Befreiung aus proprietären Gefängnissen : Erfahrungen mit der Sanierung von Anwendersystemen

Autor(en): **Sneed, Harry M.**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **87 (1996)**

Heft 19

PDF erstellt am: **12.07.2024**

Persistenter Link: https://doi.org/10.5169/seals-902365

In den letzten zwei, drei Jahrzehnten haben die Unternehmen enorme Gelder in eigenentwickelte proprietäre Softwaresysteme investiert. Deren Entwickler sind heute samt ihrem Wissen über diese Anwendungen längst in Pension oder in andere Firmen abgewandert. Für jede Firma kommt der Tag, an dem sie sich die Frage stellen muss, ob und wie diese Software-Dinosaurier in die Welt der modernen Softwaresysteme migriert werden können oder ob von Grund auf neue (eigene oder fremde) Software entwickelt und installiert werden muss. Der Autor dieses Beitrages schildert seine Erfahrungen und Erkenntnisse, die er sich beim Reengineering von Mainframe-Software erworben hat.

Befreiung aus proprietären Gefängnissen

Erfahrungen mit der Sanierung von Anwendersystemen

■ Harry M. Sneed

Ein zentrales Informatikthema unserer Zeit ist der Übergang von der alten monolithischen, Mainframe-bezogenen und prozedural orientierten Welt in die neue Welt der verteilten, vernetzten und objektorientierten Systeme. Wie können die schwerfälligen, trägen und mit Altlasten beladenen Grossunternehmen einen so gewaltigen Sprung über die «Gletscherspalte» (Bild 1) bewältigen? Grundsätzlich gibt es drei Möglichkeiten:

1. Softwaresanierung
2. Kauf von Standardsoftware und
3. Neuentwicklung des Systems [1]

Die billigste Lösung?

Für Anwender, die vor der Wahl stehen, wie sie ihre Software restrukturieren sollen, ist der Kauf von Standardsoftware die billigste Alternative mit dem geringsten technischen Risiko. Sie bekommen eine schlüsselfertige Lösung von der Stange und müssen sie nur ihren Verhältnissen anpassen. Es war Barry Boehm, der sagte: «Die billigste Software ist die Software, die man nicht entwickeln muss.» Der Haken dabei ist allerdings der Anpassungsaufwand. Ist er klein, dann ist diese Alternative zweifelsohne die günstigste. Ist er aber gross, dann könnte diese Alternative die teuerste werden. Kritisch ist also die Schätzung des Anpassungsaufwandes. Im allgemeinen dürfte es am besten sein, wenn sich die Organisation der gekauften Software anpasst; aber auch das kostet Geld und Zeit, die man sehr wohl in die Kalkulation einbeziehen muss.

Die attraktivste Lösung?

Die attraktivste Alternative ist die Neuentwicklung des Systems. Denn hiermit glaubt der Anwender alle seine bisher zurückgehaltenen funktionalen Verbesserungen verwirklichen zu können. Der Haken hierbei ist das technische Risiko. Gerade weil so viele Änderungswünsche sich gesammelt haben, werden neu entwickelte Altsysteme besonders komplex. Die Wahrscheinlichkeit, überhaupt damit fertig zu werden, steht 50 zu 50.

Abgesehen vom Risiko können auch die Kosten einer Neuentwicklung schwer abgeschätzt werden, und zwar nicht wegen der Realisierung, die man heute mit modernen Schätzmethoden relativ gut kalkulieren

Adresse des Autors
Harry M. Sneed, SES Software-Engineering Service GmbH, D-85521 Ottobrunn/München

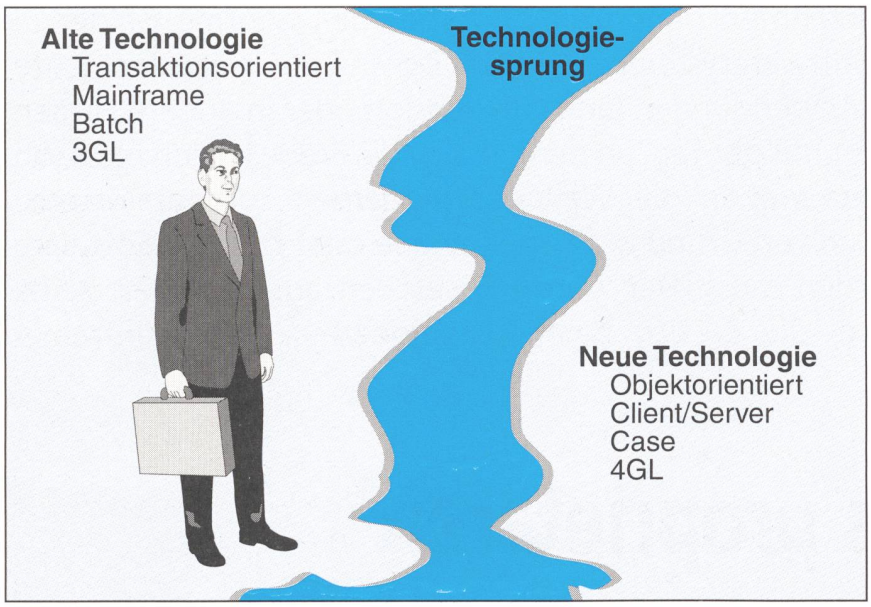

Bild 1 Sprung über die Gletscherspalte

kann, sondern wegen der Konzipierung. In grossen Organisationen wird es zunehmend schwieriger, einen Konsens über die Funktionalität neuer Softwaresysteme zu erreichen. Jeder will mitreden und seine eigenen Vorstellungen einbringen. Da in der heutigen Datenverarbeitung im Prinzip alles möglich ist, findet die Diskussion über den Umfang des neuen Vorhabens kein Ende – vor allem deshalb, weil jeder die Schwächen des Vorgängersystems kennt und jeder meint, das neue System müsse unermesslich viel besser sein. Die Neuentwicklung vorhandener Softwaresysteme hat also dort die besten Chancen, wo die zusätzliche Funktionalität eingegrenzt werden kann.

Mit den beiden Alternativen – Standardsoftware und Neuentwicklung – hat der Anwender die Wahl zwischen geringem technischem Risiko und abschätzbaren Kosten, verbunden mit organisatorischen Anpassungen, auf der einen Seite und hohem technischem Risiko mit schwer abschätzbaren Kosten, aber ohne organisatorische Anpassung auf der anderen Seite.

Die Kompromisslösung

Die Alternative «Sanierung» ist ein Kompromiss zwischen den beiden ersten Lösungen. Zum einen ist das technische Risiko geringer und das Ausmass der Kosten leichter abschätzbar. Zum anderen ist aber der Nutzen viel geringer. Es gibt keine neue Funktionalität, der betriebswirtschaftliche Nutzwert der Software bleibt konstant. Einzig der technische Nutzwert steigt. Das Ziel einer Sanierung ist die technische Überholung eines Systems. Eine Sanierung kann nur deshalb so billig sein, weil die Funktionalität eingefroren wird. Die funktionale Spezifikation für eine Sanierung ist die Software selbst – das heisst die Programme und Datenstrukturen – in ihrem momentanen Zustand. Die technische Spezifikation bildet die Transformationsregeln für die Restrukturierung, Bereinigung und Konvertierung der Programme, Masken und Datenstrukturen. Es lässt sich an der Software vieles ändern – sie kann von einer Sprache in die andere übersetzt werden, sie kann eine neue Datenbankschnittstelle erhalten, sie kann sogar von einer Host- in eine Client-Server-Architektur versetzt werden; nur eines ändert sich nicht, die Funktionalität. Denn nur so ist es möglich, die funktionale Korrektheit der sanierten Version gegen die Daten der alten Version zu bestätigen. Wer die Funktionalität ändert, muss neue Testdaten schaffen, eine neue Spezifikation erstellen, und bereits ist man in die Nähe einer Neuentwicklung gerückt.

Trennung von technischer und funktionaler Sanierung

Alle Erfahrungen mit Sanierungsprojekten haben gezeigt, dass der Erfolg nur dann gesichert ist, wenn zwischen der technischen und der funktionalen Sanierung getrennt wird. Zunächst gilt es, die Software in einen anderen technischen Zustand zu bringen; anschliessend kann die Software in eine andere technische Umgebung migriert werden, zum Beispiel von einer Netzwerkdatenbank in eine relationale Datenbank oder von einer Host- in eine Client-Server-Architektur. Erst wenn die Software in ihrem neuen technischen Zustand und in der neuen technischen Umgebung bestätigt worden ist, kann das zweite Projekt zur Änderung beziehungsweise Erweiterung der Funktionalität gestartet werden. Ein Sanierungsprojekt ist deshalb oft eine lange Kette wohldefinierter Zwischenschritte, wobei die Software nach jedem Schritt in einem produktionsreifen Zustand zu sein hat. Deshalb bedürfen Sanierungsprojekte einer sorgfältigen Planung und einer strengen Überwachung.

In den letzten Jahren ist der Autor dieses Beitrages an einigen Software-Sanierungsprojekten massgeblich beteiligt gewesen: zwei bei der Schweizerischen Bankgesellschaft (SBG), eines bei der Schweizerischen Kreditanstalt (SKA), eines für ABB-Henschel und eines für die Wella AG.

Termin- und kostengerecht

Das erste Projekt bei der SBG hatte das Ziel, 8 Bankapplikationen mit 27 Assembler-Programmen, 169 Cobol-

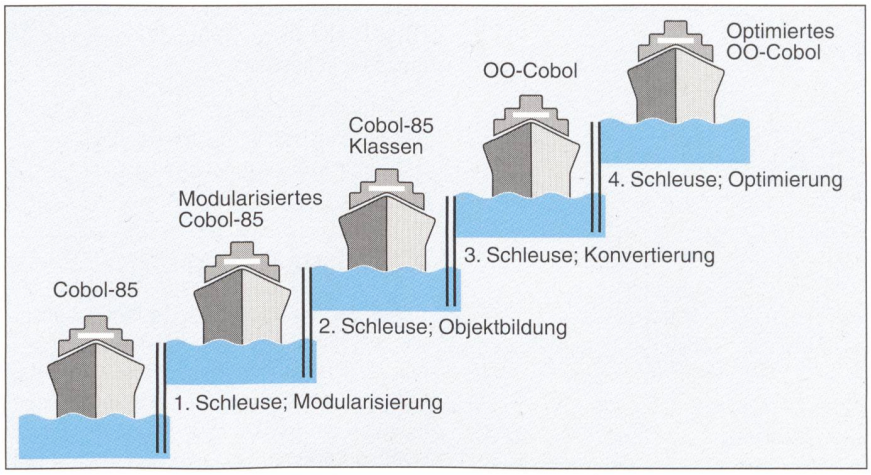

Bild 2 Evolution der Datenbanktechnik

68-Programmen, 123 Assembler-Subroutinen, 519 Dateien, 108 Dienstprogrammen und 8 Zugriffsmodulen – insgesamt 387 000 Lines of Code in Cobol-74/Delta/JSP mit einer Codasyl-Datenbank – auf einen anderen Rechner zu migrieren (Bilder 2 und 3). Der Assemblercode musste in Cobol übersetzt werden. Die Utility-Programme wurden entweder in Cobol oder in die Kommandosprache des Zielrechners übersetzt. Die Zugriffsroutinen wurden neu geschrieben. Die Dateien wurden entweder in VSAM- oder in Codasyl-Datenbanken migriert. Schliesslich wurden die Cobol-Programme restrukturiert. Dieses Projekt wurde zwei Monate vor dem Endtermin, nach genau einem Jahr und genau zu den geschätzten Kosten erfolgreich abgeschlossen [2]. In einem Folgeprojekt kam eine weitere Applikation mit nochmals zehn Cobol-Programmen und sieben grossen Assembler-Programmen dazu. Auch dieses Projekt wurde termin- und kostengerecht abgeschlossen.

Das zweite Projekt bei der SBG hatte das Ziel, Cobol-74/Delta/JSP-Programme in Cobol-85 und Codasyl-Datenbanken in relationale Datenbanken zu migrieren. Insgesamt waren 18 Datenbanken und 66 Programme betroffen. Auch hier konnte die Arbeit mit Hilfe von geeigneten Werkzeugen innerhalb der geplanten sechs Monate abgeschlossen werden.

Schrittweise, aber sicher

Das dritte Projekt für die ABB-Henschel hatte das Ziel, 20 Cobol-74-Programme, die zur Verkabelung von Lokomotiven dienten, von der IBM-Mainframe auf einen DEC-VAX-Rechner zu migrieren. Um dieses Ziel zu erreichen, mussten die 20 Programme bereinigt und nach Cobol-85 konvertiert werden. Dies erforderte viele Zwischenstufen. Zunächst wurden die Programme restrukturiert. Die neuen restrukturierten Versionen gingen zurück in die Produktion auf den IBM-Computer.

Danach wurden die Programme in Cobol-2 übersetzt und wieder auf die IBM-Mainframe in die Produktion gegeben. Schliesslich wurden die restrukturierten Cobol-2-Programme auf der DEC-Anlage übersetzt und getestet. Jetzt ist geplant, die Dateizugriffe durch eine relationale Datenbankschnittstelle zu ersetzen. Das ABB-Henschel-Projekt ist ein gutes Beispiel für eine schrittweise Migration.

Das vierte Projekt für die Wella AG hatte als Ziel, Unisys A12 Linc und Cobol-74-Programme sowie Unisys-DMS-II-Datenbanken und -Dateien auf ein IBM-AS/400-System zu migrieren. Auch in diesem Fall mussten die Programme restrukturiert und bereinigt werden. Die 35 Linc-Programme – eine 4GL von Unisys – wurden in 101-Online-Cobol-Module transformiert. Die 80 Cobol-74-Programme wurden in 80 strukturierte Cobol-85-Programme konvertiert. Obwohl dieses Projekt eine zweimonatige Verzögerung hatte, war das Ergebnis ein Erfolg. Die Programme liefen mit geringen Anlaufproblemen in der neuen AS/400-Umgebung. Die Verzögerung des Projekts ist darauf zurückzuführen, dass die Werkzeuge für dieses Projekt in dem Projekt selbst entwickelt werden mussten, das heisst es gab ein Entwicklungsprojekt in dem Sanierungsprojekt, und Entwicklungsprojekte, vor allem die Entwicklung von Softwarewerkzeugen, sind schwer kalkulierbar.

Das fünfte Projekt – für die SKA – hatte das Ziel, CICS/DLI/Assembler-Programme in Cobol-2 zu migrieren. Dieses Projekt ist nach sechs Monaten erfolgreich beendet worden. Auch hier waren mehrere Zwischenstufen vorgesehen. Erst wurde der Assembler-Code durch das Tool ASM-Recon in Cobol-74-Code übersetzt. In der zweiten Stufe wurde der Cobol-74-Code durch das Tool Cobol-Recon-74 restrukturiert. In der dritten Stufe wurde der Cobol-74-Code durch das Tool Cobol-Recon-85 in Cobol-85 umgesetzt. Nach jeder Stufe wurden die Programme daraufhin getestet, ob sie noch mit den ursprünglichen Assembler-Programmen funktional äquivalent waren.

Forderungen an Sanierungsprojekte

Folgende Lehre kann aus diesen und anderen Sanierungsprojekten gezogen werden: Die Aussichten auf Erfolg sind in Sanierungsprojekten viel grösser als in Entwicklungsprojekten, vorausgesetzt, die Ziele bleiben bescheiden.

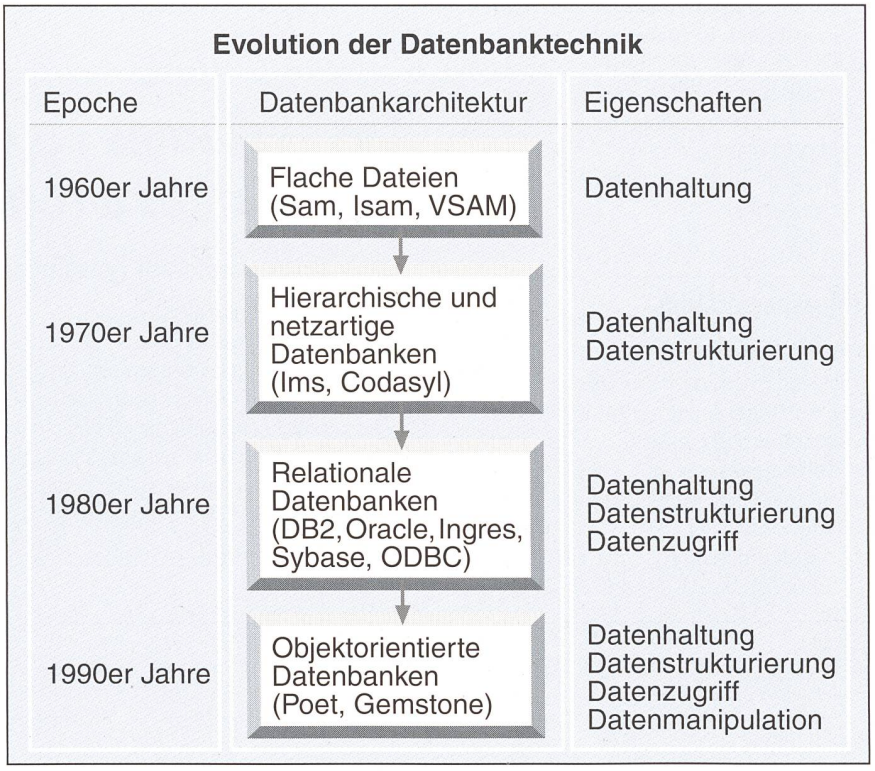

Bild 3 Cobol-Upgrading

1. Die Reengineering-Werkzeuge müssen vor dem Beginn des Sanierungsprojektes fertig ausgetestet sein.
2. Die Sanierung muss stufenweise erfolgen, wobei jede Stufe validiert werden muss.
3. Die technische Sanierung muss von der funktionalen Sanierung getrennt werden; erst wird technisch saniert, dann wird funktional erweitert.
4. Die Sanierungsarbeit muss in wohldefinierten Metriken, zum Beispiel Datenelementen, Modulen, Anweisungen und Codezeilen, messbar sein.
5. Die Testdaten müssen schon in der alten Umgebung aufgebaut, auf ihren Testdeckungseffekt geprüft und dann erst in die neue Umgebung übertragen werden.

6. Die sanierten Programme müssen unbedingt in bezug auf ihre Testüberdeckung gemessen werden.
7. Die Testpfade der alten Programme müssen mit den Testpfaden der sanierten Programme und die Testergebnisse der alten Programme mit jenen der sanierten Programme verglichen werden, um funktionale Äquivalenz festzustellen.
8. Der Regressionstest ist der Hauptressourcenfresser, die Transformation des Codes und der Daten muss weitgehend automatisiert werden.
9. Die Kosten der Sanierungsprojekte dürfen einen Drittel der Neuentwicklungskosten nicht überschreiten.
10. Die Qualität der Software darf nicht das Hauptziel sein. Das Hauptziel ist die neue Umgebung, denn eine echte Qualitätssteigerung ist schwer nachweisbar und vor allem schwer zu erreichen.

Schlussbemerkung

Zum Schluss ist zu Punkt 10 folgendes zu sagen. Ursprünglich wurde propagiert, dass Softwaresanierung beziehungsweise -restrukturierung ein Mittel zur Steigerung der Softwarequalität sei [3]. Leider konnte dieser Anspruch niemals erfüllt werden, weil niemand in der Lage ist, Softwarequalität zu definieren. Es wird sich deshalb niemand einzig um der Qualität willen auf ein Sanierungsprojekt einlassen. Es geht in jedem Fall um andere Ziele. Man will die Umgebung der Software auswechseln, oder man will von einer veralteten in eine moderne Umgebung umsiedeln. Ob die Software danach besser wird, ist ein sekundäres Ziel. Es ist schön, gut strukturierte oder gar objektorientierte Programme zu haben, aber dies ist nicht das entscheidende Kriterium. Wesentlich für die Entscheidungsträger ist die Tatsache, dass ihre Programme in einer leicht portierbaren Sprache, in einer offenen Umgebung mit flexiblen Datenbankschnittstellen laufen und dass ihre Datenstrukturen in einer normierten Form mit einer normierten Notation gespeichert sind. Softwaresanierung ist ein erprobtes Mittel, um sich aus den proprietären Gefängnissen der Hersteller zu befreien.

Literatur

[1] *H. Sneed:* Economics of Software Reengineering in Journal of Software Maintenance, 3(1991)3.

[2] *H. Sneed:* Bank Application Reengineering & Conversion at the Union Bank of Switzerland. Proc. of Int. Conference on Software Maintenance, Sorrento, Italy, Oct. 1991.

[3] *H. Sneed:* Software-Sanierung, Rudolf-Müller-Verlag, Köln, 1991.

Les systèmes «propriétaires»: des chaînes à rompre

Expériences faites dans l'assainissement des systèmes d'utilisateur

Ces vingt à trente dernières années, les entreprises ont investi des sommes énormes dans le développement de leurs propres systèmes dits «propriétaires». Aujourd'hui, il y a longtemps que les concepteurs de ces systèmes ont changé d'entreprise ou pris leur retraite. Le jour viendra pour chaque société de se demander quand son dinosaure logiciel pourra enfin migrer vers le monde des systèmes logiciels modernes ou s'il faut développer et installer un nouveau logiciel (développé par l'utilisateur ou acheté à l'extérieur). L'auteur de l'article fait part de ses expériences et des connaissances acquises dans le redéveloppement de logiciel pour Mainframe.