

1 Einleitung zum Thema Softwaremigration

1.1 Die Motivation für Softwaremigration

Das Thema »Softwaremigration« ist ein Evergreen. Es ist immer aktuell, unabhängig davon, welche Softwaretechnologie gerade vorherrscht. Sicher ist: Auch die aktuelle Technologie wird nicht von langer Dauer sein. Wer schon länger mit der IT zu tun hat, wird wissen, dass Migration ein wiederholtes Ereignis bei jedem Anwender ist. Es tritt immer dann auf, wenn Hardware oder Software ausgetauscht werden, und dies ist recht häufig. Wenn sich in der IT-Umgebung etwas ändert, zieht dies andere Änderungen nach sich. Dieser Änderungsprozess findet in immer kürzeren Intervallen statt. Für den, der mit der IT weniger vertraut ist, mag es zunächst seltsam erscheinen, dass diese Welt so sehr mit sich selbst beschäftigt ist. Es werden Unsummen von Geld investiert, um den IT-Betrieb auf dem neuesten Stand zu halten, oft mit fragwürdigem Nutzen für die Benutzer. Die Antwort auf die Frage, warum dies so ist, hat mit der instabilen Natur dieser jungen Industrie zu tun.

Die IT-Industrie ist ständig auf der Suche nach sich selbst. Sie ist eine Kombination mehrerer Technologien – der Computer-, der Kommunikations-, der Speichertechnologie und nicht zuletzt der Softwaretechnologie. Wenn sich nur eine dieser Technologien ändert, sind die anderen mit betroffen. Wir wissen von Moore's Law, dass die Kapazität der Rechner sich alle 18 Monate verdoppelt, und dies seit Jahren [Glas99]. Die Kommunikationstechnologie verhält sich ähnlich. Seit der Verbreitung der Internetnutzung hat sich die Netzwerkkapazität um das Hundertfache vermehrt. Die Massenspeichertechnologie hat ihren eigenen Lebenszyklus und verdoppelt ihre Leistung alle zwei Jahre. Heute ist es möglich, im gleichen Speicherraum 10-mal so viele Daten zu speichern wie vor 20 Jahren. Die langsamste der IT-Technologien ist die Softwaretechnologie, die sich nur alle fünf Jahre verändert. Neue Sprachen und Entwurfparadigmen setzen sich relativ zu den »härteren« Technologien nur deshalb so langsam durch, weil die Anwender sie erst erlernen müssen, und das dauert länger. Der Mensch bremst also die Softwareinnovation.

Die Anwendersysteme stützen sich auf die IT-Technologien und veralten in dem Maße, wie sie hinter die Möglichkeiten dieser Technologien zurückfallen. Sie werden dadurch im Laufe der Zeit mit dem Stand der anderen Technologien inkompatibel. Die Basissoftwarehersteller müssen immer mehr Aufwand betreiben, um die Rückwärtskompatibilität zu früheren Technologien aufrechtzuerhalten. Deswegen sind sie bemüht, die Anwender zur Übernahme der neuen Technologien zu bewegen. Welchen Nutzen der Anwender davon hat, ist eine andere Frage. Ein Paradebeispiel dafür ist das Thema SOA (serviceorientierte Architektur). Es sind nicht die Anwender, die danach rufen; sie haben noch genug damit zu tun, die Webtechnologie zu verdauen. Es sind die Softwareanbieter, vor allem die Applikationsanbieter, die den Anwendern diese neue Technologie regelrecht aufdrängen, weil es für sie bequemer und günstiger ist, einzelne Services bereitzustellen als komplette automatisierte Anwendungen in vielfachen Variationen [Andr04].

Die IT-Industrie muss die Weiterentwicklung ihrer Technologien finanzieren, also müssen sie verkauft werden. Die Anwender werden in ihrer Rolle als Abnehmer mit allen Regeln der Werbekunst beeinflusst (wobei die Fachpresse als williges Instrument der Industrie eine nicht unerhebliche Rolle spielt) und zuletzt mit Drohungen zur Einstellung der Wartung und Erhöhung der Wartungsgebühren bewegt, die neuen Technologien anzunehmen. Wenn das geschieht, steht eine Migration vor der Tür. Wenn es nach dem Willen der IT-Industrie ginge, würden die Anwender ihr ganzes Anwendungsportfolio alle fünf Jahre komplett ersetzen. Da sich dies aber kein Anwender leisten kann, bleibt ihm – um mit der fortschreitenden Technologie Schritt zu halten – nur übrig, seine alten Anwendungssysteme in die neueste technologische Umgebung zu versetzen bzw. zu migrieren [McDo02].

Softwaremigration ist deswegen das Schicksal der IT-Anwender. Sie werden damit konfrontiert, solange sie auf IT-Technologien angewiesen sind. Nur wenn die Anwendungssysteme völlig isoliert gegenüber ihrer technischen Umgebung wären, könnten sie so bleiben, wie sie sind, auch wenn die Umgebung sich wandelt. Dies ist seit jeher ein Qualitätsziel der Informatik gewesen, hat aber wenig Unterstützung durch die Industrie gefunden. Dieses Ziel steckt hinter Begriffen wie Portabilität, Interoperabilität, Konvertibilität und Wiederverwendbarkeit, die nie richtig verwirklicht worden sind. Durch die Trennung der Benutzerinteraktion von der Geschäftslogik und dieser wiederum von der Datenhaltung, also durch eine wohldurchdachte Systemarchitektur, kam man dem Ziel näher, zumindest einen Teil der Software vor Veränderungen in der Umgebung abzuschotten. Aber auch das haben bisher noch nicht alle Anwender erreicht.

Der Großteil der IT-Anwender unterliegt dem Modezwang der IT-Industrie, Neues auszuprobieren. Hinzu kommt die Notwendigkeit, Software möglichst schnell und bequem zu entwickeln. Dazu müssen die letzten Sprachen und der letzte technische Schnickschnack verwendet werden – und prompt sitzen die

Anwender in der Technologiefalle. Ihre Software ist nur so lange betriebsfähig, bis die nächste Technologiewelle (siehe Abb. 1–1) über sie hinwegrollt [Stra98]. Spätestens dann steht die nächste Migration vor der Tür.

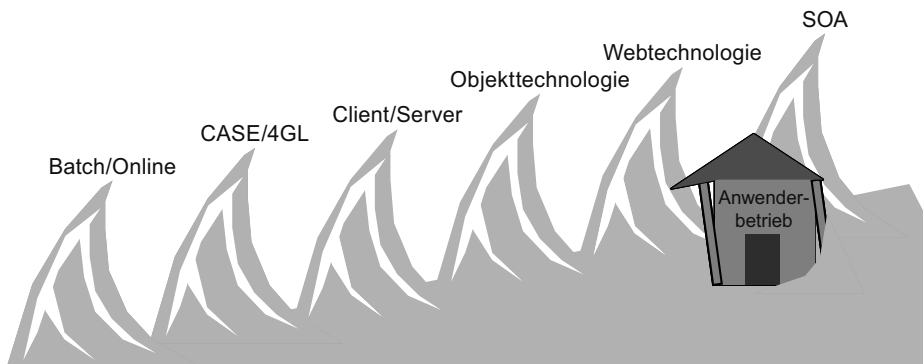


Abb. 1–1 Wellen der Veränderung

Jede Technologiewelle hinterlässt Strandgut am Ufer, für das die Softwarewelt den Begriff »Legacy-Systeme« geprägt hat; diese Systeme sind mit den Technologien der letzten oder vorletzten Welle entstanden. Da diese Technologiewellen immer schneller aufeinander folgen, häuft sich das Strandgut. Inzwischen liegt die Halbwertszeit eines Softwareprodukts bei weniger als fünf Jahren. Anwendungssoftware gehört also nach etwa fünf Jahren zu den Legacy-Systemen, ist veraltet und wird immer mehr zu einer Last. Es ist schwierig, diese alten Systeme mit den neuen zu verbinden, es ist aufwendig, sie fortzuschreiben, und es ist mühsam, sie zu korrigieren. Es ist auch nicht einfach, Personal zu finden, das sich damit befassen will. Für junge Informatiker gibt es nichts Schlimmeres, als sich mit alten Technologien herumzuplagen, nur wenige haben einen ausgesprochenen Hang zur Softwarearchäologie. Diese und andere Sorgen plagen Anwender und Benutzer alter Softwaresysteme. Sie sitzen in der sogenannten Legacy-Software-Falle [Snee02a]. Auswege aus dieser Falle sind mit hohen Kosten verbunden. Anwender können (siehe Abb. 1–2)

- das alte System beibehalten wie bisher,
- das alte System in eine moderne Umgebung migrieren,
- das alte System in einer modernen Umgebung neu entwickeln,
- das alte System durch Standardsoftware ersetzen.

Jede Alternative hat ihre Vor- und Nachteile, die in der Folge erläutert werden. Der Schwerpunkt dieses Buches liegt auf der zweiten Alternative (»das alte System in eine moderne Umgebung migrieren«), denn das versteht die Fachwelt allgemein unter Softwaremigration. Dazu wird ein generischer Prozess definiert, der auf alle Migrationsprojekte anwendbar und an sie anpassbar ist. Die Methoden,

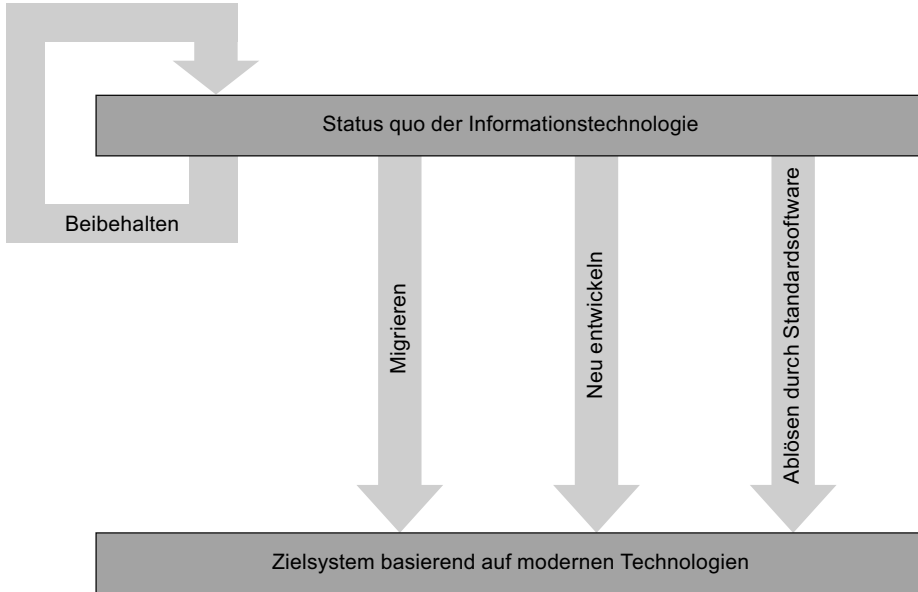


Abb. 1-2 Alternative Auswege aus der Legacy-Falle

Techniken und Werkzeuge, die zu einer solchen Migration gehören, werden beschrieben. In Kapitel 7 schaffen Fallstudien die Verbindung zur betrieblichen Migrationspraxis.

1.2 Zum Zustand der IT in der betrieblichen Praxis

Zu Beginn des 21. Jahrhunderts ist der Stand der IT in vielen Unternehmen alles andere als zufriedenstellend. Neue Technologien wurden zwar eingeführt, aber die alten Systeme blieben. Für ihre Ablösung fehlen Zeit und Geld. Da die Altsoftware sich über die Jahre akkumuliert, macht sie einen immer größeren Anteil des gesamten Softwarebestandes der Anwenderunternehmen aus. Demzufolge kann man den heutigen Status quo der IT in vielen Unternehmen als archaisch bezeichnen [SnSn03]. Kennzeichnend dafür ist eine Aussage in der Zeitschrift »IEEE Computer«, wonach die Chinesen gegenüber dem Westen im Vorteil seien, weil sie keine alten IT-Systeme zu erhalten hätten; sie könnten gleich damit beginnen, ihre Anwendungen mit den neuesten Technologien zu implementieren [Meye06].

Diese Aussagen treffen vor allem für große Unternehmen zu, die langjährige Computeranwender sind. Typische Beispiele hierfür sind Finanzdienstleister, Handelsfirmen und Behörden. Sie verharren nicht selten auf der Technologie der 70er- oder 80er-Jahre. Gleichzeitig ist ihre IT als Folge der Technologieevolution zu einer bunten, heterogenen Landschaft gewachsen, die sich aus einzelnen Insel-

lösungen zusammensetzt. Anwendungen, die in Assembler, COBOL, PL/1 oder 4GL-Sprachen implementiert sind und auf Mainframe-Systemen laufen, sind in diesem Marktsektor eher Normalfall als Ausnahme. Die Daten sind oft noch in hierarchischen, netzwerkartigen oder bestenfalls semirelationalen Datenbanken gespeichert. Die Steuerung der Programme erfolgt nicht durch benutzerfreundliche grafische Oberflächen, sondern häufig noch über text- bzw. zeichenbasierte Bildschirmmasken. Grafische Benutzungsschnittstellen werden allenfalls über Emulation der alten Bildschirmmasken auf eine grafische Benutzungsoberfläche realisiert [SnSn03].

Parallel zum Entstehen dieses Rückstaus im Anwendungsportfolio hielt ab Mitte der 90er-Jahre die große Business-Process-Reengineering-Welle weltweit Einzug in die Unternehmen. Noch heute ist BPR das Mittel der Wahl zur Erhöhung der organisatorischen Effizienz und Flexibilität. Organisatorische Umstellungen können jedoch nur so erfolgreich sein, wie es die IT-Unterstützung der daraus resultierenden Geschäftsprozesse zulässt. Beim Business Process Reengineering (BPR) hat sich die Informations- und Kommunikationstechnologie als unentbehrliches Hilfsmittel und Medium erwiesen [Lehn99]. Wenn sie jedoch nicht mitzieht, bleibt der Nutzen des Business Process Reengineering gering.

Reorganisation steht somit in unmittelbarem Zusammenhang mit technologischen Änderungen. Betrachtet man die Wechselwirkung zwischen Technologie und Organisation, lässt sich feststellen, dass fehlende technologische Anpassung oder gar IT-Stagnation in den meisten Fällen zum Scheitern von Business-Process-Reengineering-Projekten führt. Die veraltete IT wirkt als Bremse bei der Einführung neuer Geschäftsprozesse und damit bei der Weiterentwicklung der Unternehmen.

Daneben fordert auch das Internetzeitalter seinen Tribut. Angesichts des gegenwärtigen Trends zum E-Business und damit zur webbasierten Informationstechnologie gelten selbst heute noch modern erscheinende Systeme bereits als Legacy. Das bedeutet, dass nicht nur Code-Oldies, die auf veralteter Hard- und/oder Systemsoftware laufen, als Legacy-Systeme bezeichnet werden können, sondern alle Systeme, die auf einer früheren Technologie basieren als der gegenwärtig aktuellen [SnSn03]. Selbst die Client/Server-Systeme der 90er-Jahre können demzufolge bereits als überholt bezeichnet werden [Snee02b]. Um auf den Zug der Zukunft aufspringen zu können, muss die IT-Landschaft eines Unternehmens in ständiger Bewegung bleiben.

Obwohl der Einsatz bestehender Legacy-Systeme mit beträchtlichen Problemen verbunden ist, sind viele Unternehmen nicht dazu bereit, die notwendigen technischen Hürden einer State-of-the-Art-Anpassung zu überwinden. Dies ist insbesondere und vor allem dann der Fall, wenn diese Systeme noch immer erfolgreich zur Aufrechterhaltung des laufenden Geschäftsbetriebs beitragen. Warum sollten die Anwender sich die Mühe machen, ihre doch noch funktionstüchtigen Systeme auf neue Technologien zu trimmen? Zur Beantwortung dieser Frage ist es

zunächst sinnvoll, sich die Auswirkungen, die mit dem Einsatz eines Legacy-Systems verbunden sind, vor Augen zu führen und darüber hinaus die möglichen Alternativen des Anwenders für einen Technologiewechsel zu betrachten.

Der Einsatz von Computertechnologien über Jahre, ohne sich den Herausforderungen des technologischen Wandels zu stellen, hat in den meisten Fällen zu großen, komplexen Gebilden geführt, die nur noch mit unverhältnismäßig großem Aufwand zu erhalten und fortzuschreiben sind. Mittlerweile werden über 60 Prozent des IT-Budgets langjähriger Anwenderunternehmen (also jener, die eigene IT-Systeme seit mehr als zehn Jahren betreiben) dazu verwendet, die bestehenden Systeme im Betrieb zu halten [KaZh05]. Dies spiegelt sich in der häufig zitierten Wartungskrise wider, die in [SePILe03] als Legacy Crisis bezeichnet wird. Danach übersteigt der Anteil an den Softwarekosten, der für die Erhaltung und Weiterentwicklung anfällt, den der Neuentwicklung deutlich.

In Anbetracht solcher Erblasten und aufgrund neuer rechtlich oder geschäftlich bedingter Anforderungen wie Basel II, die tiefe Eingriffe in die Software erfordern, werden Anwender notwendigerweise zum Umdenken gezwungen. Die Kluft zwischen der Leistungsfähigkeit bestehender Systeme und den an sie gestellten Anforderungen wird zunehmend größer und kann sich direkt auf die Existenzfähigkeit der Unternehmen auswirken. Gefahr droht von jenen Konkurrenzunternehmen, denen es gelingt, sich von ihren Altlasten zu befreien und die Vorteile neuer Technologien zu nutzen (siehe Abb. 1–3).

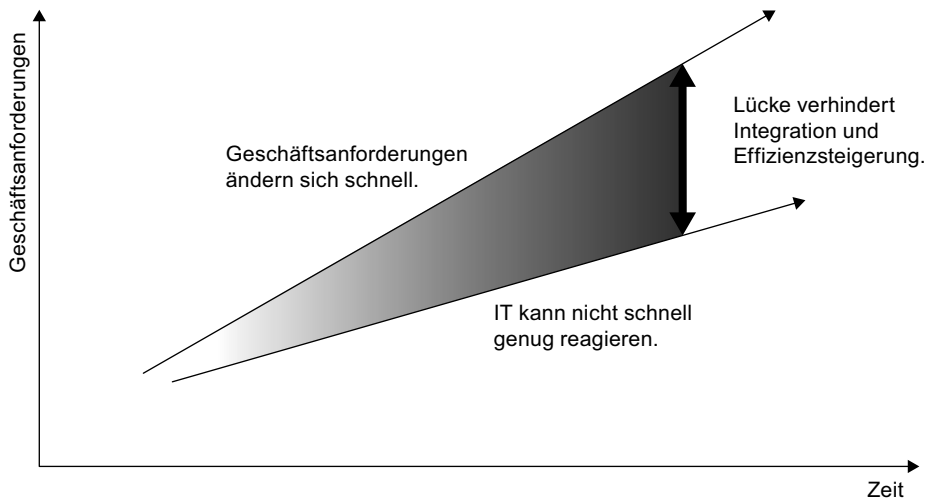


Abb. 1–3 Der Anwendungsstau

Die Herausforderungen der sich immer schneller verändernden Marktbedingungen legen den Einsatz flexibler und effizienter Systeme nahe, Maßnahmen zur Sicherstellung einer zukunftsfähigen IT im Unternehmen scheinen unumgänglich. Die heute mehr denn je geforderte Integration und Effizienzsteigerung stehen dabei in direktem Zusammenhang mit der Leistungsfähigkeit der eingesetzten

Systeme. Solange die strategischen Bereiche eines Unternehmens von veralteten Systemen und Rechnern getragen werden, lässt sich diese Zielsetzung nur schwer realisieren.

Mit diesen Erkenntnissen rücken State-of-the-Art-Technologien wie

- neuere Hardwaretechnologien (Cloud Computing, webbasierte Plattformen),
- neuere Softwareentwicklungsparadigmen (objektorientiertes Programmierparadigma, Aspektorientierung),
- Middleware-Ansätze (WebSphere, NetWeaver),
- Webtechnologien (Java, EJB, XML, J2EE, .Net) und
- serviceorientierte Architekturen (SOA)

in den Mittelpunkt des Interesses.

1.3 Alternativen zur Bewältigung von Altlasten

Obwohl moderne Technologien Integration und Effizienzsteigerung versprechen, ist der Übergang zu ihnen alles andere als einfach. Dabei stellt sich zunächst die Frage nach einer optimalen Lösungsstrategie, mit der sich ein Unternehmen von seinen Erblasten befreien und in eine moderne Umgebung wechseln kann. Dabei steht eine Migration in Konkurrenz zur Beibehaltung des Altsystems, zur totalen Neuentwicklung auf der grünen Wiese oder zum Einsatz von Standardanwendungssoftware [SnSn03, Borc95, Coll04].

Im Falle einer Beibehaltung des Altsystems bleibt dieses in der gewohnten Umgebung und unterliegt den üblichen Erhaltungs- und Weiterentwicklungsaktivitäten. Das Unternehmen spart die Kosten einer – nicht notwendigerweise erforderlichen – Umstellung. Die Vorzüge dieser pessimistischen Strategie werden in [SnSn03] anhand konkreter Praxisbeispiele veranschaulicht, die primär einen unnötigen Technologiewechsel thematisieren. Danach hält nicht jede neue Technologie ein, was sie an Nutzen verspricht. Integration und Effizienzsteigerung lassen sich leider erst nach einer Umstellung konkret messen, und niemand weiß vorher, ob er nicht Opfer einer irreführenden Marketingkampagne wird.

Diesen Vorzügen stehen allerdings gewichtige Einschränkungen gegenüber. Konkurrenzvorteile sind aufgrund der beschränkten Möglichkeiten der Systemänderung im Rahmen von Erhaltung und Weiterentwicklung nur schwer umsetzbar. Bedingt durch die vielfältigen korrektiven und adaptiven Maintenance-Aktivitäten, die sich über den gesamten Lebenszyklus eines Softwaresystems erstrecken, weisen alte Systeme eine zunehmende strukturelle Degeneration (Software-Entropie) auf. Die damit verbundene steigende Komplexität geht einher mit sinkender Qualität und steigenden Maintenance-Kosten. Seacord et al. weisen darauf hin, dass »the impact of many small changes can be greater than their sum« [SePILe03]. Lehman und Belady haben mit ihren fünf Gesetzen der Soft-

wareevolution belegt, dass steigende Komplexität und sinkende Qualität die Weiterentwicklung bremsen und langfristig zur Stagnation führen [LeBe85].

Die drei klassischen Alternativen zur Erhaltung der Aktualität von Softwaresystemen sind Neuentwicklung, Ablösung durch Standardanwendungssoftware und Migration in eine neue Umgebung.

1.3.1 Neuentwicklung

Die Neuentwicklung von Grund auf bedeutet »rewriting a legacy IS from scratch to produce the target IS using modern software techniques and hardware of the target environment« [BrSt93]. Für diese Lösungsstrategie hat sich nach [BrSt95, BrSt93] die Bezeichnung »Cold Turkey«-Ansatz durchgesetzt – ein Amerikanismus für den Sprung ins kalte Wasser. Eine Neuentwicklung dieser Art – basierend auf neuen Technologien – ist in der Regel zeitaufwendig, kostenintensiv und in der Einführung risikoreich.

Die Kosten solcher Entwicklungsprojekte sind in der Regel drei- bis viermal höher als die von Migrationsprojekten und – was noch stärker wiegt – im Vorfeld schwer abschätzbar. Gleichzeitig erreichen solche Projekte nur mit einer Wahrscheinlichkeit von 40 Prozent ihr Ziel [SnSn03]. Bei den übrigen 60 Prozent solcher Projekte wird die Software zu teuer und häufig zu spät oder sogar überhaupt nicht fertiggestellt [Snee97a]. Die Gründe liegen hauptsächlich in den langen Laufzeiten solcher Neuentwicklungsprojekte, die nicht selten mehrere Jahre beanspruchen. Während des Projektfortschritts ändern sich Technologien und die zu unterstützenden neuen Geschäftsprozesse. Bisbal et al. betonen beispielsweise in [BGOL97c] die Gefahr, dass das neue System – noch bevor es einsatzbereit ist – die Bedürfnisse des Unternehmens nicht mehr ausreichend abdeckt und die Technologie bereits »out of date« ist. Die Veränderungen der Geschäftsprozesse lassen sich über Change Requests nachziehen, die Veränderungen der Technologie müssen auf die nächste Migration warten.

Ein signifikantes Risiko besteht vor allem dann, wenn das neue System Unternehmens-Know-how integrieren soll, das über die gesamte bisherige Betriebszeit eingeflossen und sonst nirgendwo außerhalb der alten Systeme dokumentiert ist. Bei einer Neuentwicklung geht dieses Wissen mit dem alten Code verloren. Die Anwender merken bald, dass das neue System nicht alle gewohnten Funktionalitäten seines Vorgängers bietet und Anforderungen des Unternehmens nur unzureichend erfüllt [SePILe03, Warr99, Borc95]. Ein weiteres Risiko besteht dann, wenn das neu zu entwickelnde Legacy-System mit anderen Systemen und Ressourcen interagiert. Sind diese Interaktionen nicht klar dokumentiert und verstanden, kann ein Fehler in der Neuentwicklung zu Fehlverhalten in den abhängigen Systemen führen [BGLO97c].

Dem stehen gewichtige Argumente gegenüber, die für eine Neuentwicklung sprechen. So kann ein System beispielsweise bei der Neuentwicklung um zusätzli-

che Funktionen erweitert werden, was in einer Steigerung des betriebswirtschaftlichen Nutzens resultiert. Gleichzeitig wird diese Produktivitätssteigerung durch die mögliche optimale Ausrichtung des Systems auf die neue Umgebung und die damit einhergehende bessere Ausnutzung der technischen Möglichkeiten weiter begünstigt [SnSn03]. Eine Neuentwicklung ist immer dann sinnvoll, wenn sowohl Zeit als auch finanzielle und personelle Ressourcen in ausreichendem Maße verfügbar sind, das Unternehmen sich von einem neuen und funktional erweiterten System hohen Nutzen verspricht und andere Optionen nicht zum Tragen kommen können.

1.3.2 Ablösung durch Standardanwendungssoftware

Der Einsatz von Standardanwendungssoftware (beispielsweise ERP-Systeme) ist dann empfehlenswert, wenn alle Geschäftsprozesse des Unternehmens mit geringer Anpassung (Customizing) damit abgedeckt werden können. In diesem Fall bildet die Ablösung des Altsystems durch Standardsoftware die einzige vernünftige Lösung [SnSn03]. Die Einführung von Standardanwendungssoftware ist nicht nur mit geringeren Basisinvestitionen (Lizenzen) verbunden, sondern verspricht gleichzeitig die dauerhafte Aktualität der Standardgeschäftsprozesse (z.B. Buchhaltung, Personalverwaltung), da die Software durch den Hersteller gewartet und weiterentwickelt wird.

Ist diese Strategie allerdings mit einem hohen Anpassungsaufwand verbunden, kann sie schnell zu höheren Kosten als eine Eigenentwicklung führen. Entweder muss dann die Standardanwendungssoftware mit hohen Customizing-Kosten den eigenen Geschäftsabläufen oder letztere müssen bei gleichzeitigem Verlust von potenziellen Wettbewerbsvorteilen an die Standardanwendungssoftware angepasst werden. Beides bringt Nachteile für das Unternehmen, denn oft sind die unternehmenseigenen Geschäftsprozesse maßgebend für Wettbewerbsvorteile [Coll04].

1.3.3 Migration der Altsysteme

Als pragmatischer Ausweg aus der Legacy-Software-Falle bietet sich die Weiterverwendung der bestehenden Software durch eine Migration an. Diese Rettung der bestehenden Funktionalität in eine neue Welt ist in der Regel die schnellste und billigste Lösung. Die Neuentwicklung eines vorhandenen Systems hat sich als teures und risikoreiches Unterfangen erwiesen. Auch eine einfache Reimplementierung ist nicht ohne Risiko und mit höheren Kosten verbunden, vor allem was den Test anbetrifft [Borc95, SnSn03]. Migration wird deshalb als »Commonsense solution to the legacy problem« gepriesen [BrSt95].

Auch wenn eine Migration ein gewisses Verständnis des alten Systems erfordert und durch teilweise versteckte Abhängigkeiten unvorhersehbare Seiteneffekte

fekte auslösen kann, stellt sie einen guten Ansatzpunkt für eine alternative Lösung dar. Gewichtige Argumente für eine Migration sind zum einen die Erhaltung des in der Legacy Software enthaltenen Business-Know-how sowie der Schutz der im Laufe der Jahre in die bestehenden Systeme geflossenen Investitionen [Haug05]. Zum anderen kann im Gegensatz zu einer Neuentwicklung von einem vorhandenen System mit messbarem Codeumfang ausgegangen werden, sodass eine zuverlässigere Aufwandsschätzung möglich ist. Eine solche Schätzung ist weitgehend automatisierbar und daher auch mit geringem Aufwand verbunden [Snee03b]. Hinzu kommt, dass Migrationsprojekte in der Regel zum Festpreis angeboten werden, was bei Neuentwicklungen seltener der Fall ist, weil bei diesen nicht alle Anforderungen und Probleme vorhersehbar sind.

Andererseits wird der Nutzen eines migrierten Softwaresystems nie so hoch sein wie der eines mit den Möglichkeiten der neuen Technologien neu konzipierten Systems. So ist im Zuge einer Migration zwar die Überführung in eine neue Umgebung möglich, doch bleibt die Funktionalität unverändert. Bei einer Migration handelt es sich nur um die neue Verpackung eines alten Inhalts.

Softwaremigration wird seit Jahrzehnten praktiziert. Dennoch – trotz massiver Forschungsarbeiten in den Bereichen Maintenance, Sanierung und Reengineering – ist das Gebiet der Systemmigration nur unzureichend von der Wissenschaft erschlossen. Das mag unter anderem daran liegen, dass Altlasten erst seit einigen Jahren als solche wahrgenommen werden [Aebi96]. Weitere Gründe sind die vielen Einschränkungen einer Migration sowie der Zwang, sich mit veralteten Technologien auseinanderzusetzen. Das alles macht das Thema für Forscher wenig interessant. Andererseits ist es für die Industrie aber von höchster Bedeutung. Mit diesem Buch soll das Thema »Migration« auch als Forschungsobjekt ins allgemeine Bewusstsein gerückt werden.

1.4 Migrationsstrategien

Wichtige Voraussetzung für die Wiederverwendung eines bestehenden Systems ist die Wahl einer geeigneten Migrationsstrategie. In diesem Zusammenhang sind namentlich Strategien zu erwähnen, die zur Transformation, Umstellung und Übergabe im Kontext eines Migrationsprojekts zum Einsatz kommen. Softwaremigration überführt bestehende Programme, Daten und Benutzungsschnittstellen in eine andere Form, damit sie in einer anderen Umgebung wiederverwendet werden können. Die Migrationsstrategie spezifiziert für jedes System, durch welche Technik die bestehenden Daten, Programme und Schnittstellen in die neue Umgebung versetzt werden. Grundsätzlich kann eine von drei alternativen Überführungs- bzw. Versetzungsstrategien Anwendung finden: Reimplementierung, Konversion und Kapselung (siehe Abb. 1–4).

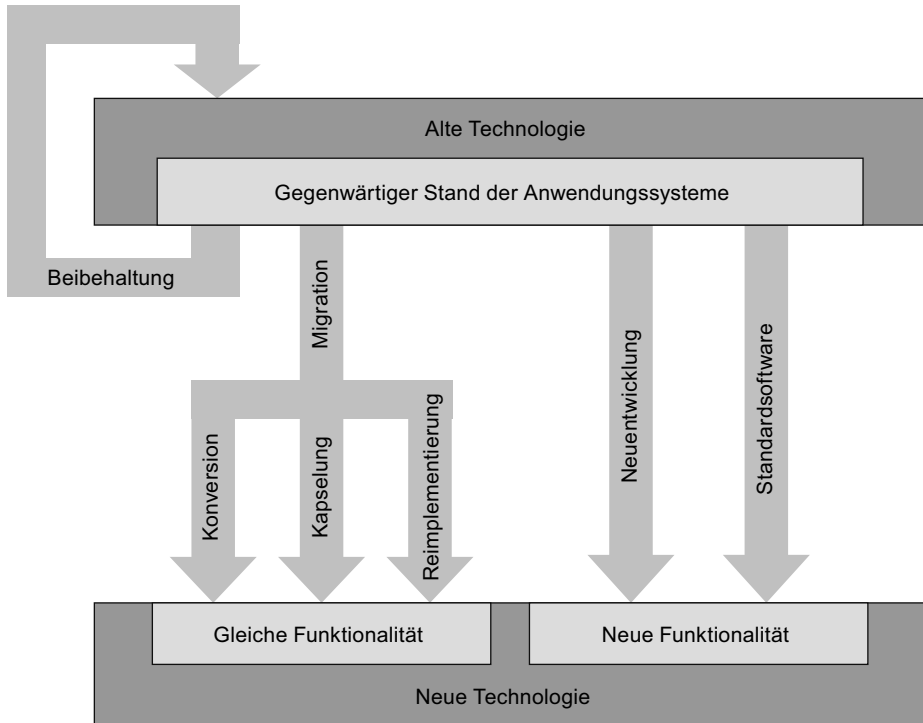


Abb. 1–4 Mögliche Migrationspfade

1.4.1 Reimplementierungsstrategie

Die Reimplementierung umfasst die Wiederverwendung aller aus dem Altsystem extrahierten Informationen zum Zweck der 1:1-Neucodierung des Systems für eine andere Umgebung [Borc95]. Im Gegensatz zum »Cold Turkey«-Ansatz, bei dem losgelöst vom bisherigen System ein neues erstellt wird, bildet hier das Altsystem die Ausgangsbasis. Seine Architektur bleibt erhalten, nur der Code ändert sich. Hierbei werden auch die Daten des Altsystems – wenn notwendig – in neue Strukturen übertragen. Funktionalität und Architektur der Altsoftware werden durch Reverse-Engineering-Techniken reproduziert. Die aus dem Altsystem rekonstruierten Artefakte dienen dann als Basis zur Reimplementierung [Snee01a]. Das Ableiten funktionaler Anforderungen aus dem Altsystem und deren Dokumentation bildet eine wesentliche Voraussetzung.

1.4.2 Konversionsstrategie

Konversion (auch: Konvertierung) bezieht sich auf die in der Regel automatisierte Transformation der Daten und/oder Programme in die für die Zielumgebung notwendige Form. Konversion setzt das Verständnis des Altsystems voraus, das teil-

weise mittels Reverse-Engineering-Techniken erreicht wird. Wer ein Altsystem in COBOL verstehen will, muss auch COBOL verstehen: Reverse Engineering setzt voraus, dass der Benutzer der Reverse-Engineering-Ergebnisse mit der dahinter liegenden Softwaretechnologie vertraut ist. Erst nach der Istanalyse erfolgt die eigentliche Umsetzung der Programme bzw. der Daten. Ein weiterer zu Konversion bzw. Konvertierung synonymen Begriff ist die sogenannte Whitebox Modernization [SePILe03, CRSW00, NSTW97].

Angesichts der Tatsache, dass durch eine Konversion das Altsystem in neuer Form 1:1 in eine neue Umgebung überführt wird, bildet sie die reinste Form der Migration. Es gibt aber gewichtige Argumente, die eine Konversion nicht immer als vorteilhaft erscheinen lassen. Eine Transformation, ob von Code oder Daten, kann nur so gut sein wie das Werkzeug, das sie durchführt. Es ist nicht einfach, solche Werkzeuge zu entwickeln, und sie erfordern eine hohe Vorinvestition. Zu ihrer Amortisation muss die Menge der zu migrierenden Software groß genug sein. Eine manuelle Transformation wäre mit hohen Kosten verbunden, die an jene einer Reimplementierung herankommen. Deshalb kommt die Konversionsstrategie nur zum Tragen, wenn die Transformation der Programme und Daten wirklich weitgehend automatisiert durchgeführt werden kann [SePILe03].

1.4.3 Kapselungsstrategie (Wrapping)

Bei der Kapselung bleiben Daten und Programme des Altsystems in ihrer ursprünglichen Umgebung. Sie werden dort von einem sogenannten Wrapper (einer Softwareschale) umhüllt, der das Altsystem kapselt und geeignete Zugriffsschnittstellen implementiert, über die das Neusystem auf die Dienste des Altsystems zugreifen kann. Die bei einer solchen Verbindung notwendige Übersetzung zwischen den verschiedenen Sprachen ist gleichfalls Aufgabe des Wrapper [SnSn03, Snee99b]. Ein zur Kapselung und zum Wrapping synonymen Begriff ist Blackbox Modernization [SePILe03, CRSW00, NSTW97].

Im Gegensatz zur Konversion ist Kapselung eine Integrationstechnologie und kurzfristig gesehen die ökonomisch sinnvollste Methode der Weiterverwendung [Snee00a]. Eine Kapselung verlangt das Reengineering der Systemschnittstellen, während der Legacy-Code selbst weitestgehend unverändert bleibt. Konsequenzen sind minimales Risiko (da keine tiefen Eingriffe in die Software erfolgen) und minimaler Aufwand. So müssen z.B. nur die externen Schnittstellen des Legacy-Systems durch eine Analyse der Inputs und Outputs untersucht werden. System-interna wie Programme und Datenbanken können als Blackboxes betrachtet werden. Seacord et al. weisen in [SePILe03] jedoch darauf hin, dass sich diese Vorteile nur im Idealfall ergeben. Sie begründen ihre Einschränkung damit, dass sich Kapselung nicht immer als praktikabel erweise und außerdem ein entsprechendes Verständnis der Softwareinterna voraussetze.

Die Kapselungsstrategie hat deutliche Nachteile. Nicht nur, dass die Qualität der Lösung unverändert bleibt, ihre Performance leidet auch darunter. Sie gleicht einem Pflaster auf einer offenen Wunde ohne medikamentöse Versorgung. Sie hat aber zwei schwerwiegende Vorteile: Zum einen ist sie billig und zum anderen relativ risikolos. Dies begründet ihre Anziehungskraft für kurzfristig denkende Führungskräfte, die schnelle Erfolge vorweisen wollen oder müssen.

1.4.4 Auswahl einer geeigneten Strategie

Die hier separat dargestellten Migrationsstrategien kommen in Migrationsprojekten aus technischen und wirtschaftlichen Gründen oft in kombinierter Form zum Einsatz. In einem Bericht des Software Engineering Institute steht »in fact, a combination strategy is often the soundest and most cost-effective approach« [NSTW97]. Allerdings bezieht sich dieser Bericht auf Forschungsprojekte im amerikanischen Verteidigungsministerium, wo Geld keine so große Rolle spielt.

Theoretisch wäre es denkbar, für jede zu migrierende Komponente die jeweils angemessene Strategie zu wählen. Eine solche Entscheidung hinge im Wesentlichen von der Wiederverwendbarkeit, Konvertierbarkeit und allgemeinen Qualität der jeweiligen Komponente, der angestrebten Qualität des Neusystems und der betriebswirtschaftlichen Bedeutung der Komponente ab. Dies zu determinieren, wäre Aufgabe einer Portfolioanalyse. Gleichfalls wären aber auch die vorhandene Zeit bis zur Ablösung, die Anzahl der einsetzbaren Mitarbeiter und letztlich natürlich auch das zur Verfügung stehende Budget wichtige Einflussfaktoren.

In der Praxis werden alle Komponenten aus Kostengründen meist nach der gleichen Strategie umgesetzt [GiWi05]. Die Kosten einer differenzierten Migration sind höher, weil diese das Migrationsprojekt zwingen würde, quasi parallele Fertigungsstraßen aufzubauen.

1.5 Umstellungs- und Übergabestrategien

Neben der Entscheidung über die jeweils geeignete(n) Migrationsstrategie(n) ist die Form der Umstellung und Übergabe (Cut over) des migrierten Systems festzulegen. Unter Umstellung ist hier die Transformation der Software zu verstehen, mit Übergabe ihre Auslieferung. Umstellung und Übergabe bedingen sich in der Regel gegenseitig. Die Umstellungsstrategie bezieht sich auf die Festlegung der Art, nach der das zu migrierende Legacy-System umgesetzt werden soll. Grundsätzlich differenziert man zwischen einer inkrementellen (schrittweisen) und einer Big-Bang- (ganzheitlichen-) Umstellung. Die Übergabestrategie legt, mitbestimmt von der gewählten Umstellungsstrategie, fest, wie die Übergabe (Cut over) des migrierten Systems und damit zusammenhängend die Ablösung des Legacy-Systems erfolgt. Hier wird unterschieden, ob das migrierte System als Gesamtheit zu einem bestimmten Termin oder schrittweise in den operativen Einsatz übergeben

wird. Im Hinblick auf die Ablösung des Altsystems durch das Neusystem finden sich in der Literatur mehrere alternative Übergabestrategien, die alle ihre eigene Charakteristik aufweisen. So liegt jeder Strategie ein eigener Kompromiss zwischen der Minimierung des generellen Fehlerrisikos bei der Übergabe und des zusätzlich durch die Komplexität der Strategie bedingten Risikos zugrunde [BGLO97c].

In der Veröffentlichung von Brodie und Stonebraker [BrSt95] werden zwei Strategieansätze – inkrementell und Big Bang – miteinander verglichen. Die Big-Bang-Umstellung (oder Punktumstellung) geht mit einer Big-Bang-Übergabe einher. Die inkrementelle Paketumstellung bedingt grundsätzlich auch eine inkrementelle Übergabe in Form einzelner, aufeinander folgender Komponentenablösungen. Eine inkrementelle Umstellung mit Paralleleinsatz wird in [BGLW99] vorgeschlagen; diese Umstellungsstrategie ist mit einer Big-Bang-Übergabe assoziiert. Borchers et al. ergänzen die Möglichkeiten in [BoHi98, Borc93] um eine weitere Strategie, die Langfristumstellung, bei der die migrierten Pakete inkrementell übergeben werden. Eine Übergabe stellt sich dabei weitaus komplexer dar als ein einfaches Umschalten vom Alt- auf das Neusystem. Die Herausforderungen liegen im Wesentlichen in der Minimierung der Risiken in Bezug auf ein Fehlverhalten des Neusystems und die damit verbundene Möglichkeit einer Rückkehr zum Altsystem (Fallback) als Sicherheit für einen potenziellen Zielsystemausfall [BrSt95].

Gleichzeitig muss während der Dauer des Migrationsprojekts die parallele Aufrechterhaltung des Normalgeschäfts, d.h. die Kontinuität der IT-Dienstleistung, gewährleistet sein. Dies ist insbesondere bei betriebskritischen Systemen der Fall, deren gesamte Funktionalität – einschließlich der Daten – über die Projektlaufzeit einsatzfähig bleiben muss [BrSt95].

1.5.1 Punktumstellung und Punktübergabe (Big Bang)

Die Punktumstellung bezieht sich, wie in Abbildung 1–5 dargestellt, auf die zunächst vollständige Abarbeitung des gesamten Softwareportfolios in einem möglichst kurzen Zeitraum mit anschließender Ablösung des Altsystems auf einen Schlag durch das Neusystem. In [BGLW99] wird für die Punktumstellung die synonyme Bezeichnung Cut-and-Run-Strategie verwendet. Das Ablösen eines Altsystems durch ein Neusystem in einem Schritt ist mit gravierenden Nachteilen verbunden. Bisbal et al. weisen in [BGLW99, BGLO97c] insbesondere auf das mit dieser Strategie verbundene hohe Risiko hin: Der Informationsfluss im Unternehmen nach der Umstellung basiert auf einem unerprobten und somit potenziell unzuverlässigen System. Auch Borchers betont in [Borc93] ein hohes Produktionsrisiko und kommt zu der Schlussfolgerung, dass eine Big-Bang-Umstellung für große Migrationsprojekte nicht zu empfehlen ist. In Deutschland, wo man etwas vorsichtiger handelt als in Amerika, findet die Punktumstellung in abgeschwäch-

ter Form statt: Hier erfolgt ein begrenzter Piloteinsatz des Neusystems an einem Ort mit ausgewählten Anwendern, dem dann ein schrittweises Rollout folgt. Das Hauptargument für die Big-Bang-Umstellung sind – bei korrekter und erfolgreicher Transformation – die reduzierten Gesamtkosten der Migration. So muss beispielsweise das Gesamtsystem nur einmalig getestet und übergeben werden [SePILe03].

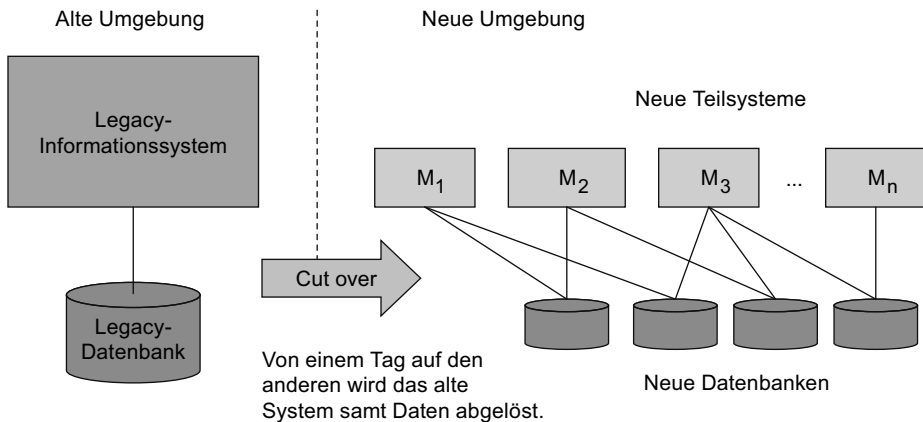


Abb. 1-5 Punktumstellung

1.5.2 Langfristumstellung

Bei der Langfristumstellung werden jeweils nur die Softwarekomponenten bearbeitet und in die neue Umgebung übernommen, die ohnehin im Rahmen der normalen Wartungsaktivitäten zur Änderung angefasst werden müssen [Borc93, BoHi98]. Die restlichen Komponenten bzw. diejenigen, die unverändert bleiben, werden erst bei Bedarf für die Zielumgebung umgestellt.

Die Eignung einer Langfristumstellung ist, primär aufgrund technischer Restriktionen, auf rein technische und unkomplizierte Migrationsaufgaben beschränkt. Ein typisches Beispiel ist nach [BoHi98] die Umstellung auf eine neue Compiler-version. Betrachtet man dagegen eine Datenumstellung, so ist aufgrund der dann notwendigen umfassenden Bearbeitung aller betroffenen Softwarekomponenten leicht ersichtlich, dass eine Langfristumstellung nicht infrage kommt [Borc93].

Eine nicht unwesentliche Gefahr bei dieser Strategie liegt in der extrem langen Projektlaufzeit, die sich zu einer unendlichen Geschichte [Borc93] hinziehen kann. So ist eine Langfristumstellung in der Regel immer damit verbunden, dass noch nach Jahren Legacy-Komponenten übrig bleiben, die irgendwann doch noch bearbeitet werden müssen [BoHi98].

1.5.3 Inkrementelle Paketumstellung

Bei paketerorientierter Vorgehensweise wird das Altsystem inkrementell durch das Neusystem abgelöst. Hierzu wird das gesamte zu bearbeitende Softwareportfolio in überschaubare – aber notwendigerweise voneinander unabhängige – Pakete zerlegt, die nacheinander, in einer bestimmten Reihenfolge und einem überschaubaren Zeitraum, bearbeitet werden. Eine anspruchsvolle Migrationsaufgabe ist nach [Borc93, BoHi98] im Vergleich zur Big-Bang- oder zur Langfristumstellung nur mit der inkrementellen Paketumstellung beherrschbar. Das Aufbrechen des gesamten zu migrierenden Portfolios in eine überschaubare Menge von getrennten Paketen, die schrittweise überführt werden können und so zu einer Risikominimierung innerhalb der Umstellungsphase führen, ist eine der Stärken dieser Strategie [BGLO97c, BoHi98]. Die Risikominimierung resultiert beispielsweise aus den Erfahrungen der ersten Umstellungen, die in die nachfolgenden Umsetzungen einfließen können. Darüber hinaus sind kleinere Schritte typischerweise besser handhabbar und leichter zu evaluieren [SePILe03].

Andererseits ist die inkrementelle Paketumstellung mit einem höheren Planungs- und Koordinationsaufwand verbunden. Zum einen stellt die Bildung voneinander unabhängiger Pakete eine erhebliche Herausforderung dar. Zum anderen führt die technische Kopplung zwischen beiden Systemen zu einer höheren Komplexität bei der Umstellung [BrSt95, BGLO97c].

Die Paketbildung bildet den kritischen Faktor bei dieser Form der Umstellung. Hierzu müssen alle bestehenden Abhängigkeiten, auch die nicht direkt erkennbaren, sowohl in fachlicher als auch in technischer Sicht identifizierbar sein [BoHi98]. Sowohl Anwendungen als auch Daten in (funktional) unabhängige Pakete zu zerlegen, die dann nacheinander migriert werden können, ist eine äußerst anspruchsvolle Aufgabe. Diese wird zusätzlich erschwert durch den typischerweise monolithischen und unstrukturierten Charakter bestehender Legacy-Systeme [BGLO97b]. In solchen Fällen kann die Bildung voneinander unabhängiger Pakete sogar unmöglich zu realisieren sein, sodass entweder geeignete Entkopplungsmechanismen [BoHi98] implementiert werden oder Komponenten in beiden Systemen (als Kopien) existieren müssen [BrSt95]. Eine Auflistung der bei der Paketbildung zu beachtenden Aspekte gibt [BoHi98].

Die Paketumstellung findet in zwei Varianten Einsatz, die im Hinblick auf die während des Migrationsprojekts existierende Systemarchitektur divergieren: Es sind die inkrementelle Paketumstellung mit Komponentenersatz und die inkrementelle Paketumstellung mit Paralleleinsatz.

1.5.4 Inkrementelle Paketumstellung mit Komponentenersatz

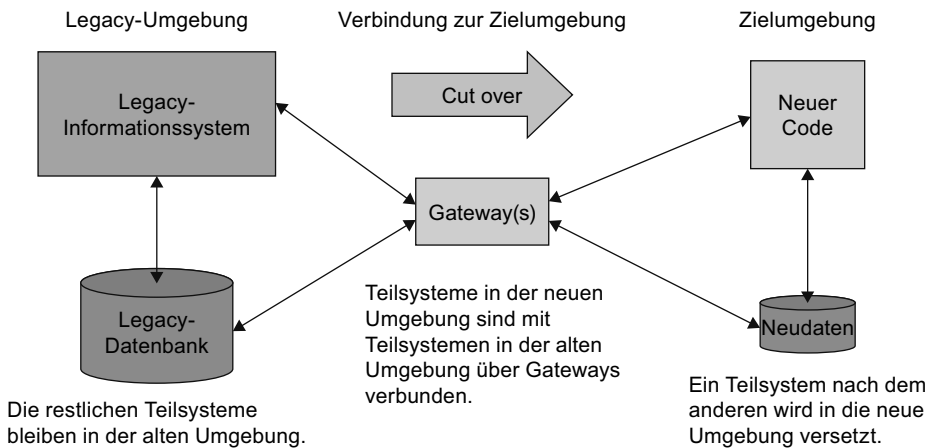


Abb. 1-6 Inkrementeller paketweiser Übergang

Jeder Übergabeschritt resultiert, wie Abbildung 1-6 darstellt, in einem Austausch der Komponenten des Legacy-Systems durch korrespondierende Komponenten im Zielsystem; das Altsystem schrumpft, das Zielsystem wächst [BGLO97b]. Angesichts dieser inkrementellen Vorgehensweise wird die Paketumstellung im Vergleich zur Big-Bang-Umstellung häufig auch als sanfte Migration bezeichnet [GiWi05].

Während der Dauer des Migrationsprojekts besteht das im Einsatz befindliche Gesamtsystem aus einer Kombination von Ziel- und Legacy-System. Die erforderliche Interoperabilität der beiden Systeme wird durch den temporären Einsatz von Gateways (Datenaustauschbrücken) sichergestellt [BGLO97a, BrSt95]. Ein Gateway ist ein Stück Kommunikationssoftware, »a module introduced between operational software components to mediate between them and to translate requests and data between the mediated components« [BrSt95]. Gateways übernehmen im Wesentlichen drei Funktionen:

- Sie isolieren Komponenten, an denen Veränderungen vorgenommen werden, von den restlichen Komponenten.
- Sie übersetzen Nachrichten zwischen Legacy- und Zielkomponenten.
- Sie koordinieren zwischen den Komponenten und implementieren die zur Konsistenzhaltung zwischen den beiden (heterogenen) Systemen notwendigen Mechanismen.

Die Kopplung zwischen schon bearbeiteten und noch nicht bearbeiteten Komponenten resultiert in einer heterogenen Umgebung mit verteilten Anwendungen und verteilten Datenbanken. Die notwendigerweise zur Interoperabilität eingesetzten Gateways führen zu einer nicht unerheblichen Komplexitätssteigerung bei der

Umstellung. Eine wesentliche technische Herausforderung liegt vor allem in der Koordination von Abfragen (Queries) und Updates hinsichtlich replizierter Daten und Funktionen [BrSt95]. Die hierfür zusätzlich erforderlichen Maßnahmen beziehen sich auf das Management von Transaktionen sowie auf die Wahrung der Konsistenz replizierter Daten in verteilten Datenbanken [BGLO97c, BrSt95]. Brodie und Stonebraker [BrSt95] beschreiben die hieraus resultierenden Konsequenzen folgendermaßen: »Update consistency across heterogeneous information systems is a much more complex technical problem with no general solution.«

Zusammenfassend kommen Bisbal et al. in ihrer Studie »A Survey of Research into Legacy System Migration« [BGLO97c] zu der Schlussfolgerung, dass die Verwendung von Gateways den ohnehin schon komplexen Umstellungsprozess zusätzlich erschweren kann. Sie behaupten: »Migrating a legacy system in an incremental fashion is designed to reduce the risk of the migration phase. However, its inherent complexity carries with it a high risk which may actually result in increasing the risk involved in migration«, und postulieren daher, dass die erfolgreiche Durchführung einer Migration eine Balance zwischen diesen beiden Risikoquellen verlange.

1.5.5 Inkrementelle Paketumstellung mit Paralleleinsatz

Diese Umstellungsstrategie beinhaltet ebenfalls eine inkrementelle Überführung des Altsystems in die Zielumgebung. Im Gegensatz zur Paketumstellung mit Komponentenersatz befindet sich zur Aufrechterhaltung des Normalbetriebs während des Migrationsprozesses aber nur das Altsystem im Einsatz. Seine eigentliche Ablösung erfolgt erst nach der vollständigen Migration aller betroffenen Komponenten und dem Sicherstellen der Zuverlässigkeit des gesamten Neusystems (siehe Abb. 1–7).

Während der Migration wird das Neusystem durch entsprechende Transformationen unter permanentem Testen schrittweise in die neue Umgebung überführt. Das Zielsystem dient vorläufig jedoch nur für Entwicklungs-, Test- und Trainingszwecke. Diese Strategie versucht durch eine Gateway-freie Umstellung weitestgehend die Nachteile der inkrementellen Paketumstellung mit Komponentenersatz zu vermeiden.

Eine Umstellung dieser Art bedingt das »Einfrieren« (Sperrern) aller bereits migrierten Komponenten des Altsystems für weitere Wartungs- und Änderungsaktivitäten. Für das Einfrieren der Legacy-Datenbank bei gleichzeitiger Möglichkeit einer Datenänderung innerhalb des laufenden Geschäftsbetriebs werden sowohl temporäre Speicher (TempStores, TS) als auch ein Data Access Allocator (DAA) benötigt. Der Data Access Allocator adressiert alle angesprochenen Daten auf den jeweils gültigen TempStore um und ist zugleich für die Zugriffe auf den korrekten Datenspeicher zuständig.

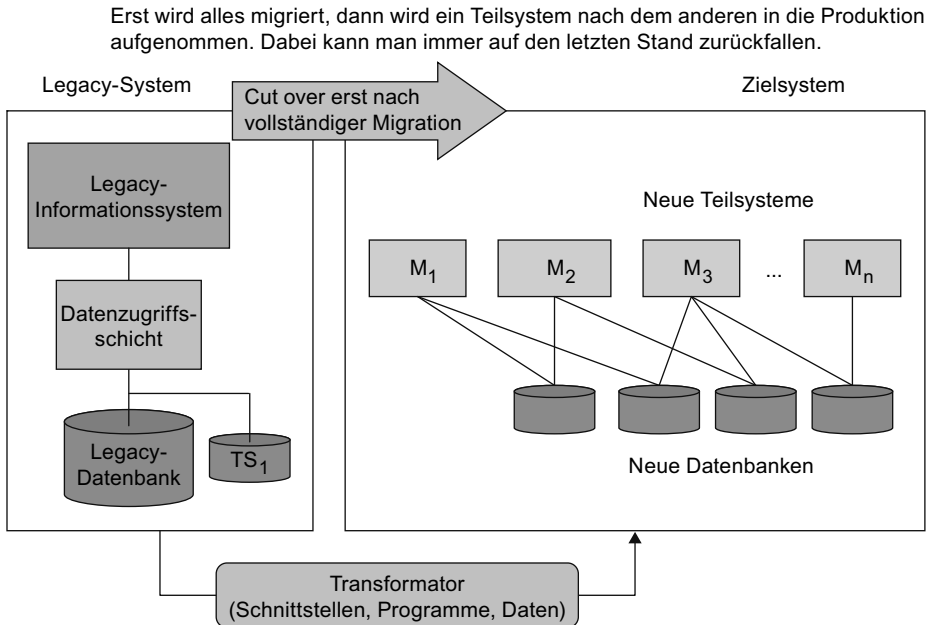


Abb. 1-7 Ablösung nach vollständiger Migration

1.6 Zur Wirtschaftlichkeit einer Softwaremigration

Die Kosten einer Migration sind nach [SnHaTe05] abhängig von der Systemgröße, dem Automatisierungsgrad, dem Testaufwand und der Kluft zwischen dem Istzustand und dem Zielsystem. Letzteres lässt sich nur dann genau bestimmen, wenn man die Auswirkung der Systemabhängigkeiten aus der technischen Perspektive betrachtet und deren – nicht immer offensichtliche – Wechselwirkungen identifiziert. Solche Wechselwirkungen variieren von Projekt zu Projekt in Abhängigkeit von der Art des Migrationsobjekts. Nicht zu vergessen ist, dass sich fachlich nichts ändern darf. Am besten ist es, wenn die Anwender nur an der neuen Gestaltung der Benutzungsoberfläche merken, dass sich etwas geändert hat. Der Inhalt muss unverändert bleiben.

So kann beispielsweise der Wechsel eines bestehenden Datenbankmanagementsystems nicht nur die Migration der Daten in neue Datenstrukturen und/oder Datentypen auslösen, sondern auch eine simultane Modifikation aller betroffenen Teile der Programmlogik (I/O-Operationen und Datenbankzugriffe) herbeiführen. Ein anderer Zusammenhang ergibt sich beim Übergang vom Host in eine Client/Server-Umgebung. Da UNIX und PC-Rechner keine Compiler für Assembler und PC-Rechner keine Compiler für PL/1-Programme zur Verfügung stellen, müssen die Anwendungen notwendigerweise in eine andere Sprache wie beispielsweise C++ konvertiert werden.

In Anlehnung an [Coll04] weist jedes Migrationsvorhaben eine eigene Problematik auf, die es von allen anderen Migrationsprojekten unterscheidet. Gemeinsam sind ihnen dagegen generelle Randbedingungen. Diese Randbedingungen werden im Folgenden aus unterschiedlichen Perspektiven betrachtet und sollen als abstraktes Modell verstanden werden, das wesentliche Faktoren zur Charakterisierung eines Migrationsprojekts identifiziert.

Diesem Modell zur Untersuchung einer Migration liegen drei unterschiedliche Sichten zugrunde:

- Betriebswirtschaftliche Sicht
- Projektmanagement-Sicht
- Technische Sicht

1.6.1 Betriebswirtschaftliche Sicht

Vom betriebswirtschaftlichen Gesichtspunkt aus sind die Unternehmensziele hinsichtlich der Migration von besonderer Bedeutung. Diese umfassen primär die optimale Unterstützung der Kerngeschäftsprozesse durch neue Technologien. Sekundäre Ziele berücksichtigen sowohl die Minimierung des Migrationsrisikos und die Reduzierung des Migrationsaufwands als auch den unterbrechungsfreien Betrieb des Systems während der Migrationsdurchführung. Aus wirtschaftlicher Sicht empfiehlt sich zunächst die Durchführung einer Kosten-Nutzen-Risiko-Analyse. Ein hierfür wesentliches Hindernis ergibt sich daraus, dass Migrationsprojekte meistens durch die übermäßige Komplexität und mangelhafte Qualität der Altsysteme sowie durch deren fehlende oder veraltete Dokumentation, gekoppelt mit fehlendem System-Know-how zu den alten Technologien, erschwert werden.

Das Verstehen des zu migrierenden Systems und der Zielumgebung bildet nicht nur die Grundlage für die Durchführung einer solchen Analyse, sondern kann auch die wesentliche Prämisse dafür sein, dass das resultierende Zielsystem weiterhin optimal zur Unterstützung der Geschäftsprozesse beiträgt. Zur erforderlichen Komplettierung der Dokumente des Altsystems werden u.a. Reverse-Engineering-Techniken eingesetzt. Aus Gründen der Aufwandsminimierung sollte die Redokumentation zielgerichtet erfolgen [Coll04]. Es ist auch wichtig, die Altsoftware zu messen, um Kennzahlen für die Kalkulation des Migrationsaufwands zu gewinnen. Zur Minimierung des Migrationsrisikos wird nach [SnHaTe05] die Durchführung von Probeprojekten (Pilotprojekten) empfohlen, die unüberwindbare Hürden frühzeitig erkennen lassen.

1.6.2 Projektmanagement-Sicht

Aus Sicht des Projektmanagements ist zu bedenken, dass Migrationsprojekte Mitarbeiter mit speziellen Erfahrungen und Kompetenzen im Bereich Migration voraussetzen. Für den Einsatz anspruchsvoller Methoden, Techniken und Werkzeuge als Garant für die effiziente Projektdurchführung ist zudem eine angemessene Qualifizierung in Migrationstechniken erforderlich. Die von einem Migrationsprojekt verlangte technische Spezialisierung macht die Vergabe des gesamten Projekts oder von Teilen davon an externe Dienstleister fast immer notwendig. Angesichts der besonderen charakteristischen Eigenart von Migrationsprojekten, insbesondere der technischen Spezialisierung sowie der einfachen Überprüfung von Projektergebnissen durch Regressionstests, stellen sie sogar prädestinierte Kandidaten für das Outsourcing dar [SnHaTe05]. Demzufolge sollte die Möglichkeit der Fremdvergabe auf jeden Fall berücksichtigt werden.

Weiterhin ist schon während der Projektlaufzeit eine entsprechende Qualifizierung der Benutzer im Umgang mit dem Neusystem zu planen und durchzuführen. Im Rahmen dieser Maßnahme ist die Verfügbarkeit einer entsprechenden Trainingsumgebung sicherzustellen. Auch wenn Betriebsfremde die eigentliche Migration durchführen, darf der Aufwand für das Training der eigenen Mitarbeiter nicht unterschätzt werden.

1.6.3 Technische Sicht

Die technische Sicht richtet sich einerseits auf das Altsystem und andererseits auf das gewünschte Zielsystem. Migrationsprojekte können migrationsbedingte Änderungen sowohl in der Hardwareumgebung, der Laufzeitumgebung, der Systemarchitektur als auch in der Softwareentwicklungsumgebung auslösen. Betroffen sind dabei Daten, Programme und/oder Benutzungsschnittstellen [GiWi05]. Alle diese technischen System- und Softwareaspekte bedingen sich in der Regel gegenseitig. Sie zu erkennen ist eine wesentliche Voraussetzung für eine erfolgreiche Migration.

Die Wiedergewinnung wichtiger Systeminformationen ist ein nicht zu unterschätzender Aufgabenbereich, der nur durch den Einsatz entsprechender Reverse-Engineering-Techniken und korrespondierender Werkzeuge bewältigt werden kann. Für die Durchführung von Änderungen bedarf es guter technischer Kompetenzen hinsichtlich Ausgangs- und Zielsystem. In vielen Fällen gelten vor allem Kenntnisse und Erfahrungen in Bezug auf die Zielumgebung als primäre Erfolgsvoraussetzung. Dies zeigt sich letztlich dann, wenn die Möglichkeiten der neuen Technologie aufgrund mangelnder Kenntnisse nicht ausgeschöpft werden und somit der versprochene Nutzen, den die Migration bringen soll, nicht realisiert werden kann.

1.7 Zum Aufbau des Buches

Zum Verständnis einer Technologie wie der Softwaremigration ist es notwendig, zwischen Prozessen, Methoden, Techniken und Werkzeugen zu unterscheiden. Sie bilden einen sogenannten Technologiebaum mit vier Stufen (siehe Abb. 1–8).

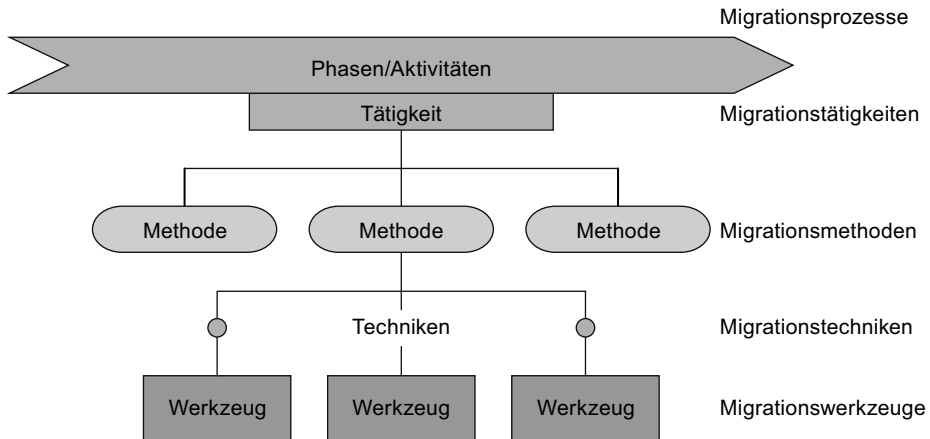


Abb. 1–8 Migrationsmethodenhierarchie

An der Spitze des Baums steht der Prozess, in dem die einzelnen Phasen bzw. Tätigkeiten abgegrenzt, definiert und zeitlich abgestimmt sind. Sie können nach- oder nebeneinander ablaufen. Auf der zweiten Stufe werden die Methoden zur Durchführung der im Prozess geschilderten Aktivitäten spezifiziert. Es besteht eine 1:n-Beziehung zwischen Aktivitäten und Methoden: Eine Methode bezieht sich immer auf eine ganz bestimmte Aktivität, aber zu einer Aktivität können viele Methoden gehören. Auf der dritten Stufe kommen die einzelnen Techniken dazu. Eine Technik ist eine genaue Vorschrift zur Implementierung einer Methode. Sie liefert auch ein ganz bestimmtes Ergebnis, ein Deliverable. Eine Technik implementiert einen Teilaspekt einer einzigen Methode. Da sie sich auf einen Teilaspekt beschränkt, benötigt eine Methode viele Techniken für ihre Implementierung. Auf der vierten und untersten Stufe des Technologiebaums befinden sich die Werkzeuge. Eine wohldefinierte Technik ist auch automatisierbar, Softwarewerkzeuge sind nichts anderes als programmierte Techniken. Da Migrationsprozesse, ihre Methoden und Techniken ziemlich exakt definierbar sind, liegt es auf der Hand, dass sie auch automatisierbar sind. Ein Werkzeug kann eine oder mehrere Techniken automatisieren. Eine Technik wird wiederum nur einem Werkzeug zugeordnet. Darum besteht eine m:1-Beziehung zwischen Werkzeugen und Techniken.

Die Aufbaustruktur des vorliegenden Buches ist angelehnt an diesen Technologiebaum:

- Auf dieses erste einleitende Kapitel folgen in Kapitel 2 die Erläuterung der Begriffe und die Darlegung der Grundlagen.
- In Kapitel 3 werden verschiedene Migrationsprozesse aus Forschung und Praxis vorgestellt.
- In Kapitel 4 wird ein generischer Migrationsprozess namens ReMiP als Bezugsrahmen für sämtliche Migrationsprojektarten vorgeschlagen. Dieses Referenzprozessmodell steht im Mittelpunkt der Betrachtung: Alle danach folgenden Methoden und Techniken beziehen sich darauf.
- In Kapitel 5 werden die Methoden und Techniken sowie
- in Kapitel 6 die Werkzeuge beschrieben, die zur Implementierung des Prozessmodells erforderlich sind.
- Um zusätzlichen Praxisbezug herzustellen, werden in Kapitel 7 Migrationsfallstudien aus Europa vorgestellt. Damit erhält der Leser einen Einblick in die Migrationspraxis.
- Zum Schluss folgt in Kapitel 8 ein Wegweiser für die Migration in eine serviceorientierte Architektur. Der Leser erfährt dort, wie es in der nächsten Migrationswelle weitergeht: Sie wird bestimmt nicht die letzte sein.