreich sein will, muss in der Lage sein, schnell und flexibel auf wechselnde Marktbedingungen zu reagieren und seine Kostenstrukturen wettbewerbsfähig zu halten. Bezogen auf die Business-IT und die darunter liegende Infrastruktur bedeutet dies, Kundenerwartungen zu erfüllen, neue Anforderungen und Innovationen schnell an den Markt zu bringen, eine effiziente Entwicklung zu betreiben, die Kosten für Infrastruktur und Betrieb zu optimieren und dabei Sicherheit, Compliance sowie die Einhaltung aller gesetzlichen Rahmenbedingungen sicherzustellen. Welche Bedeutung eine gut aufgestellte IT hat, zeigte nicht zuletzt die Corona-Krise, in der viele Unternehmen vor der Aufgabe standen, kurzfristig eine große Anzahl Mitarbeiter gleichzeitig in die Lage zu versetzen, im Homeoffice zu arbeiten.

Die Herausforderungen betreffen jedoch nicht nur Unternehmen, sondern sind auf alle Organisationen übertragbar, deren Leistungen im Wesentlichen von einer funktionierenden IT abhängen.

Dem beschriebenen Idealbild steht aber allzu oft eine Realität gegenüber, in der mainframebasierte Anwendungen das Rückgrat der Datenverarbeitung darstellen oder aber zumindest bedeutende Anteile. Dies mag aufgrund einer Reihe von Vorteilen der Mainframes verständlich sein, doch trotzdem sehen sich gerade Betreiber solcher Infrastrukturen einem erhöhten Modernisierungsdruck ausgesetzt. Neben dem eher subjektiven Gefühl, dass Mainframes ein Relikt der 60-er bis 80-er Jahre des letzten Jahrhunderts sind, gibt es jedoch auch eine Reihe objektiver Gründe, eine Modernisierung oder gar Ablösung dieser Systeme ins Auge zu fassen:

- hohe Betriebskosten,
- schlechtere Verfügbarkeit von Fachpersonal,
- erschwerte Wartung und Erweiterung durch monolithische Anwendungsstrukturen,
- nicht mehr zeitgemäße Programmiersprachen und Datenhaltungssysteme,
- begrenzte Skalierbarkeit und
- begrenzte Interoperabilität mit modernen IT-Systemen.

Spätestens, wenn sich diese Schwachstellen in einer Art und Weise auswirken, dass Unternehmen und Organisationen nicht mehr in der Lage sind, die Geschäftsprozesse flexibel und kostengünstig zu unterstützen, ist Nichthandeln keine Option mehr.

Modernisierungsstrategie

Aus den Erfahrungen des Autors heraus hat sich die Strategie der kleinen Schritte und der Gedanke der Modernisierung als permanente Daueraufgabe als die erfolgversprechendste Herangehensweise an die Mainframe-Modernisierung herauskristallisiert.

Methodischer Ansatz

Die Strategie der kleinen Schritte folgt dem Kerngedanken, dass Modernisie-

rungsmaßnahmen, die unabhängig voneinander durchgeführt werden können, auch unabhängig in separaten Projekten umgesetzt und die Ergebnisse unmittelbar in die Produktion überführt werden. Dies schließt große Einzelprojekte nicht grundsätzlich aus, sofern diese aufgrund schwer auflösbarer Abhängigkeiten nicht sinnvoll in kleinere Projekte zerlegt werden können.

Modernisierungsziele
Ist die grundsätzliche Entscheidung gefallen, mainframebasierte Anwendungen zu modernisieren, stellt sich auch unmittelbar die Frage nach den Zielen einer Modernisierung. Oft stehen hier sehr schnell technische Aspekte wie Architekturfragen und Technologie-Stack im Fokus. Man ist jedoch sehr gut beraten, im Vorfeld eine angepasste Form des Requirement Engineering zu betreiben, um klarzustellen, über welche Eigenschaften die modernisierte Anwendung am Ende verfügen soll. Die Ergebnisse dieses Schritts sind maßgeblich für die Modernisierungsstrategie und den Technologie-Stack, der sich pragmatisch an den tatsächlichen Notwendigkeiten und nicht an den neuesten Hypes orientiert.

Grundfragen der Modernisierung
Als Hilfestellung zur Erarbeitung einer Modernisierungsstrategie ist in **Tabelle 1** eine Reihe von Fragen zusammengestellt, welche 6 Themenbereichen zuzuordnen sind.

Modernisierungsalternativen
Im Zuge der Erarbeitung der Modernisierungsstrategie empfehlen wir, die gesamte Anwendungslandschaft zu betrachten und eine Klassifizierung aller Anwendungen, Teilanwendungen und Subsysteme nach Modernisierungsalternativen vorzunehmen (**siehe Tabelle 2**).

Einzelaspekte der Modernisierung von Mainframe-Systemen

Im Folgenden wollen wir unsere Überlegungen auf ein praktisches Beispiel anwenden und auf Einzelaspekte der Mainframemodernisierung eingehen. Die betrachtete Anwendung ist fiktiv, spiegelt aber die Realität insgesamt gut wider.

Einführung Beispiel
Der Kern unseres Beispielsystems aus der Finanzbranche ist auf einem z/OS-Mainframe gehostet. Historisch bedingt besteht die Anwendung aus DB2, CICS-, IMS/DB, IMS/DC und Batch-Anteilen. Der

Themenbereich	Fragen
Fachliche Anforderungen und Qualität	✔ Erfüllt die bestehende Anwendung die erforderlichen Anforderungen des Unternehmens bzw. der Organisation? ✔ Welche Aussagen können über die Qualität der aktuellen Lösung getroffen werden? ✔ Falls die Anforderungen nicht abgedeckt werden, welche Möglichkeiten bestehen, um diesen gerecht zu werden, und ist eine Realisierung dieser Aspekte nach einer Modernisierung nachhaltig einfacher?
Machbarkeit	✔ Ist das Modernisierungsvorhaben realisierbar? ✔ Gibt es Teile der Anwendung, die nicht zu modernisieren sind, oder die von Komponenten abhängig sind, für die es keine adäquaten Alternativen gibt?
Aufwand	✔ Welche Aufgaben und Aufwände fallen bei einer Modernisierung an? ✔ Welche Voraussetzungen müssen hinsichtlich der Erbringung von Aufwänden erfüllt werden? ✔ Welche Kosten entstehen? ✔ Werden andere Vorhaben durch die Erbringung der Aufwände beeinträchtigt? ✔ Sind Ressourcen für eine Modernisierung verfügbar?
Perspektiven	✔ Welche Perspektiven ergeben sich nach einer Modernisierung für das Unternehmen, die optimierte Anwendung sowie für die Entwickler? ✔ Welche weiteren Vorteile bringt eine Modernisierung mit sich, die nicht direkt an den ROI geknüpft ist, z. B. für Entwickler, Werkzeuge, Softwarekomponenten usw.? ✔ Welche Einsparungen bringt eine Modernisierung mit sich und können diese die anfallenden Aufwände abdecken?
Risiken	✔ Welche Risiken bestehen in der Durchführung der Modernisierung? ✔ Welche Risiken bestehen, wenn die Modernisierung nicht durchgeführt wird?
Alternativen	✔ Welche Alternativen gibt es zu einer Modernisierung der Legacy-Anwendung und welche Kosten können dabei entstehen?

Tabelle 1: Fragestellungen zur Modernisierungsstrategie

Modernisierungsalternative	Erläuterung
Retire	Komplette Entfernung von Applikationen oder Teilen davon aus dem Anwendungs-Stack, da diese in Zukunft nicht mehr benötigt werden. Dies verringert den Wartungsaufwand und erhöht die Übersichtlichkeit.
Redesign	Implementierung einer neuen Gesamtarchitektur unter Verwendung der bisher bestehenden Anwendungen, Teilanwendungen und Komponenten, die zur Integration ggf. geringfügig angepasst werden müssen.
Refit	Überarbeitung des Technologie-Stacks. Dies kann sowohl die Betriebsplattform als auch die verwendeten Programmiersprachen, Datenhaltungssysteme und peripheren Systeme, wie z. B. das Druck- bzw. Outputmanagement, Versionskontrolle usw. betreffen. Anstelle des Begriffs „Refit“ werden häufig die Begriffe Migration und Rehosting genutzt, wobei das Rehosting explizit den Wechsel der Betriebsplattform weg vom Mainframe beinhaltet, während eine Migration auch ohne Wechsel der Betriebsplattform erfolgen kann. Auch der Begriff Transformation fällt sehr oft im Zusammenhang mit den Themen „Refit“, „Migration“ und „Rehosting“ und bezeichnet den Prozess, Anwendungen und Daten in den Ziel-Technologie-Stack zu überführen.
Rewrite	Komplette Neuimplementierung der bestehenden Funktionalitäten auf Basis eines modernen Software- und Infrastruktur-Stacks. Erfahrungsgemäß werden hier häufig die Aufwände sehr stark unterschätzt. Für kleine und überschaubare Funktionalitäten kann „Rewrite“ jedoch eine gute Option sein.
Write	Neuimplementierung ggf. noch fehlender Funktionalitäten auf Basis des bestehenden oder eines neuen, modernen Technologie-Stacks. Dabei ist zu beachten, dass bei Auswahl des Technologie-Stacks eine Konsistenz im Kontext der Gesamtmodernisierung gewahrt bleibt.
Replace	Ersetzung vorhandener Funktionalitäten durch Standardsoftware. Ein wichtiger Vorteil ist hier die unmittelbare Verfügbarkeit der benötigten Funktionalitäten. Test und Wartung der Software liegen beim Anbieter und werden mit Lizenzen und Wartungsverträgen erworben. In der Praxis wird jedoch häufig der Adaptierungs- und Integrationsaufwand unterschätzt, was den kalkulierten Kosten- und Aufwandsvorteil gegenüber Alternativen signifikant schrumpfen oder gar entfallen lässt.

Tabelle 2: Modernisierungsalternativen

Kern der Geschäftslogik ist in COBOL implementiert, während Teile des Rechenkerns mit PL/I umgesetzt wurden. Weitere Anwendungsteile liegen in Assembler vor. Das System kommuniziert über eine Reihe von synchronen und asynchronen Schnittstellen mit SAP-, reinen Java- sowie weiteren mainframebasierten Anwendungen. Fachliche Tests werden auf Basis einer vorhandenen Dokumentation manuell durchgeführt. Eine vollständige Abdeckung ist dabei allerdings nicht gewährleistet.

Als langfristige Ziele der Modernisierung wurden festgelegt:

- Ablösung des Mainframes durch Linux-Serversysteme,
- open-source-orientierter Technologie-Stack,
- signifikante Reduzierung der Betriebskosten,
- durchgängiger Einsatz zeitgemäßer Technologien wie Java und Web-Frontends,
- Beibehaltung der hohen Performanz und Verfügbarkeit gemessen an den Mainframe-Systemen.

Assessment

Die Umsetzung des Vorhabens beginnt mit einer tief greifenden Bestandsaufnahme in Form eines Assessments. Die Hinzuziehung externer Expertise ist hierbei unerlässlich, da die Praxis zeigt, dass das Wissen um die eigene Anwendung oft eher subjektiv beziehungsweise selektiv ist. Zudem verfügen externe Spezialisten über Know-how und geeignete Werkzeuge, die organisationsintern in der Regel nicht vorhanden sind.

Die Basis des Assessments ist immer eine maschinelle Analyse sämtlicher Quellcodes und bestimmter Systemkonfigurationsdateien. Ergänzt wird die maschinelle Analyse durch Experteninterviews und Auswertung der vorhandenen Anwendungsdokumentation. Kernziele des Assessments sind:

- qualitative und quantitative Bewertung des Modernisierungsvorhabens (Wie viele Komponenten welchen Typs sind von der Modernisierung betroffen, wie komplex wird sich die Modernisierung der verschiedenen Komponententypen gestalten?),
- klare Abgrenzung des zu modernisierenden Systems (Welche Komponenten sind Bestandteil des Modernisierungsumfangs? Welche Schnittstellen zu anderen Systemen bestehen?).

In einer weiter gefassten Aufgabenstellung können aber auch weitere Themenbereiche betrachtet werden:

- Aufzeigen von Modernisierungsoptionen,
- Entwicklung einer Modernisierungsstrategie,
- erste Aufwandsindikation in Bezug auf Ressourcen, Projektlaufzeiten und Kosten.

Die im Assessment erarbeiteten Ergebnisse bilden die Grundlage aller weiteren Entscheidungen und Planungen im Modernisierungsvorhaben.

Modernisierungsansatz

Dem methodischen Ansatz der kleinen Schritte folgend werden für unser Beispiel 4 Hauptphasen der Modernisierung definiert (**siehe Abbildung 1**). Innerhalb dieser werden anschließend die unabhängig voneinander umsetzbaren Teilprojekte identifiziert.

Phase 1: Konsolidierung

Am Beginn des Modernisierungswegs steht eine Konsolidierungsphase, welche Komplexität reduzieren sowie Migrationshemmnisse und -hindernisse beseitigen soll. Für das Beispiel wurde eine Reihe von unabhängigen Einzelprojekten identifiziert:

Testunterstützung: Im Rahmen sämtlicher Modernisierungsschritte kommt dem Testen eine herausragende Bedeutung zu. Daher ist man gut beraten, dieses Thema von Anfang an in seine Überlegungen aufzunehmen. Es bietet ein hohes Potenzial zur Optimierung der folgenden Modernisierungsmaßnahmen und zur Risikominimierung. In unserem konkreten Beispiel wird ein Ausbau der Testunterstützung festgelegt, welcher folgende Punkte umfasst:

- Systematisierung der Vorgehensweise,
- Komplettierung von Testszenarien,
- Automatisierung des Testvorgehens (automatisierte Bereitstellung von Testdaten für Einzelszenarien, automatisierte Datenvergleiche, Testjobs und Record-/Replay-Mechanismen).

IMS/DB: Für IMS/DB steht auf Zielplattformen abseits des Mainframes keine Produktalternative zur Verfügung. Ein üblicher Weg, die gewünschten Funktionalitäten trotzdem bereitzustellen, besteht in der Umstellung der IMS/DB-Datenhaltung auf eine relationale Datenhaltung inklusive einer Zugriffsschicht, welche in Richtung Anwendung über die bestehenden Schnittstellen die IMS/DB-Funktionalitäten bereitstellt. Diese Maßnahme ist mit Großrechnertechnologie realisierbar und wird daher als Einzelprojekt im Rahmen der Konsolidierung vorgesehen.

IMS/DC: Zur Vereinheitlichung des Technologie-Stacks ist die Reduzierung auf einen Transaktionsmonitor vorgesehen. Da die Alternativen zu CICS auf dem Linux-Zielsystem zahlreicher sind, wurde als weiteres Konsolidierungsprojekt die Umstellung von IMS/DC nach CICS auf dem Mainframe identifiziert.

Assembler: Im Rahmen des Assessments wurden drei Typen von Assemblercode identifiziert. Es handelt sich um mainframespezifischen, rein technischen Code, Zugriffsmodule für IMS/DB sowie Geschäftslogik.

Der rein mainframespezifische technische Code wird im Rahmen einer möglichen Konsolidierung nicht betrachtet. Dies wird erst in einer späteren Phase geschehen.

Die Zugriffsmodule werden bereits im Projekt zur Umstellung von IMS/DB als Bestandteil der Zugriffsschicht in einer anderen Zielsprache realisiert.

Die Assembler-Geschäftslogik wird im Rahmen eines Konsolidierungsprojekts (**siehe Abbildung 2**) nach COBOL umgestellt.

PL/I: Für die Umstellung von PL/I existiert eine Reihe von Alternativen:

- Nutzung eine PL/I-Compilers auf dem Zielsystem,
- Umstellung des PL/I-Codes nach COBOL,
- Umstellung des PL/I-Codes nach Java.

Da für unseren fiktiven Kunden die Nutzung eines PL/I-Compilers auf dem Zielsystem nicht infrage kam, blieb nur die Frage, in welche Zielsprache die Umsetzung erfolgen soll. Hier fiel die Wahl auf COBOL, da diese Sprache zumindest kurz- und mittelfristig durch den Kunden entwicklungsseitig besser unterstützt werden kann.

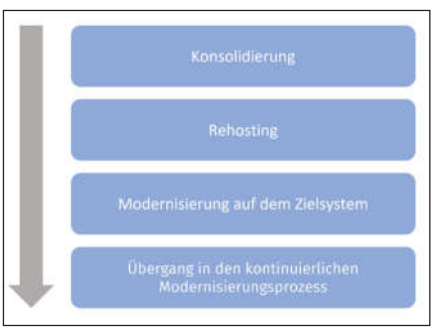


Abb. 1: Hauptphasen der Modernisierung

Phase 2: Rehosting

In der Phase des Rehostings findet nun der Umzug in ein Linux-Umfeld statt. Prinzipiell wäre es zwar möglich, auch diesen Schritt in mehreren Stufen zu vollziehen, die Praxis zeigt jedoch, dass die damit verbundenen Zusatzaufwände für die Sicherstellung eines zuverlässigen Hybrid-Betriebs in keinem Verhältnis zum Nutzen stehen. Deshalb wird das Rehosting in einem Gesamtschritt vollzogen, der durch eine Reihe von Teilprojekten umfassend vorbereitet und integrativ getestet wird. Trotzdem wird auch in dieser Phase angestrebt, die jeweils kleinstmöglichen und risikoärmsten Schritte zu gehen. Dies spiegelt sich in der Wahl der verwendeten Technologien wider.

COBOL: Für das Rehosting der COBOL-Anwendung (**siehe Abbildung 3**) wird eine marktgängige COBOL-Java-Transpiler-Lösung verwendet. Diese überführt den COBOL-Quellcode nach jeder Änderung beziehungsweise Anpassung zunächst in reinen Java-Quellcode, welcher durch einen Java-Compiler schließlich in Java-Klassen gewandelt wird. Damit ist betriebsseitig bereits in dieser Phase ein kompletter Übergang in die Java-Welt erfolgt, während entwicklungsseitig noch COBOL als Entwicklungssprache dient. Dies hat den Vorteil, die ohnehin hohen Schulungsaufwände in dieser Phase etwas zu reduzieren.

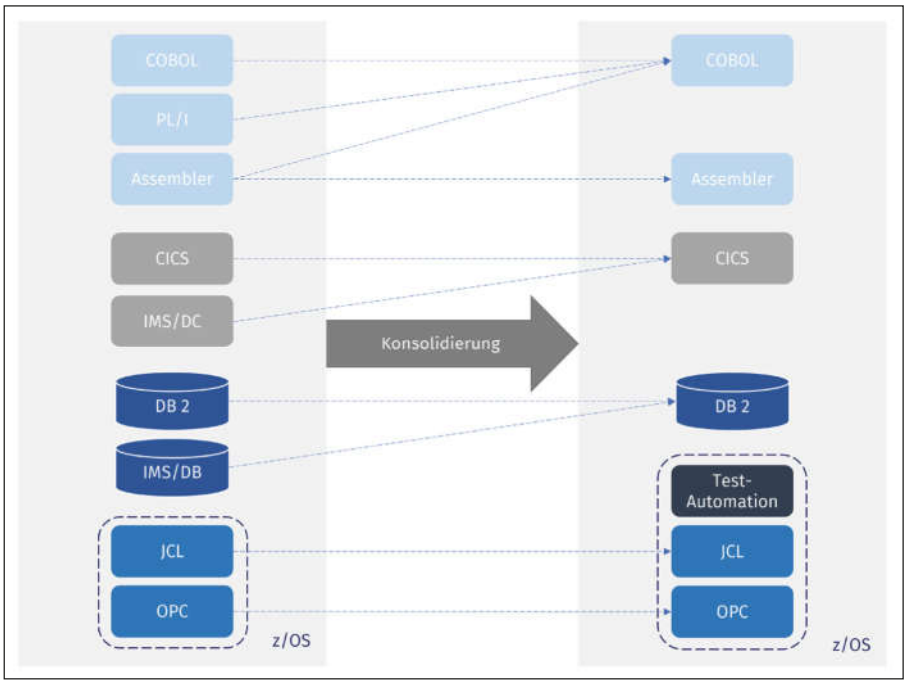


Abb. 2: Konsolidierungsphase am Beispiel

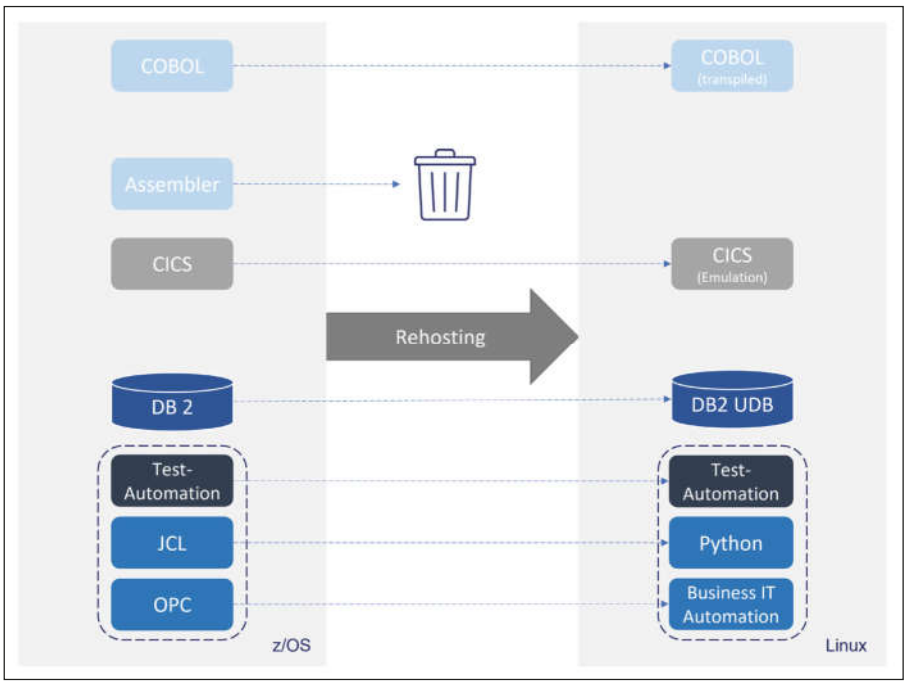


Abb. 3: Rehosting-Phase am Beispiel

JCL und Scheduling: Obwohl Emulationslösungen für den Bereich Job Control verfügbar sind, geht der Trend im Bereich JCL (Job Control Language) eher in Richtung der Transformation des JCL-Quellcodes in eine Skriptsprache, wie Bash, Perl oder Python. Als Ablösung für den Scheduler steht in der Linux-Welt ebenfalls eine Reihe von Alternativen zur Verfügung, von einfachen Schedulern bis hin zu komplexen Business-IT-Automation-Lösungen.

CICS: Der Transaktionsmonitor wird mit einer marktgängigen CICS-Emulation abgelöst.

Datenbank: Um Aufwand und Risiken in der Rehosting-Phase klein zu halten, wird in unserem Beispiel für die Ablösung der DB2-Datenbank auf das Linux-Pedant DB2 UDB gesetzt, da dies eine weitgehende Kompatibilität in Bezug auf Funktionsumfang und SQL-Syntax gewährleistet. Die Migration der Daten kann in diesem Fall recht einfach mit den DB2 und DB2 UDB-Bordmitteln realisiert werden.

Testsystem: Aufbauend auf den Mechanismen des Testsystems wird eine Lösung für den Paralleltest unter Betriebsbedingungen erstellt.

Betriebsübergangs- und Betriebskonzept: Neben den rein technischen Tätigkeiten sind auch konzeptuelle Arbeiten erforderlich. Diese umfassen das Betriebsübergangs- und das Betriebskonzept für die rehostete Anwendung.

Produktivsetzung: Die Produktivsetzung der modernisierten Anwendung folgt dem Betriebsübergangskonzept. Sie wird mehrmals im Vorfeld geprobt und besteht in der Regel aus einer finalen Datenmigration, finalen Tests und der Anpassung der Systemkonfiguration in Bezug auf externe Schnittstellen. Erfahrungsgemäß treten hier bei den ersten Proben noch eine Reihe von Fehlern und Problemen zutage, welche in einer Optimierung des Betriebsübergangskonzepts münden. Nicht selten ist es in der Praxis der Fall, dass am Ende die erfolgreiche Generalprobe direkt zur Produktivsetzung erklärt wird.

Phase 3: Weitere Modernisierung auf dem Zielsystem

Mit dem erfolgreichen Abschluss des Rehostings sind die Voraussetzungen für eine weitere Modernisierung der Anwendung auf dem Zielsystem geschaffen. In dieser Phase kann wieder eine Reihe klei-

nerer, unabhängiger Modernisierungsschritte identifiziert werden (**siehe Abbildung 4**).

Migration des Datenbanksystems: Dem Modernisierungsziel eines Open-Source-Technologie-Stacks folgend wird die Datenhaltung auf eine Postgres-Datenbank migriert. Neben der Transformation der Datenstrukturen und Daten sind dazu auch einige Änderungen im Programmcode selbst erforderlich. Diese Änderungen beziehen sich vor allem auf die SQL-Syntax und die Verarbeitung von Returncodes des Datenbanksystems.

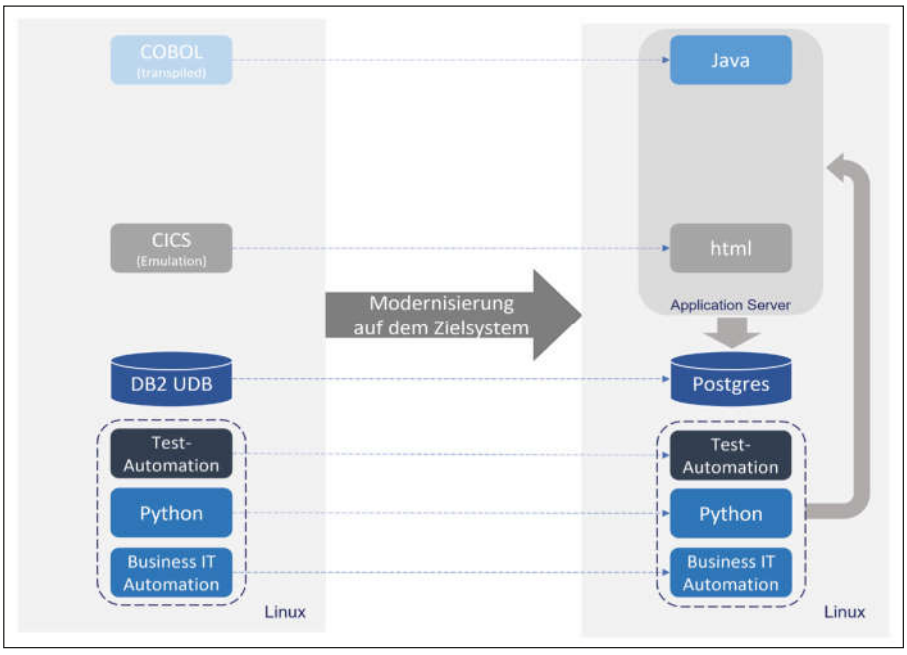


Abb. 4: Modernisierung auf dem Zielsystem (Beispiel)

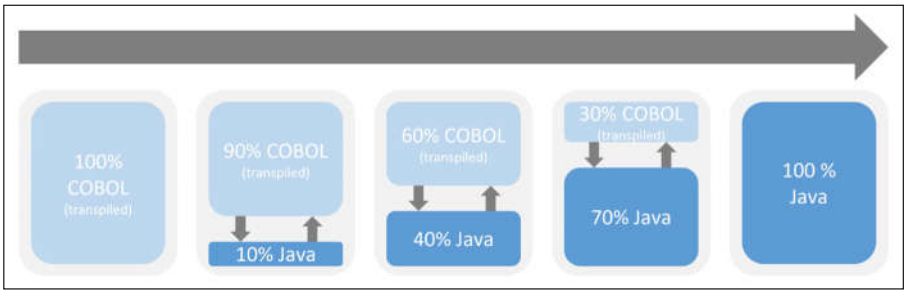


Abb. 5: Sukzessiver Übergang nach Java auf Basis einer Transpiler-Lösung

Übergang nach Java: Für den Übergang nach Java stehen prinzipiell zwei Ansätze zur Auswahl, die automatisierte Transformation des COBOL-Codes nach Java oder die sukzessive manuelle Umsetzung der Anwendung. Für unser Projekt entscheiden wir uns für die sukzessive manuelle Umsetzung (**siehe Abbildung 5**). Ausschlaggebend ist hierfür die Tatsache, dass eine reine optimierte Java-Anwendung das Ziel der Modernisierung ist. Zudem ermöglicht diese Vorgehensweise weitgehende Freiheiten in Bezug auf die Gestaltung der Zielarchitektur. Diese Vorzüge kann uns eine automatisierte Transformationslösung nicht bieten. Mit der Technologieauswahl in Bezug auf COBOL im Rahmen des Rehostings wurde auf eine Lösung zurückgegriffen, die diesen Ansatz optimal unterstützt.

Modernisierung des Transaktionsmonitors: Die Ablösung des CICS-orientierten Transaktionsmonitors durch eine moderne Lösung umfasst sowohl das Transaktionsmanagement als auch die Benutzeroberflächen. Da die Datenhaltung insgesamt auf eine Datenbank konsolidiert wurde, kann das Transaktionsmanagement in der Zukunft komplett über die Datenbankverbindung erfolgen. Dies kann über eine selbst entwickelte Minimallösung oder aber unter Nutzung eines Applikationsservers geschehen. Die Oberflächen selbst werden mit einem zeitgemäßen Web-Framework neu implementiert. Der gesamte Modernisierungspfad bis zum Ende der dritten Phase ist noch einmal in **Abbildung 6** zusammengefasst.

Phase 4: Übergang in den kontinuierlichen Modernisierungsprozess

Mit dem Abschluss der Phase 3 sind die gesetzten Ziele der Modernisierung er-

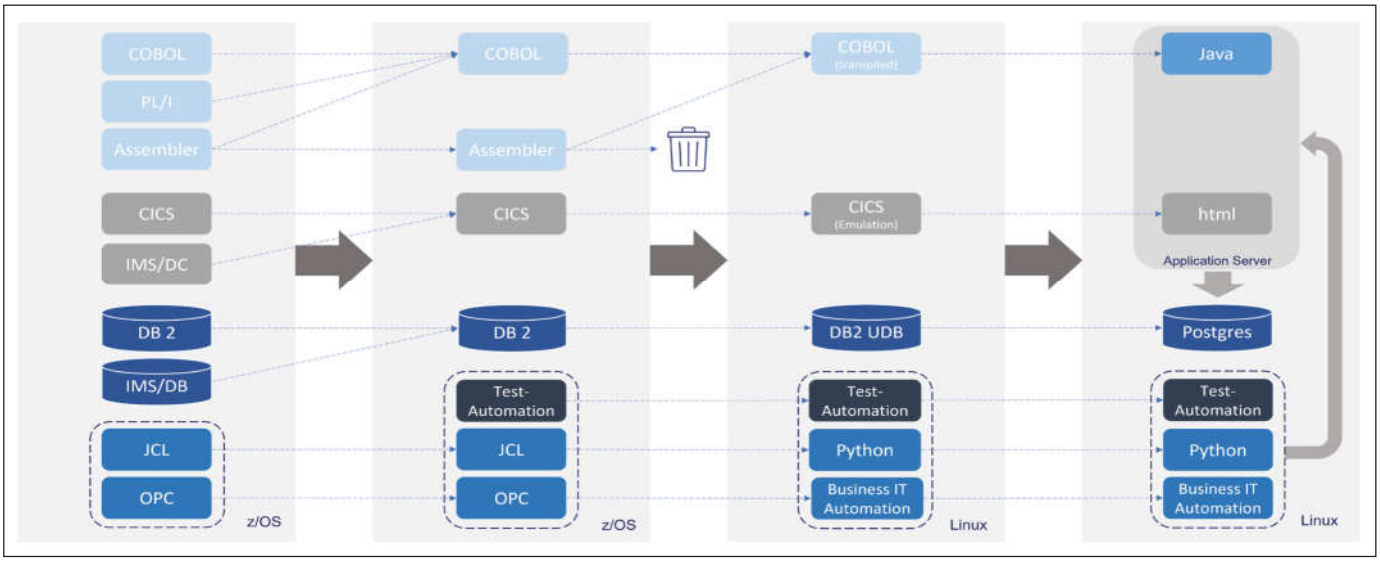


Abb. 6: Kompletter Modernisierungspfad unseres Beispiels

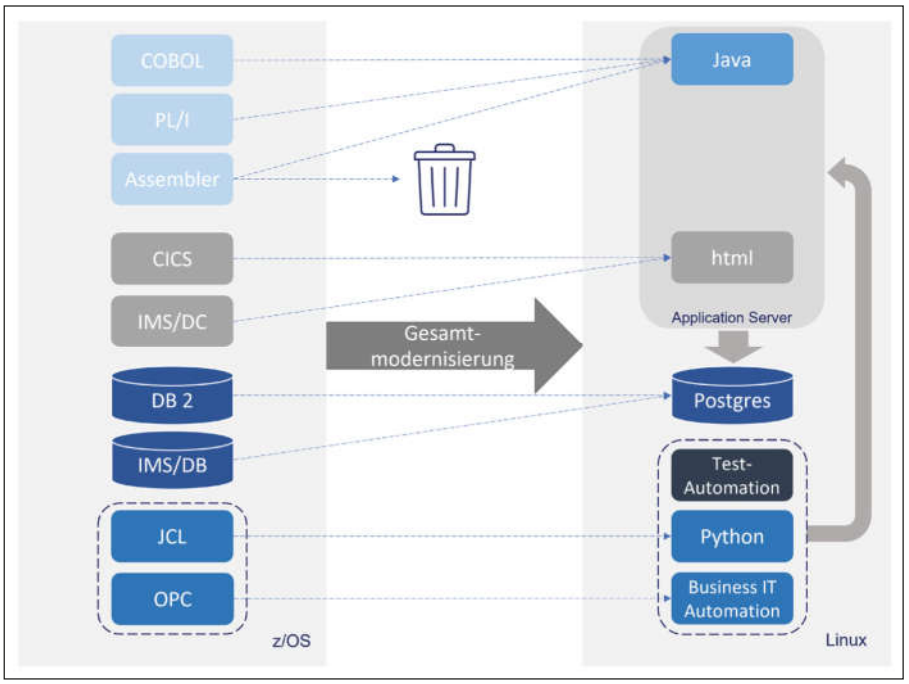


Abb. 7: Zusammenfassender Blick auf die Gesamtmodernisierung (Beispiel)

reicht. In der letzten Phase geht es nun darum, den Modernisierungsprozess zu verstetigen und den Aufbau neuer technischer Schulden zu vermeiden. Dies erfordert ein Umdenken, welches die Modernisierung des Systems als ebenso bedeutende permanente Pflichtaufgabe betrachtet wie die Umsetzung von Change Requests der Fachabteilungen.

Die Praxis lehrt, dass dieses Vorhaben in der Praxis oft schwerer zu verwirklichen ist, als es klingt. Hilfreich kann es hier sein, die Absicht durch organisatorische Maßnahmen zu unterstreichen, wie zum Beispiel die permanente Schaffung einer eigenen Einheit innerhalb der Entwicklungsabteilung, welche sich ausschließlich mit dem Thema kontinuierliche Modernisierung befasst. Ihre Aufgaben liegen in der permanenten Analyse des Systems, Marktbeobachtungen und Recherche, Erarbeitung von Modernisierungsvorschlägen sowie Projektleitung für Modernisierungsmaßnahmen. In Anbetracht häufig mangelnder Ressourcen mag diese Maßnahme eine Herausforderung darstellen. Diese zu lösen, wird sich aber mit Sicherheit langfristig auszahlen.

Modernisierungsunterstützung

Obwohl wir mit der oben beschriebenen Strategie bereits einen ersten Schritt zum Aufbrechen der Komplexität der Mainframe-Modernisierung getan haben, sind auch die Herausforderungen der Einzelprojekte nicht zu unterschätzen. An der Hinzuziehung externer Expertise und Nutzung bereits vorhandener technischer Lösungen für die Modernisierungsunterstützung führt daher kein Weg vorbei. Zum Abschluss wollen wir einen kleinen Überblick über technische Möglichkeiten der Modernisierungsunterstützung geben.

Analysewerkzeuge

Analysewerkzeuge ermöglichen es, Informationen zu gewinnen, die eine Beurteilung von Qualität und Quantität der Anwendung ermöglichen, aber auch Unterstützung bei der Lösung technischer Einzelaufgaben.

Hierbei unterscheiden wir zwischen statischer Analyse, die ihre Informationen ausschließlich aus dem Quellcode und gegebenenfalls anderen statischen Datenquellen bezieht, und der dynamischen Analyse, die auf Basis instrumentierten Codes auswertbare Daten zur Laufzeit gewinnt. Im Kontext der Modernisierung findet die statische Analyse Anwendung insbesondere im Bereich des Assessments auf technischer Ebene und der Validierung von Modernisierungsmaßnahmen, während die dynamische Analyse die Zusammenführung von technischer und fachlicher Sicht auf die Anwendungen unterstützt. Dies hat Relevanz zum Beispiel beim Schneiden von Services und dem Übergang in ein Domain-Driven Design.

Transformationstechnologien

In einer Reihe von Modernisierungsschritten haben wir bereits auf die Transformation von Quellcode verwiesen. Hierfür existieren auf dem Markt bereits eine ganze Reihe automatisierter Lösungen, die in der Regel auch kundenspezifisch adaptiert und optimiert werden können wie:

- Assembler nach COBOL,
- COBOL nach COBOL (Codeanpassungen zur Integration in neue Betriebsumgebungen),
- JCL nach Bash, Perl, Python usw.,
- COBOL nach Java,
- PL/I nach COBOL oder Java,
- ADABAS/NATURAL nach COBOL oder Java und relationalem Datenbanksystem.

Zusätzlich existiert eine Reihe von Lösungen, für den Bereich der Datenmigration, zum Beispiel:

- strukturbasierte Datentransformation für Dateien (z. B. EBCDIC nach ASCII),
- Umstellung VSAM/ISAM-Datenhaltung auf relationale Datenbanksysteme (automatisierte Migration von Datenstrukturen, Daten und Programmcode).

Fazit

Die Modernisierung mainframebasierter Anwendungssysteme ist eine komplexe Herausforderung und erfordert einen langen Atem. Durch eine Strategie der kleinen Schritte kann die Komplexität aufgebrochen werden. Die Einbindung externer Partner mit entsprechendem Know-how sowie der Einsatz bereits vorhandener Werkzeuge und Modernisierungslösungen schont die eigenen Ressourcen und minimiert die Risiken. **Abbildung 7** fasst das Modernisierungsbeispiel noch einmal zusammen. ||

Der Autor

Dipl.-Ing. Udo Thiele
(uthiele@easirun.de)
ist Entwicklungsleiter der EasiRun Europa GmbH. Sein Arbeitsfeld liegt seit mehr als 15 Jahren in den Bereichen Modernisierungsstrategie sowie Entwicklung von Werkzeugen und Lösungen zur Codetransformation, Anwendungsanalyse und -modernisierung.