

Hauptseminar
„Management von Softwaresystemen“
Prof. Dr. Dr. h.c. Manfred Broy
Thema
„Legacy Migrationsstrategien“

Bearbeitung: Christoph Erdle

Betreuung: Dr. Markus Pizka

13. Dezember 2005

Inhaltsverzeichnis

1	Begriffsklärung und Ausgangssituation	3
1.1	Was sind „Legacy Systeme“	3
1.2	Was ist „Migration“	3
1.3	Wann sollte migriert werden	3
1.4	Warum sollte migriert werden	4
1.5	Wirtschaftliche Aspekte der Migrationsfrage	5
2	Verschiedene Migrationsstrategien	6
2.1	Anforderungen an eine Migration	6
2.2	„Chicken Little“ - Ansatz	7
2.2.1	Database First	8
2.2.2	Database Last	8
2.3	„Cold Turkey“/Big Bang - Ansatz	9
2.4	Butterfly	10
2.5	Vergleich und Bewertung der Ansätze „Chicken Little“, „Cold Turkey“ und „Butterfly“	12
3	Migration verschiedener Systemstrukturen mit „Chicken Little“	13
3.1	Migration von vollständig zerlegbaren Systemen	14
3.2	Migration von teilweise zerlegbaren Systemen	16
3.3	Migration von monolithischen Systemen	17
4	Datenmigration	18
5	Beispiel einer einfachen Migration: Übergang von einer Windows-NT4-Domain auf eine Windows-2000/2003-Domain	19
5.1	Warum und wann findet eine Migration statt	19
5.2	Zu berücksichtigende Probleme	20
5.3	Durchführung der Migration	21
6	Zusammenfassung	22
7	Abbildungsverzeichnis	22
8	Literaturverzeichnis	22

1 Begriffsklärung und Ausgangssituation

1.1 Was sind „Legacy Systeme“

Als „Legacy System“ wird ein Computersystem (sowohl Hard- als auch Software) bezeichnet, das im Verlauf des Einsatzes insgesamt komplizierter wurde, als es zu sein nötig wäre. Der Grund hierfür ist, dass das System immer wieder an aktuelle Gegebenheiten angepasst werden musste, was oft durch sog. „Quick Fixes“ gemacht wurde, und sich das System somit immer weiter von der zugrundeliegenden Spezifikation entfernte. Solche System sollten auf lange Sicht von aktuellen Systemen abgelöst werden.

Der Begriff impliziert im informatischen Verständnis die Eigenschaften „überholt“, „veraltet“ oder „wartungsresistent“. Vor allem im Zusammenhang mit betrieblichen Informationssystemen wie CRM (Customer Relationship Management), ERP (Enterprise Resource Planning) wird der Begriff der Legacy Systeme gebraucht. Sie zeichnen sich dadurch aus, dass nur noch wenige Personen Wissen über die Wartung und Anpassung haben, und allein hierdurch bereits Probleme entstehen können, da auch durch die Einarbeitung neuer Mitarbeiter in das System Kosten bzw. Zeitaufwand entstehen, die möglicherweise durch die Ablösung des Systems auf lange Sicht verringert werden können.

1.2 Was ist „Migration“

Unter Migration wird in der Informationstechnik der Umstieg eines wesentlichen Teiles eines Systems auf ein anderes sowie den Transfer von Daten aus einer Umgebung in eine andere verstanden. Diese beiden oft eng miteinander verknüpften Prozesse lassen sich unterteilen in Datenmigration und Systemmigration.

1.3 Wann sollte migriert werden

Die Migration eines Legacysystemes ist in seiner zeitlichen Einordnung wohl zu überlegen. Sollte es sich beim neuen System um ein bereits existierendes Produkt handeln, ist es wichtig, dass es bereits einen gewissen Reifegrad besitzt, um Korrekturen und Erweiterungen, die im Lebenszyklus eines Systems zwangsläufig auftreten, nicht schon zu Beginn der Arbeitszeit des Systemes ausführen zu müssen, sondern eine möglichst stabile, in diesem Sinne sich nicht ändernde, Co-debasis zu verwenden.

Auch ist wichtig für die Entscheidung für ein System, ob bereits genügend Erfahrungswerte und (möglicherweise zukaufbares) Know-How für den Einsatz des Systems vorhanden sind, oder ob noch abgewartet werden sollte, bis diese Voraussetzungen erfüllt sind. Andernfalls besteht die Gefahr für „early adopters“,

dass bei Problemen sowohl bei der Migration als auch im anschließenden Betrieb des Systems erst langwierige Fehlersuche betrieben werden muss, die durch eine entsprechende Erfahrungsbasis vermieden werden kann.

Als Alternative zur Ablösung durch ein fertiges Produkt bietet sich die Entwicklung einer individuellen, auf das eigene Unternehmen zugeschnittenen Lösung durch das Unternehmen selbst oder externe Dienstleister an. Dies ist möglicherweise in der Anfangsphase kostenintensiver, jedoch entfallen etwa Kosten für die Anpassung der internen Arbeitsabläufe des Unternehmens an die Software, im Gegensatz: das System wird exakt an die Ansprüche des Unternehmens angepasst.

Als Entscheidungskriterium für eine Migration bieten sich Bewertungsprozesse wie „SRAH“ an, die eine objektive Sicht auf ein System ermöglichen (s. 1.4).

Es ist daher abzuwägen, ob Gefahren wie Zeit- und Geldverlust bei Problemen in der Migrationsphase durch die Vorteile des neuen Systems, z.B. neue Funktionen, kompensiert werden.

1.4 Warum sollte migriert werden

Die Entscheidung für eine Migration wird oftmals nicht objektiv, sondern „aus dem Bauch“ gefällt, was jedoch oft nicht zum gewünschten Ergebnis führt. Eine objektive Methode, zu einer Entscheidung über eine Migration zu finden, ist der Einsatz von Bewertungsprozessen wie „SRAH“, der vom amerikanischen Verteidigungsministerium (United States Department of Defense, DoD) entwickelt wurde.

„SRAH“ verwendet verschiedene Kriterien und definiert ein strukturiertes Vorgehen für die Bewertung der einzelnen Faktoren in den Bereichen „Technik“, „Wirtschaftlichkeit“ und „Management“:

- Im Bereich „Technik“ ist die Frage angesiedelt, welches System grundsätzlich für eine Migration in Frage kommen kann, und welche Alternativsysteme zur Verfügung stehen (Ersatzprodukt oder Neuentwicklung).
- Unter „Wirtschaftlichkeit“ findet man Punkte wie „Kostenschätzung“ und „Risikoanalyse“ aufbauend auf den Ergebnissen im Bereich „Technik“.
- Mit diesen Ergebnissen als Grundlage werden im Bereich „Management“ Kriterien wie Kundenwirkung, strategische Entscheidung für einen bestimmten Hersteller, u.a. mit in die Entscheidung einbezogen.

Beispielhafte Fragen sind etwa:

- Gibt es für das System noch Support durch den Hersteller? Ist noch genügend Know-How zum Einsatz des Systems vorhanden, so dass im Problemfall schnell reagiert und Lösungen gefunden werden können?

- Informationssysteme werden oft über lange Zeiträume (> 15 Jahre) eingesetzt. Funktioniert das System auch auf aktueller Hardware, falls kein gleichwertiger Ersatz für defekte Komponenten gefunden werden kann?

Dies sind essentielle Fragestellungen, wie sie nach längerem Betrieb eines Systems entstehen. Können die Fragen nicht mehr zufriedenstellend beantwortet werden, sollte eine Systemlösung gesucht werden, die die Anforderungen wieder erfüllen kann.

1.5 Wirtschaftliche Aspekte der Migrationsfrage

Auf die Entscheidung, ob eine Migration durchgeführt werden soll, haben beispielsweise folgende technische Aspekte Einfluss:

- Umstellung von Zentralstruktur auf Client-Server-Konzept (Mainframe mit Thin Clients vs. Server mit Fat Clients)
- Ist für das System noch Ersatzhardware vorhanden?

Größeren Einfluss sollten aus Sicht der Entscheider (in den seltensten Fällen mit technischem, sondern betriebswirtschaftlichem Hintergrund) (betriebs-)wirtschaftliche Aspekte haben.

Hier stellt sich u.a. die Frage, ob bereits ein geeignetes Alternativsystem auf dem Markt existiert bzw. einfach an die Anforderungen angepasst werden kann, oder ob die vollständige Neuentwicklung einer Individuallösung notwendig ist. Der Einsatz bereits existierender Lösungen ist am Anfang meist kostengünstiger, da der Hersteller durch eine breitere Einsatzbasis die Entwicklungskosten über alle eingesetzten Systeme aufteilen kann, die Kosten von Individuallösungen sich jedoch durch einen einzigen Kunden amortisieren müssen.

Ein weiterer Aspekt ist die finanzielle Prosperität des Unternehmens, das eine Ersetzung erwägt. Kann sich das Unternehmen die Ersetzung finanziell zum jetzigen Zeitpunkt überhaupt leisten, müssen Rücklagen hierfür aufgelöst werden?

Auch ist zu überlegen, wer die Migration und Implementierung im Unternehmen durchführt. Ein Unternehmen mit entsprechendem Wissen innerhalb des eigenen Unternehmens kann durchaus in der Lage sein, eine entsprechende Migration selbst durchzuführen. Ist dies nicht der Fall, muss externes Wissen eingekauft werden, was die Kosten im Gegensatz zur In-House-Lösung ansteigen lässt.

Zu Berücksichtigen bleiben ausserdem noch die Kosten für entsprechende Schulungen der Mitarbeiter auf dem neuen System. Bestehende Verträge mit Dienstleistern für das Altsystem ziehen ebenfalls zusätzliche Kosten nach sich, da die Verträge meist nicht zum Betriebsende des Altsystems gelöst werden können

(Kündigungsfristen, feste Laufzeiten, ...). Oft ist auch der Ersatz der Betriebsplattform nötig, da das neue System auf dem bereits vom Altsystem vorhandenen Hardware nicht funktioniert. Hier sind wichtige Kennzahlen beispielsweise wirtschaftliche Restwert des Altsystems (Abschreibungen, ...).

Eine weitere Frage ist, in wie weit sich das Unternehmen durch eine Migration an einen Anbieter bindet, und wie lange von diesem Unterstützung für das System zu erwarten ist (Probleme bei Firmen der New Economy mit schnellen, nicht vorhersehbaren Pleiten, dot-com bubble).

2 Verschiedene Migrationsstrategien

2.1 Anforderungen an eine Migration

Laut Brodie und Stonebraker sollte eine Migration mindestens den folgenden Anforderungen gerecht werden:

- ununterbrochenen, sicheren, zuverlässigen Betrieb garantieren: Ausfälle zentraler Systeme wie betrieblicher Informationssysteme kann eine Unternehmung nicht über längere Zeit verkraften, auch kürzeste Ausfälle führen zu (finanziellen) Verlusten.
- so viele Änderungen durchführen, wie es notwendig erscheint um aktuelle und zukünftig erwartete Anforderung abzudecken: hierdurch wird erreicht dass nicht bereits kurz nach Fertigstellung der Migration das neue System angepasst werden muss, und unter Umständen eine weitere Migration ansteht.
- so wenige Änderungen wie möglich durchzuführen, um den Umfang und das Risiko der Migration zu verringern: je komplexer eine Migration ist, desto höher ist die Fehlergefahr, die Komplexität einer Migration steigt mit der Anzahl der durchgeführten Änderungen.
- alten Code so wenig wie möglich abändern, um Risiken zu minimieren: solange der Code funktioniert und keine neue Funktionalität notwendig ist, sollte er übernommen werden wie er ist bzw. nur minimale Änderungen durchgeführt werden, da Änderungen zwangsweise auch Fehler in der Implementierung nach sich ziehen. Dieses Prinzip wird jedoch meist nicht angewendet, da das ganze System ohne Übernahme von Code neu entwickelt wird.
- alten Code soweit abändern, dass er die Migration unterstützt: wenn durch Änderungen mit vertretbarem Aufwand am Code die Migration vereinfacht wird, sollte dies gemacht werden.

- möglichst große Flexibilität einbauen, um zukünftige Änderungen zu erleichtern: beispielsweise durch die Kapselung der Funktionen und Bereitstellung eines APIs (Application Programming Interface) können zukünftige Entwicklungen und Anpassungen erleichtert werden.
- mögliche negative Auswirkungen der Änderungen minimieren: bei allen Änderungen am System sollte geprüft werden, ob diese Änderungen noch mit dem System verträglich sind, um hierdurch bereits frühzeitig Fehlentwicklungen vorzubeugen.
- den Nutzen moderner Technologien und Methoden maximieren: hierdurch wird zum einen eine zukünftige Anpassung leichter, zum anderen lassen sich Systemwerte, die zur Entscheidung für eine Migration, beispielsweise Performanz, Datendurchsatz, positiv beeinflussen.

2.2 „Chicken Little“ - Ansatz

Diese Migrationsstrategie besteht aus elf „kleinen Schritten“, die iterativ durchgeführt werden, wodurch die Migration überschaubar wird, da die eigentliche Migration in kleinere Teile aufgespalten wird (Prinzip „Divide & Conquer“). „Chicken Little“ bedeutet dennoch eine vollständige Neuentwicklung des Systems.

1. Analyse des Altsystems: Für eine erfolgreiche Migration ist es unabdingbar, zuerst die Funktionsweise des Altsystems zu verstehen. Hierbei hilft eine hoffentlich vorhandene Dokumentation, andernfalls Reverse Engineering.
2. Zerlegen des Altsystems: Das Altsystem muss insoweit geändert werden, dass definierte Schnittstellen zwischen den einzelnen Modulen und den Datenbankbackends bestehen.
3. Entwickeln der Benutzeroberfläche des Zielsystems.
4. Entwickeln der Zielanwendungen: Abwägung, ob die Funktionalität der Anwendung des Altsystems nachgebaut werden soll, oder denen der Altanwendung nur möglichst nahe kommen soll.
5. Entwickeln des Datenbankbackends: Hierbei werden die Ergebnisse der vorausgegangenen Schritte mit einbezogen, empfohlen wird die Entwicklung mit einer relationalen Datenbank auf Basis von SQL.
6. Installation der Zielumgebung: Aufbau einer Testumgebung und Testen dieser Umgebung.

7. Entwicklung und Installation von Gateways: Die Gateways sind dafür zuständig, die Daten aus dem Altsystem zu extrahieren und in das Zielsystem zu überführen.
8. Migration der Datenbank des Altsystems: Installation des neuen Datenbanksystems, anschliessend Migration der Daten zwischen Alt- und Zielsystem.
9. Migration der Altanwendungen: Nach und nach Austausch der einzelnen Module der Altanwendungen und deren Einbindung in das Gesamtsystem.
10. Migration der Benutzeroberfläche
11. Umschalten vom Altsystem auf das Zielsystem: Aktivieren des neuen Systems und Abschalten des alten Systems

2.2.1 Database First

Hierbei wird als erstes das Datenbanksystem auf ein modernes System migriert, bevor Anwendungen und Benutzeroberflächen migriert werden. Für den Zugriff der Komponenten des Altsystems auf das neue Datenbanksystem werden sog. Forward Gateways verwendet. Nach vollständiger Migration der Anwendungen und Oberflächen kann das Forward Gateway deaktiviert werden.

Der gravierende Vorteil dieser Methode ist, dass am Ende der Migration auf jeden Fall die entwickelte Anwendung und die Datenbank zusammenpassen, da die Anwendungsentwicklung erst beginnt, wenn die Migration der Datenbank abgeschlossen ist. Somit kann für Tests der noch zu entwickelnden Anwendung bereits die neue Datenbasis verwendet werden. Ebenfalls vorteilhaft ist, dass durch die Umstellung auf die neue Datenbank sofortige Verbesserungen im Bereich Reporting durch aktuelle Programmiersprachen erzielt werden können. Auch können die einzelnen Anwendungen anschliessend eine nach der anderen migriert werden, ohne die Funktion des Systems zu beeinträchtigen.

Hauptnachteil dieses Vorgehens ist, dass es nur auf Systeme anwendbar ist, die eine definierte Schnittstelle zum Datenbackend, also eine strikte Trennung von Anwendung und Datenbackend, aufweisen. Ausserdem muss vor Beginn der Migration die Datenbankstruktur entwickelt werden. Das zu entwickelnde Forward Gateway kann außerdem so kompliziert sein, dass es manchmal überhaupt nicht möglich ist, ein solches zu erstellen.

2.2.2 Database Last

Dieser Ansatz ist der Gegensatz zu Database First und kann ebenfalls nur auf Systeme mit definierter Datenbackendschnittstelle angewandt werden. Nach und

nach werden die Anwendungen des Altsystems migriert, für den gleichzeitigen Zugriff von Komponenten des Alt- und Neusystems auf den Datenbestand müssen alle Komponenten des Neusystems den Zugriff über Reverse Gateways abwickeln. Der letzte Schritt dieser Migrationsmethode ist die Migration des Datenbanksystems auf die neue Plattform. Wie bei Database First kann dieses Vorgehen nur bei Systemen mit definierter Datenschnittstelle angewendet werden. Ebenfalls kann das dann zu

2.3 „Cold Turkey“/Big Bang - Ansatz

Bei „Cold Turkey“ handelt es sich, wie bei „Chicken Little“ um eine komplette Neuentwicklung des Altsystems mit Hilfe moderner Entwicklungsmethoden. Hierbei wird parallel zum Altsystem das neue System entwickelt und getestet. Hat das neue System alle notwendigen Tests bestanden, wird in einem finalen Schritt, dem „Big Bang“, das Altsystem deaktiviert und das neue System übernimmt die Arbeit. Hierdurch bedingt ergeben sich aber hohe Risiken für eine funktionierende Migration:

- Eine vollständige Neuentwicklung benötigt Zeit. Mit der Zeit allerdings, die die Entwicklung benötigt, werden auch Änderungen am Altsystem durchgeführt, um die aktuellen Bedürfnisse des Unternehmens zu erfüllen. Hierdurch entsteht ein generelles Problem, da diese Änderungen am Altsystem auch in bereits fertiggestellte Teile des neuen System eingepflegt werden müssen. Dies ist fehleranfällig und kostenintensiv.
- Meist gibt es für die Altsysteme keine Dokumentation außer das System selbst, also beispielsweise den Sourcecode. Es besteht nunmehr das Problem, dass die ursprünglichen Entwickler des Systems meist nicht mehr verfügbar sind, und somit anhand des Sourcecodes das System und dessen Funktionsweise verstanden werden muss. Zudem bereiten bestimmte Programmier Techniken, wie selbstmodifizierender Code, Probleme bei der Migration.
- Oft existieren nicht dokumentierte Abhängigkeiten zwischen dem Altsystem und anderen Systemen. Diese Abhängigkeiten können im gesamten Spektrum von unkritischen Abhängigkeiten bis zu unternehmenskritischen Abhängigkeiten reichen. Im Entwicklungszyklus des Altsystems steigt die Anzahl der Abhängigkeiten immer weiter an und die Existenz dieser Abhängigkeiten ist, durch die meist fehlende Dokumentation, oft gar nicht bekannt.

- Die schiere Menge an Daten stellt ein weiteres Problem für diesen Ansatz dar. Selbst wenn das Zielsystem vollständig verfügbar ist, würde es in manchen Fällen Wochen dauern, um die Daten aus dem Altsystem in das neue System zu überführen. Während dieser Zeit wären keine Änderungen an den Daten möglich, und somit das System nicht nutzbar. Dies ist für kaum ein Unternehmen ein gangbarer Weg.
Auch wird bei einer Migration meist das Datenschema verändert, was während der Datenmigration ebenfalls berücksichtigt werden muss.
- Das Management solch großer Projekte ist extrem schwierig.
- Die oben aufgeführten Punkte führen dazu, dass die Roadmap für die Entwicklung kaum eingehalten werden kann, sich die Fertigstellung des Systems immer weiter verzögert.

2.4 Butterfly

Beim Ansatz der Butterfly-Migration (Abbildung 1) handelt es sich um eine Strategie, die, im Gegensatz zu „Chicken Little“ ohne den Einsatz von Gateways auskommt. Die Methode basiert auf einer initialen Migration der Daten-Backends: das Legacy-System bleibt betriebsbereit, während in einer Testumgebung bereits das neue System entwickelt und getestet werden kann, ohne den normalen Betrieb zu beeinflussen oder gar zu stören.

Wichtig für diese Technik der Migration ist die Grundannahme, dass es nur um die reine Datenmigration geht und eine Kooperation zwischen Alt- und Zielsystem nicht notwendig ist.

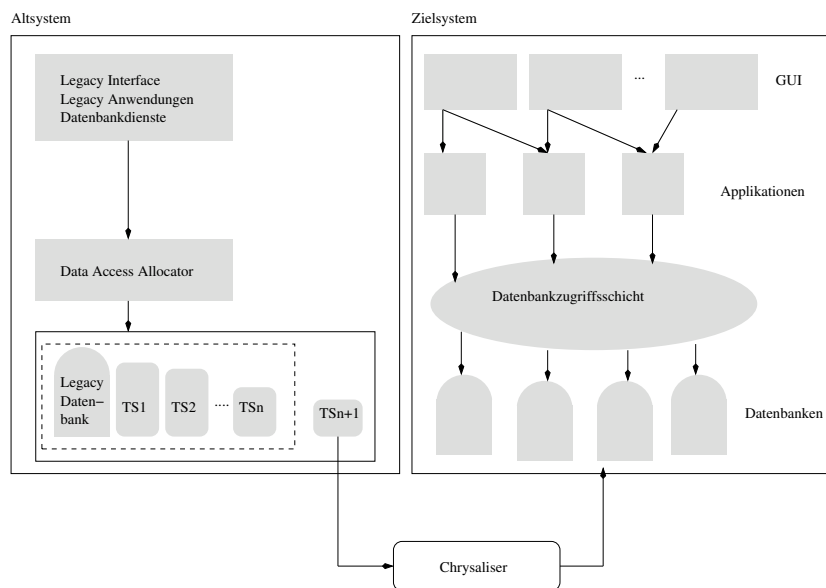


Abbildung 1: Butterfly-Migration

Zu Beginn des Migrationsprozesses werden neben der Datenbasis des Altsystems mehrere Temporärspeicher eingerichtet und die Datenbasis mit einem Schreibschutz versehen. Durch den Data Access Allocator werden die Zugriffe umgeleitet: noch nicht zugegriffene Information wird aus der Datenbasis geholt, Änderungen werden zu Beginn in den temporären Speicher TS_1 geschrieben. Falls geänderte Informationen abgerufen werden müssen, werden diese aus TS_1 geholt.

Anschliessend kann gefahrlos, also ohne Datenverlust und Einbuße an Servicequalität des Systems, der Datenbestand des Altsystems über den sog. „Chrysaliser“, eine Komponente, die die Daten vom Datenschema des Altsystems in das neue Datenschema überführt und im neuen Datenbackend ablegt, in das neue System überführt werden. Während dieser Migration werden alle Datenänderungen, wie oben beschrieben, nicht mehr im Legacy-Datenbackend gespeichert, sondern im Temporärspeicher TS_1 . Ist die Migration der Altdatenbank abgeschlossen, müssen auch noch die Informationen, die in der Zwischenzeit in TS_1 gespeichert worden sind, migriert werden. Hierzu wird TS_1 für Schreibzugriffe gesperrt und der neue Speicher TS_2 geöffnet. Änderungen am Datenbestand werden nun nicht mehr in TS_1 , sondern in TS_2 gespeichert. Jedesmal, wenn nun ein Temporärspeicher TS_n vom „Chrysaliser“ migriert wurde, wird der Speicher TS_{n+1} für Schreibzugriffe gesperrt, der Speicher TS_{n+2} für schreibende Zugriffe des Legacysystems geöffnet, und der Speicher TS_{n+1} an den „Chrysaliser“ übergeben. Kann der Inhalt eines Temporärspeichers schneller migriert werden, als der neue Speicher

vom Altsystem angelegt wird, dass also während der Migration eines Temporärspeichers TS_n keine Schreibzugriffe des Legacysystems stattfinden, wird die Größe des Temporärspeichers TS_{n+2} im nächsten Schritt verringert. Die Größe des Speichers hat für $n \rightarrow \infty$ mathematisch also einen Grenzwert, der gegen 0 konvergiert.

Während der gesamten Migration arbeitet das System ganz normal weiter, bis die Größe des letzten Temporärspeichers einen bestimmten Schwellenwert unterschreitet, so dass die Zeit, die die Migration dieses letzten Speichers benötigt, extrem kurz ist. Dann kann im letzten Schritt das Altsystem gestoppt werden, der letzte Temporärspeicher in das neue System überführt werden, und das neue System in Betrieb genommen werden, da nunmehr zwischen dem Datenbestand des Alt- und Neusystemes Konsistenz erreicht wurde.

Die Vorteile des Butterfly-Verfahrens liegen auf der Hand: Zum einen ist das komplette System, bis auf den Schritt des finalen Umschaltens auf das Neusystem, dessen Dauer durch die geschickte Wahl des Schwellenwertes weiter minimiert werden kann, zu jeder Zeit verfügbar. Ebenfalls kann zu jeder beliebigen Zeit vor dem Umschalten der Systeme die Migration abgebrochen werden, da die Migration solange umkehrbar ist, solange alle Daten über den Data Access Allocator gelaufen sind. Sollte die Migration abgebrochen werden müssen, müssen nur nacheinander die Temporepeicher beginnend bei TS_1 wieder in die Datenspeicher eingebunden werden.

Nachteil dieser Strategie ist, dass je nach Aktivität im Altsystem extrem viele Temporärspeicher notwendig sein können, die alle, um einen möglichen Abbruch der Migration zu ermöglichen, nicht beispielsweise im Round-Robin-Verfahren überschrieben werden dürfen. Es kann also im Extremfall während der Migration hoher Hardwareeinsatz für Speicherbackends erforderlich sein. Ein anderes Problem stellt die Entwicklung des Data Access Allocators dar: man spart sich, im Gegensatz zu „Chicken Little“ zwar die Entwicklung von Gateways zwischen den Systemen, der Data Access Allocator ist jedoch ebenfalls eine sehr komplexe Komponente, die hohen Entwicklungsaufwand erfordern kann.

2.5 Vergleich und Bewertung der Ansätze „Chicken Little“, „Cold Turkey“ und „Butterfly“

„Cold Turkey“ hat im Vergleich zu „Chicken Little“ deutliche Schwächen.

So ist etwa das Risiko, dass die Migration nicht funktioniert, bei Cold Turkey extrem hoch, da die Umschaltung aller Systemkomponenten zu einem einzigen Zeitpunkt geschieht. Sollte nur an einem Punkt ein Fehler auftreten, bricht das komplette System zusammen. Anders bei Chicken Little, hier ist das Risiko bei der Migration recht überschaubar, da nach jedem Schritt in der Migration getestet

werden kann, ob sich das System den Wünschen entsprechend verhält.

Ebenfalls besteht ein Unterschied beim Auftreten eines Fehlers. Bei „Cold Turkey“ schlägt im Fehlerfall das komplette Projekt fehl, und es ist nicht sofort ersichtlich, an welchem Punkt der Fehler aufgetreten ist. Bei „Chicken Little“ jedoch kann immer nur ein einzelner der iterativen Schritte fehlschlagen, eventuell sogar nur eine einzige Komponente, somit ist auch die Suche nach der Ursache des Fehlers einfacher durchzuführen, da klar ist, in welcher Komponente nachgebessert werden muss.

Extremer Nachteil von „Chicken Little“ ist die Notwendigkeit der Entwicklung von Gateways, um zwischen Alt- und Neusystem zu vermitteln. Diese Gateways können hochkomplex sein, teilweise ist es auch überhaupt nicht möglich, solche Gateways zu entwickeln.

Der Nutzen von „Cold Turkey“ ist sofort nach Fertigstellung sichtbar, jedoch möglicherweise nur von kurzer Dauer. Bei „Chicken Little“ steigt der Nutzen stetig mit der Erstellung des neuen Systems voran.

Beim „Butterfly“-Ansatz wird eine parallele Test- und Entwicklungsumgebung zum Legacysystem aufgebaut, auf dem vorab alle Schritte der Migration durchgeführt werden. Der einzige Interaktionspunkt zwischen den beiden Systemen ist die Datenbank, es sind anders als bei „Chicken Little“ keine komplexen Gateways notwendig, und durch die Taktik dieser Datenmigration kann ähnlich geringe Downtime wie bei „Chicken Little“ erreicht werden. Die Möglichkeit, auf die Gateways zu verzichten, erkaufte man sich jedoch durch die Notwendigkeit der Entwicklung des Data Access Allocators, der in der Entwicklung genauso komplex sein kann wie die bei „Chicken Little“ notwendigen Gateways. Ebenfalls ist die Annahme, dass zwischen Alt- und Neusystem keine Kooperation stattfindet, nicht realistisch.

3 Migration verschiedener Systemstrukturen mit „Chicken Little“

Im folgenden soll anhand verschiedener Systemstrukturen der Ablauf einer Migration mit Hilfe der „Chicken Little“-Strategie veranschaulicht werden. Bei „Cold Turkey“ und Butterfly ist der Aufbau des Altsystems irrelevant, da zwischen den beiden Systemen keine Kooperation stattfinden muss bis auf den finalen Schritt des Umschaltens vom Altsystem auf das Neusystem sowie die Datenübernahme. Bei diesen Strategien ist es somit egal, ob es sich beim Altsystem um ein aus Komponenten aufgebautes oder monolithisches System handelt. Einzig für „Chicken Little“ ergeben sich durch die Modularität der Migration interessante Aspekte. Nachdem anhand von 2.2.1 bzw. 2.2.2 die Position der Gateways in waagrechter

Ebene festgelegt werden kann, ist auch die Positionierung der Gateways in der senkrechten Ebene möglich, also ob die Gateways bereits auf der Datenzugriffsschicht implementiert werden können, oder die Implementierung nur durch den Aufbau einer neuen GUI-Schicht oberhalb der System-GUIs stattfinden kann.

3.1 Migration von vollständig zerlegbaren Systemen

Hierbei handelt es sich um Systeme, die vollständig aus voneinander schwach zusammenhängenden Modulen aufgebaut sind. Sie besitzen definierte Schnittstellen zwischen den einzelnen Modulen sowie zur Datenbank.

Hierbei gibt es verschiedene Vorgehensweisen:

- Vorwärtsmigration (Abbildung 2): Hierbei wird die „Database First“-Strategie aus 2.2.1 verwendet, es findet also eine initiale Migration der Datenbank und Implementierung eines Forward Gateways statt. Das Gateway ist nachträglich nur durch hohen Aufwand abänderbar, weshalb vor Erstellung des neuen Datenschemas eine genaue Planung stattfinden muss. Vorteile dieser Technik sind beispielsweise die Nutzung zusätzlicher Funktionen des neuen Datenbanksystems auch in Verbindung mit dem Altsystem.

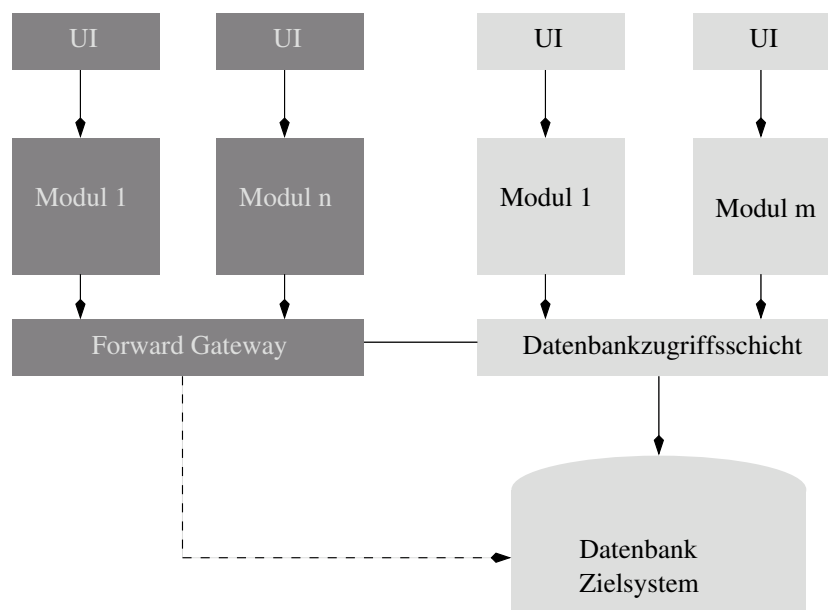


Abbildung 2: vollständig zerlegbare Systeme - Vorwärtsmigration

- Rückwärtsmigration (Abbildung 3): Hier wird die Strategie „Database Last“ aus 2.2.2 angewandt. Ein Nachteil ist hierbei, dass Funktionen des neuen

Datenbanksystems mühsam über das Reverse Gateway nachgebildet werden müssen, was sehr zeitaufwändig und kostenintensiv sein kann. Ein wichtiger Vorteil ist jedoch, dass das Altsystem bis zum endgültigen Abschalten ohne Modifikationen weitergenutzt werden kann

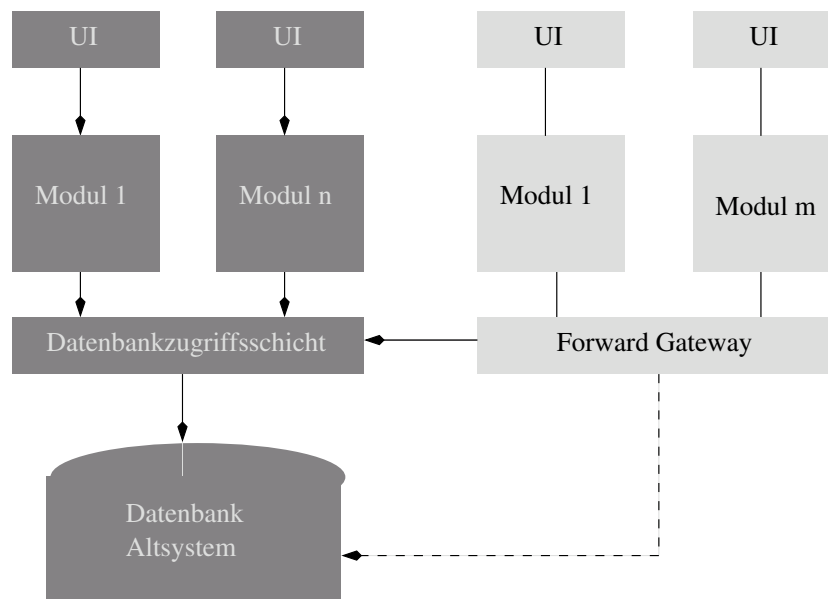


Abbildung 3: vollständig zerlegbare Systeme - Rückwärtsmigration

- Allgemeine Migration (Abbildung 4): Hierbei handelt es sich um eine Verschmelzung von Vorwärts- und Rückwärtsmigration. Über einen „Coordinator“ und eine „Mapping Table“ werden die entsprechenden Anfragen der Alt- und Neusysteme umgesetzt und an die entsprechenden Forward bzw. Reverse Gateways weitergeleitet, die dann ihrerseits die eigentliche Verbindung zur Datenbank herstellen. Vorteil dieser Methode ist, dass kein Ausfall des Benutzerbetriebs in irgendeiner Phase der Migration entsteht, diese Art also auf für unternehmenskritische Anwendungen verwendet werden kann. Auch werden hierbei die Vorteile der Vorwärts- und Rückwärtsmigration kombiniert, und deren jeweilige Nachteile mit Hilfe der anderen Methode größtenteils kompensiert.

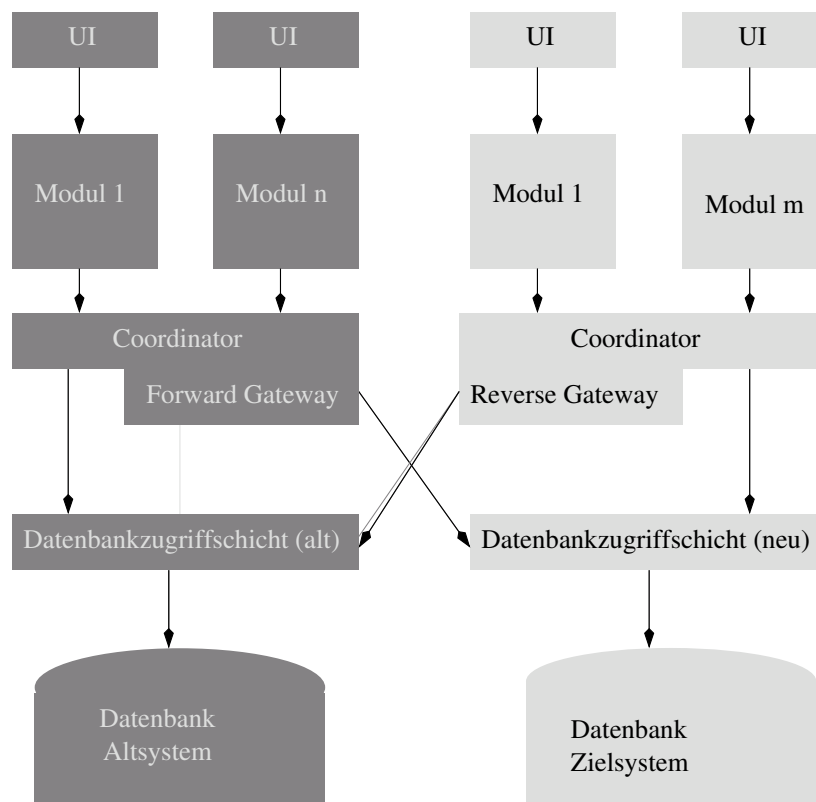


Abbildung 4: vollständig zerlegbare Systeme - allgemeine Migration

3.2 Migration von teilweise zerlegbaren Systemen

Bei dieser Art von Systemen ist die Modularisierung des Anwendungskerns nicht mehr möglich, es handelt sich um eine große Einheit. Einzig die Benutzerschnittstellen können noch als verschiedene Module betrachtet werden (s. Abbildung 5).

Bei der Durchführung der Migration wird dann der monolithische Anwendungskern im neuen System in Module aufgespalten. Diese Module beinhalten dann schon die Konnektoren zur neuen Datenbank. Wie bei der allgemeinen Migration vollständig zerlegbarer Systeme ist auch hier ein sog. „Coordinator“ und eine „Mapping Table“ nötig, um die Zugriffe auf Alt- und Neusystem durchzuführen und über die einzelnen Gateways zu leiten. Jedoch kann durch den monolithischen Anwendungskern der Coordinator nicht mehr erst auf Ebene der Datenbankzugriffsschicht erstellt werden, er muss zwischen dem UI und dem Anwendungskern des Altsystems implementiert werden. Der Coordinator des Altsystems reicht dann die Anfragen an das Altsystem weiter, zugleich verzweigt er die Anfragen über das Forward Gateway auf die neu erstellten Module des Anwen-

dungskerns des Neusystems; der Coordinator des Neusystems reicht die Anfragen weiter auf die Anwendungskernmodule des Neusystems, über das Reverse Gateway verzweigt er auf den monolithischen Kern des Altsystems.

Das größte Problem dieses Ansatzes ist durch die vorgefundenen Gegebenheiten bedingt: durch den monolithischen Anwendungskern ist die Erstellung des Coordinators extrem schwierig und aufwändig.

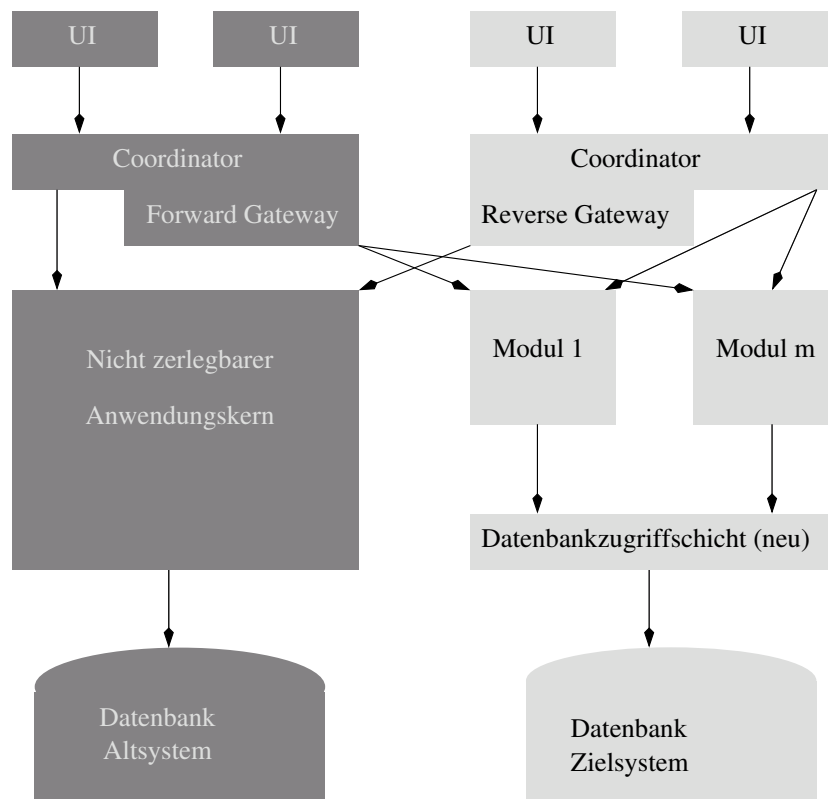


Abbildung 5: Migration teilweise zerlegbarer Systeme

3.3 Migration von monolithischen Systemen

Beim Altsystem handelt es sich um ein nicht modularisierbares System, bei dem intensive Abhängigkeiten bestehen. Die Möglichkeit eines Gateways bietet sich also nur oberhalb der Nutzungsschicht. Zugriffe von aussen werden über das Interface Gateway sowohl an die im Monolith integrierte Oberfläche als auch an die migrierten Komponenten des Neusystems übermittelt. Zum Erstellen der einzelnen Module ist es nötig, die Eigenschaften des Altsystems genau zu erfassen und zu dokumentieren. Die Module werden dann als funktionale Einheiten erstellt.

Über eine „Mapping Table“ wird ebenfalls die Datenbank schrittweise migriert werden, indem bereits implementierte Funktionseinheiten schon auf das neue Datenbankbackend zugreifen, noch nicht migrierte Einheiten noch das alte System verwenden. Somit kann gleichzeitig mit der schrittweisen Aufspaltung des Monoliths auf die zugehörige Datenbank migriert werden (s. Abbildung 6).

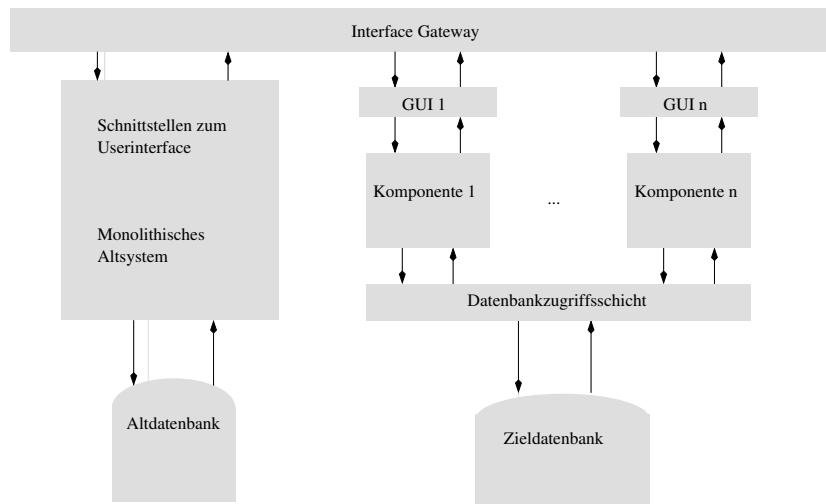


Abbildung 6: Migration eines monolithischen Systems

4 Datenmigration

Ein Problem, das bisher bei keinem der Ansätze berücksichtigt wurde, ist, dass sich nicht nur der Code des Systems bei der Migration ändert, sondern Daten ebenfalls einer Anpassung unterworfen sind. Hier ist es beispielsweise manchmal nötig, eine Konsolidierung von Werten vorzunehmen. So kann in der Altdatenbank ein Freitextfeld zur Eingabe des Titels einer Person zur Verfügung gestanden haben, in der neuen Datenbank jedoch soll der Titel aus einer vorgegebenen Liste ausgewählt werden. Es stellt sich also das Problem, ein Mapping zwischen existierenden und zukünftigen Daten zu finden und dieses umzusetzen. Am Beispiel des Titels erläutert bedeutet das, dass etwa in einem ersten Schritt alle voneinander verschiedenen Inhalte des Feldes „Titel“ analysiert werden, eine Konsolidierung der Schreibweise durchgeführt wird, da beispielsweise „Professor“ und „Prof.“ den gleichen Titel bezeichnen, sowie das anschließende Mapping auf den dann in der neuen Datenbank zu verwendenden Titel „Prof.“. Bei Kombinationen wie „Prof. Dr.“ und „Professor Dr.“ wird dies beliebig kompliziert. Ein weiteres Beispiel könnte sein, dass ein Teil der Kundennummer eines Kundenstammsatzes die

Postleitzahl des Firmensitzes ist. Da im Jahr 1990 das Postleitzahlensystem in Deutschland von 4- auf 5-stellige Zeichenketten geändert wurde, muss auch dieses Remapping konsistent durch die gesamte Datenbank neu gemacht werden, da etwa Bestellungen als Sekundärschlüssel die Kundennummer verwenden, um die Zuordnung zum jeweiligen Kunden zu ermöglichen. In solchen Fällen ist die Entwicklung von Gateways extrem kompliziert, da bei jedem Datenbankzugriff eine Transformation zwischen den Daten durchgeführt werden muss.

5 Beispiel einer einfachen Migration: Übergang von einer Windows-NT4-Domain auf eine Windows-2000/2003-Domain

Dieses Beispiel stellt eine Migration innerhalb einer größeren Gesamtstruktur dar, wie es beispielsweise in der Systemverwaltung am Lehrstuhl Prof. Broy auftritt. Die beschriebene Migration basiert auf von Microsoft dokumentierten „best practices“.

5.1 Warum und wann findet eine Migration statt

Für eine Migration existieren, je nach Standpunkt, verschiedene Gründe.

Für neue Hardwarekomponenten werden Systemkomponenten (in diesem Falle Treiber) oftmals nur noch für aktuelle Systeme angeboten und deren Funktion auch nur für diese Systeme garantiert. Beim Betrieb der Komponenten mit einer alten Version des Betriebssystems kann es dazu führen, dass die Komponente nur eingeschränkt nutzbar ist, überhaupt nicht funktioniert, oder im schlimmsten Fall andere Komponenten des Systems negativ beeinflusst.

Ein weiterer Punkt ist der geplante Einsatz neuer Softwareversionen, die ihren vollen Funktionsumfang erst mit einem aktuellen Betriebssystem entfalten können. Als Paradebeispiel ist hier der Microsoft Exchange Server 2003 zu nennen, der einige Funktionen des Programms erst zur Verfügung stellt, wenn er in einer Windows-2003-Directory-Struktur betrieben wird, d.h. alle Directory-Server der Domäne müssen unter Windows 2003 arbeiten.

Extrem wichtig in der jüngsten Geschichte ist der Aspekt Support und Sicherheit (Safety und Security). Microsoft verfolgt die Supportstrategie, nach einer gewissen Zeit die (kostenfreie) Unterstützung für Altsysteme einzustellen, wie beispielsweise für NT4. Wer danach noch Unterstützung für ein solches System möchte, muss einen kostenpflichtigen Supportvertrag abschliessen. Dies schließt auch die Versorgung mit Updates zu existierenden Sicherheitslücken mit ein. Ein Anwender ist also nach einer gewissen Zeit des Einsatzes gewissermaßen dazu

gezwungen, auf eine neue Version zu wechseln, möchte er sein System in einem sicheren Zustand halten.

Die Funktionalität, die neuere Versionen bereitstellen, kann ebenfalls einen gewichtigen Grund für eine Migration darstellen. Mit Windows 2000 wurde von Microsoft ein neues System für die Verwaltung eines Microsoft-Netzwerkes eingeführt (Nutzer-, System- und Dienstverwaltung), das Active Directory, das das alte Konzept der PDCs und BDCs (Primary Domain Controller, Backup Domain Controller) ersetzt. Bei Windows NT existierte nur eine einzige beschreibbare Version der Nutzer- und Systemdatenbank auf dem PDC, die BDCs kopierten sich die Daten. Beim Ausfall der PDCs konnten zwar noch Anmeldungen durchgeführt werden, Änderungen am Inhalt des Directories jedoch nicht. Dies erforderte eine unbedingte Wiederherstellung des ausgefallenen PDCs. Nicht so bei Active Directory: Jeder Domaincontroller hat Schreibzugriff auf das Directory, die Domaincontroller gleichen Änderungen untereinander durch laufende Replikation ab.

5.2 Zu berücksichtigende Probleme

Bei dieser Migration können einige Probleme auftauchen, die im Vorfeld abgeklärt werden müssen.

Es besteht die Möglichkeit, dass im Zielsystem durch das neue Betriebssystem nicht mehr unterstützte Hardware eingesetzt wird. Es existiert beispielsweise eine bestimmte Klasse von Intelprozessoren, auf denen Windows 2003 nicht ausgeführt werden kann bzw. bei denen auf Mehrprozessorsystemen nur ein Prozessor erkannt wird (MS KB 319091). Dies führt zur Entscheidung, ob nicht nur die Kosten für die Softwarelizenzen aufgebracht werden müssen, sondern auch die Beschaffung neuer Hardware nötig ist.

Ein weiteres Problem stellt die auf dem System verwendete Software dar. Hier spielt primär die Frage eine Rolle, ob die bisher eingesetzte Software auch unter dem neuen Betriebssystem funktioniert. Falls dies nicht der Fall sein sollte muss eruiert werden, ob der Softwarehersteller bereits eine aktualisierte Version der Software anbietet, die dann problemlos funktioniert. Ist auch dies nicht der Fall, muss Ersatzsoftware evaluiert werden, um die Funktionalität zu erhalten. Beispiel hierfür sind Sicherungsprogramme (am Lehrstuhl Prof. Broy wird IBM Tivoli in Verbindung mit dem Backup-Dienst am LRZ eingesetzt): beim Übergang von NT4 auf 2000/2003 hat sich die Ablage von Systeminformationen grundlegend geändert. Ebenfalls ist beim Umstieg von 2000 auf 2003 zu beachten, dass die Backupsoftware mit der Funktion der „Volume Shadow Copies“ (mehrere Versionen einer Datei können im Dateisystem abgelegt werden und durch den Nutzer selbst wieder restauriert werden) zurecht kommt, da ansonsten diese Funktion beim Einspielen eines Backups verloren geht, da nur die letzte Version gesichert wurde.

Ein weiteres Beispiel für ein Softwareproblem stellen Installationen des Microsoft SQL Servers 2000 pre-SP3 dar, die nach erfolgter Installation unter Windows 2003 nur noch lokal verwendbar und ansprechbar sind, bis ein entsprechendes Update auf mindestens SP3 eingespielt wurde.

5.3 Durchführung der Migration

Zu Beginn der Migrationsphase muss anhand der Dokumentation des Altsystems analysiert werden, welche Aufgaben das System erfüllt hat. Anhand dieser Analyse wird das Pflichtenheft für das neue System erstellt.

Anschliessend wird das Altsystem auf seine Funktionsfähigkeit getestet, wie es in seiner Dokumentation festgelegt wurde. Schlägt dies fehl, muss geklärt werden, ob dies ein für die Migration kritischer Fehler ist, der vor der Migration behoben werden muss, oder ob dieser Fehler erst nach der Migration gesucht werden soll, da er möglicherweise durch die Migration selbst gelöst wird.

Der nächste Schritt ist, dass vom NT4-System ein Backup erstellt wird, und dieses Backup anschliessend getestet wird. Wichtig ist hierbei, dass das Backup nicht nur auf Dateisystemebene erfolgt, sondern auch Eigenheiten wie die Windows-Registry und Domainkonfiguration berücksichtigt, so dass ein funktionierender Betriebszustand auch bei Fehlschlägen einer Migration wieder hergestellt werden kann.

Der nächste Schritt besteht darin, auf einem neuen/anderen Rechner ein Windows-2000/2003-Domaincontroller einzurichten und diesen zu testen.

Für die Migration der Benutzer- und Computerkonten wird von Microsoft ein spezielles Tool angeboten, um die Benutzer in das gerade neu installierte Directory zu integrieren.

Ist dies abgeschlossen, müssen alle benötigten Daten vom NT4-System auf das Windows-2000/2003-System übertragen werden. Hierbei handelt es sich beispielsweise um die User Homes und Projektverzeichnisse, falls der NT4-Server als Fileserver diente sowie der Roaming Profiles. Anschliessend muss die korrekte Funktion dieser Daten getestet werden.

Als nächstes wird nun ein weiterer Windows-2000/2003-Domaincontroller auf einem weiteren Rechner eingerichtet. Dies sollte nach Möglichkeit keiner der Rechner sein, auf denen im Augenblick noch die funktionierende Altdomain gehostet wird, da dann im Fehlerfall nicht mehr einfach bzw. überhaupt nicht auf die alte Domain zurückzuwechseln. Ist der zweite Domaincontroller installiert, führt dies zur initialen Replikation des Active Directory und anschliessendem Einbinden in die laufend durchgeführten Replikationszyklen. Anschliessen folgt wie immer das Testen des in diesem Schritt durchgeführten.

Als letzter Schritt wird noch gemäß Pflichtenheft die erforderliche Software installiert sowie die Systemdienste konfiguriert (DHCP, DNS, Cluster Services,

Volume Shadow Copy, ...) und finaler Test des neuen Systems.

6 Zusammenfassung

Zusammenfassend lässt sich sagen, dass es keine global geltende Empfehlung für eine Migrationsstrategie geben kann. Ausschlaggebend sind immer die lokalen Gegebenheiten.

Wird das Datenschema nur marginal bzw. überhaupt nicht geändert oder die Daten selbst nicht neu strukturiert, sondern nur auf ein anderes Datenbanksystem umgezogen, ist die Entwicklung entsprechender Gateways ungleich leichter. In solchen Fällen bietet es sich an, die Strategie „Chicken Little“ anzuwenden.

Sollte sich jedoch das Datenschema grundlegend ändern, ist es oft nicht mehr möglich, entsprechende Gateways zu entwickeln. Entsprechend kann dann oft auch kein entsprechender Data Access Allocator implementiert werden, was dazu führt, dass die einzige Strategie zur Migration die des „Big Bang“ ist.

7 Abbildungsverzeichnis

Abbildungsverzeichnis

1	Butterfly-Migration	11
2	vollständig zerlegbare Systeme - Vorwärtsmigration	14
3	vollständig zerlegbare Systeme - Rückwärtsmigration	15
4	vollständig zerlegbare Systeme - allgemeine Migration	16
5	Migration teilweise zerlegbarer Systeme	17
6	Migration eines monolithischen Systems	18

8 Literaturverzeichnis

Brodie, Stonebraker: Migrating Legacy Systems; Morgan Kaufmann Publishers Inc., 1995

Lesny, Marschall, Salzmann: Ablösung von Altsystemen; Lehrstuhl für Software & Systems Engineering - Prof. Dr. Manfred Broy, August 1999

Bisbal, et al.: A Survey of Research into Legacy System Migration

Ohlsson, Mayrhauser, McGuire, Wohlin: Code Decay Analysis of Legacy Software through Successive Releases

Eick, Graves, Karr, Marron, Mockus: Does Code Decay? Assessing the Evidence From Change Management Data; IEEE Transactions on Software Engineering,

Vol. 27, No. 1, Januar 2001

Bisbal, Lawless, Wu, Grimson: Legacy Information Systems: Issues and Directions; IEEE Software, issue September/October 1999

Technical Report - Software Reengineering Assessment Handbook Version 3.0 - US Department of Defense, 1997

<http://www.microsoft.com/windowsserver2003/upgrading/nt4/tooldocs/default.msp>