it 1/2009

Was geschrieben ist, ist geschrieben – Legacy CICS-Anwendungen im neuen Gewand

What is written, is written – Modern Legacy CICS-Applications

Fred Stefan, Paul Herrmann, Universität Leipzig, Wilhelm G. Spruth, Eberhard-Karls-Universität Tübingen

Zusammenfassung Integrationsprojekte sind ein wichtiger Bestandteil einer anpassungsfähigen Informationstechnologie und für einen flexiblen Geschäftsbetrieb unumgänglich. Gerade im Großrechnerbereich besteht ein wachsender Bedarf an Modernisierungsmöglichkeiten für Altanwendungen. Dieser Artikel beleuchtet einen von IBM verfolgten Ansatz zur Modernisierung von Legacy-Anwendungen und diskutiert die nötigen Schritte, um Altanwendungen auf den neusten Stand zu bringen. Das Service Flow Feature (SFF) bietet einen interessanten Ansatz zur Modernisierung von Legacy CICS-Anwendungen (CICS: Customer Information Control System), weil bereits laufende Anwendungen weiterverwendet und in vorhandene Architekturen integriert werden können. Zudem werden bessere Modernisierungsmöglichkeiten bereitgestellt als lediglich die Anpassung der Oberfläche. Hinzu kommt, dass mehrere CICS-Anwendungen aggregiert werden können und die damit erzeugten Geschäftsprozesse über offene Schnittstellen erreichbar sind. Gleichzeitig findet eine Entkopplung der Präsentationslogik von der Anwendungslogik statt. Letzteres bietet dem Entwickler die Möglichkeit, modernere Methoden der Präsentationslogik wie z. B. mittels JSP zu implementieren. Über den Einsatz des CICS SFF in der Entwicklungsumgebung RDz besteht für Firmen kurz- bis mittelfristig die Chance, ihr Reservoire an Legacy CICS-Anwendungen mit relativ wenig Aufwand den modernen IT-Anforderungen anzupassen. ►►►

Summary Integration projects are importent elements of a flexible information technology and essential for a flexible business concern. Current discussions show that even in the field of mainframe computing, an increasing need of modernisation possibilities for Legacy applications exist. This article highlights an IBM pursued method to modernize such applications and discusses the necessary steps to make them up-to-date again. The Service Flow Feature delivers an interesting modernization method for Legacy CICS (Customer Information Control System) applications because it reuses and integrates existing applications in available architectures. Furthermore it delivers more possibilities than only modernizing the user interface. In addition, it can aggregate multiple CICS applications, which forms business processes, that are accessible via open interfaces. At the same time, an uncoupling of the presentation logic from the application logic takes place.

KEYWORDS D.2.7 [Software: Software Engineering: Distribution, Maintenance, and Enhancement]; D.2.11 [Software: Software Engineering: Software Architectures]; D.2.13 [Software: Software Engineering: Reusable Software] Integration, z/OS, CICS, Legacy Anwendungen, Modernisierung, Service-orientierte Architektur / integration, legacy applications, modernisation, service-oriented architecture

1 Einleitung

In vielen Firmen sind Anwendungen im täglichen Einsatz, die bereits vor vielen Jahren entwickelt wurden und auf großen Transaktionsservern laufen. Um den modernen Geschäftsanforderungen schnell und flexibel gerecht werden zu können, müssen diese Anwendungen erweitert und auf den neusten Stand gebracht werden.

Das Ziel der meisten Firmen ist bei der Modernisierung der bestehenden Anwendungen nicht

die Neuentwicklung oder Umgestaltung, sondern deren Integration in eine anpassungsfähige Informationstechnologie. Schließlich steuern diese, meist in Cobol geschriebenen Anwendungen noch immer einen Großteil der Kernprozesse vieler Unternehmen. Dafür werden Werkzeuge benötigt, die die oft über viele Jahre gewachsenen Anwendungen weiterverwenden und mit anderen Funktionen zu einem Prozess integrieren, der später auch Bestandteil einer Service-orientierten Architektur (SOA) werden kann. Die unter großem Aufwand in der Vergangenheit entwickelten qualitativ hochwertigen Funktionen sind oft nicht flexibel und nur unter großer Mühe in eine SOA-Architektur zu integrieren. Sie wurden meist mit viel Redundanz rein nach praktischen Gesichtspunkten implementiert.

Der Fokus dieses Artikels liegt auf der Modernisierung von Anwendungen des Transaktionsmonitors CICS (*Customer Information Control System*), bei welcher aus Legacy-Anwendungen ein SOA-fähiger Geschäftsprozess modelliert wird. CICS ist ein Transaktionsmonitor, der unter den IBM/390-Großrechner-Betriebssystemen z/OS (MVS) und VSE verfügbar ist. CICS-Anwendungen sind in der Regel COBOL Software-Monolithen, die zwar geschäftskritische Prozesse unterstützen und den Ruf hoher Verfügbarkeit und Performance besitzen, jedoch scheinbar nur schwer änder-, erweiter- und integrierbar sind. Auf eine saubere Trennung von Präsentations-, Geschäfts- und Datenbank-Logik, die die modernen Paradigmen der Informatik fordern, wurde in der Regel kein Wert gelegt.

Hinzu kommt, dass solche Anwendungen aufgrund ihrer nicht zeitgemäßen 3270-Darstellung keine, beziehungsweise nur eingeschränkte GUI-Funktionalitäten aufweisen. Sie bieten darüber hinaus keine Möglichkeit, andere CICS-Anwendungen automatisch und parallel aufzurufen und zu nutzen [2]. Das Problem besteht darin, dass nie ein Datenaustausch der Transaktionen vorgesehen war.

Eine Möglichkeit der Wiederverwendung von Legacy CICS-Anwendungen wurde am Institut für Informatik an der Universität Leipzig untersucht [9]. Hierfür wird der IBM Rational Developer for System z (RDz) eingesetzt. Diese Software dient speziell der plattformübergreifenden Entwicklung konventioneller Mainframeanwendungen.

2 Der Rational Developer for System z (RDz)

RDz stellt eine auf Eclipse basierende moderne Entwicklungsumgebung dar, die unter anderem ein universelles Werkzeug für die Anwendungsentwicklung auf Mainframes implementiert. RDz wird weltweit von Unternehmen in der Entwicklung Internet-fähiger Anwendungen eingesetzt.

Die RDz-Architektur sieht eine Windows-Komponente vor, die entweder auf einem virtuellen Windows-Server oder direkt auf dem Client des Nutzers installiert wird. Diese Komponente bildet die eigentliche RDz-Entwicklungsumgebung, die einerseits lokal zur Anwendungsentwicklung und zu Testzwecken und andererseits zum finalen Einsatz auf dem Host genutzt werden kann.

Das CICS Service Flow Feature (SFF) ist eine Komponente des RDz und ermöglicht eine Überführung der Legacy-Anwendungen in eine SOA.

3 Modernisierung mit dem Service Flow Feature

Für eine Weiterverwendung von Legacy CICS-Anwendungen und gleichzeitiger Steigerung ihrer Agilität, Flexibilität und Benutzbarkeit, sind drei Ansätze denkbar:

- *Modernisierung der Endbenutzer-Oberfläche:* Anhand neuer Technologien wie Java Server Pages (JSP) [3] oder unter Verwendung der IBM Rational Host Access Transformation Services (HATS) [5] wird die eingeschränkte Präsentationslogik des Basic Mapping Support Subsystems (BMS[1]) ersetzt und mit neuen Funktionalitäten versehen. Auf diese Weise können Hostterminal-basierende Anwendungen ohne Änderungen Browser-fähig gemacht werden.
- *Modernisierung von Anwendungskomponenten:* Mit Java und CICS lassen sich bestehende Anwendungen schrittweise modernisieren. Bei diesem Prozess werden Anwendungskomponenten (z. B. CICS-COBOL-Anwendungen) sukzessive durch Java-Komponenten [7] ersetzt.
- *Modernisierung der Anwendungskommunikation:* Bei diesem Ansatz können bereits vorhandene Anwendungskomponenten weiterverwendet werden, während es gleichzeitig möglich ist, neue Funktionalitäten in den bestehenden Geschäftsprozess zu integrieren. Dieses Verfahren wird durch das im RDz integrierte CICS SFF unterstützt [8]. Es ermöglicht die Aggregation mehrerer CICS-Anwendungen zu einem Geschäftsprozess, der dann hochgradig für den CICS Transaction Server optimiert ist. Das CICS SFF bietet dabei die Möglichkeit, eine Reihe von Interaktionen zwischen den CICS-Anwendungen zu automatisieren.

Letzterer Ansatz zur Anwendungsmodernisierung wird nachfolgend näher beschrieben. Das CICS SFF ist ein zusätzliches Element des CICS Transaction Server (CICS TS[2]) seit Version 3 und in zwei Teile

[1] Als ein Bestandteil von CICS erzeugt und verwaltet BMS sogenannte „Maps“, die auch als „Screens“ bezeichnet werden. Jede Map beschreibt das Aussehen eines Bildschirm-Fensters des Klienten und definiert dabei die Anordnung und die Art (Attribut) von Feldern. Diese Felder werden zur Laufzeit mit statischen oder dynamischen Inhalten beschrieben.

[2] Der CICS TS ist ein Subsystem vom CICS und sorgt für die Datenintegrität.

gegliedert: den Service Flow Modeler (SFM) und die Service Flow Runtime (SFR).

Das CICS SFF bietet eine integrierte grafische Entwicklungsumgebung an, um aus verschiedensten CICS-Anwendungen einen Flow zu erstellen, dazu einen Generator, der eine CICS-TS-optimierte Laufzeitkomponente aus den aggregierten CICS-Anwendungen erstellt, und Adapter, welche die CICS-TS-Umgebung so erweitern, dass die erzeugte Laufzeitkomponente ausgeführt werden kann.

Während der gesamten Entwicklung mit dem SFM benötigt der Entwickler kein zusätzliches Wissen über die Implementation der CICS-Anwendungen. Um die Entwicklungsarbeit zu erleichtern, können BMS-Maps, also die 3270-Bildschirmdefinitionen, direkt in den SFM importiert werden und als Erweiterung dienen.

Die SFR implementiert eine Ausführungsumgebung, um die generierten Geschäftsprozesse ausführen zu können, während der SFM den RDz mit Entwicklungswerkzeugen ergänzt, die für die Geschäftsprozessentwicklung benötigt werden. SFR-Adapter stellen den Zugriff auf existierende CICS-Anwendungen sicher. Aufgrund dieser nicht-invasiven Methode sind keine Änderungen der CICS-Applikationen nötig.

Der SFM ist eine Zusammenstellung von Werkzeugen in der Entwicklungsumgebung vom RDz für die Aggregation neuer Geschäftsprozesse und Flüsse aus bestehenden Anwendungen und deren Interfaces. Sie unterstützen die Programmierung von Bildschirmmasken und stellen SOA-konforme Schnittstellen bereit. Der RDz ermöglicht die Daten-Transformation zwischen den einzelnen CICS-Anwendungen, wobei die neuen Geschäftsprozesse zum Schluss als Webservice bereitgestellt werden.

Am Ende des Modellierungsprozesses wird anhand von SFF-Generatoren der Programmquellcode erzeugt, der das Verhalten des modellierten Flows umsetzt. Zum Schluss werden die Laufzeit-Eigenschaften für die jeweilige CICS-Konfiguration definiert, alle Job Control Language (JCL) Skripte[3] für das Deployment erzeugt und eine ausführbare Web Service Description Language (WSDL)-Datei des Geschäftsprozesses generiert. Für den Entwicklungsprozess des Flows im SFM gibt es verschiedene Werkzeuge zum Importieren und Editieren von CICS-Anwendungen und für das Erzeugen der Laufzeitkomponenten. Der Ablauf ist folgendergestalt (vgl. Bild 1):

(1) Importieren und Editieren vorhandener Transaktionen anhand der integrierten Import-Werkzeuge und Editoren.
(2) Modellierung des Service Flows (z. B. Aggregation von anderen CICS-Anwendungen, Variablen-Mappings).
(3) Anhand von Template-Dateien, die systemspezifische Parameter beinhalten, erstellen Generatoren die Laufzeitkomponenten. Die Template-Dateien werden bei der SFR-Installation automatisch erstellt [4].

Ein Beispiel eines Flows, in dem zwei Eingabemasken eines Katalogs (3) und (4) durchlaufen werden, zeigt Bild 2. Die Pfeile zwischen den einzelnen Symbolen zeigen die Durchlaufrichtung des Flows [1]. Der Flow beginnt mit dem Einsprungknoten (1). Danach folgt der Knoten (2), welcher die Eingaben in den BMS-Screens (3) und (4) verarbeitet. Der Knoten (5) extrahiert die vorgenommenen Eingaben und kann diese dann anderen Knoten und Flows zur Verfügung stellen. Der Flow endet mit dem Endknoten (2) nach abgeschlossener Bestellung. Durch einfaches Hinzufügen

[3] JCL ist die Steuersprache für Stapelverarbeitungen in einem Großrechnerumfeld und hat seinen Ursprung im Lochkarten-Zeitalter. Aufgabe von JCL-Skripten ist es, die auszuführenden Programme, deren Reihenfolge, sowie eine Laufzeitumgebung (Verbindung zu physischer Hardware, E/A und Dateien) für den Host zu definieren.

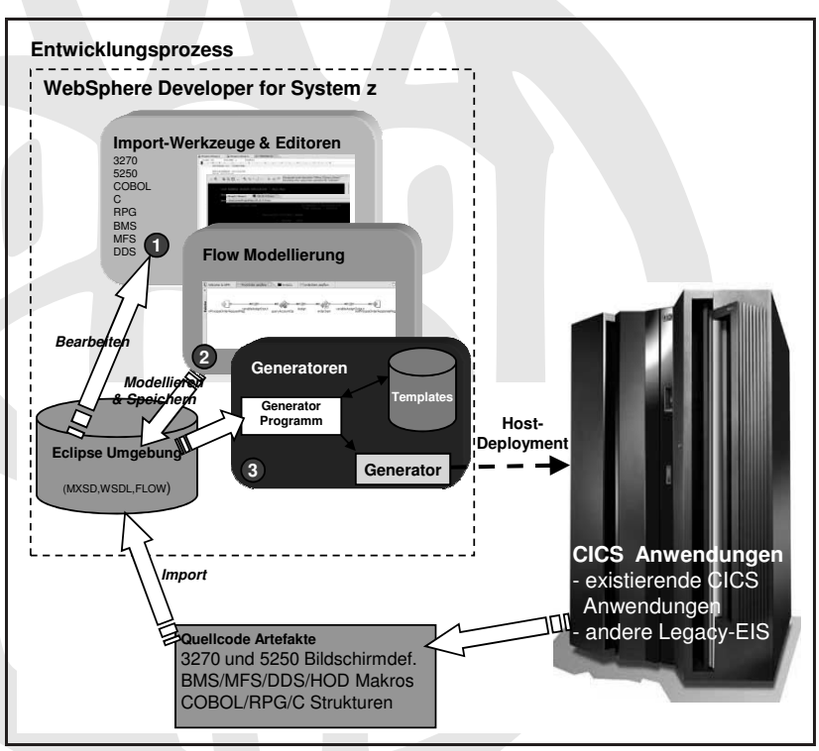

Bild 1 Zu durchlaufende Schritte, um einen SOA-fähigen Geschäftsprozess zu modellieren, der später auf dem Host deployed werden kann.

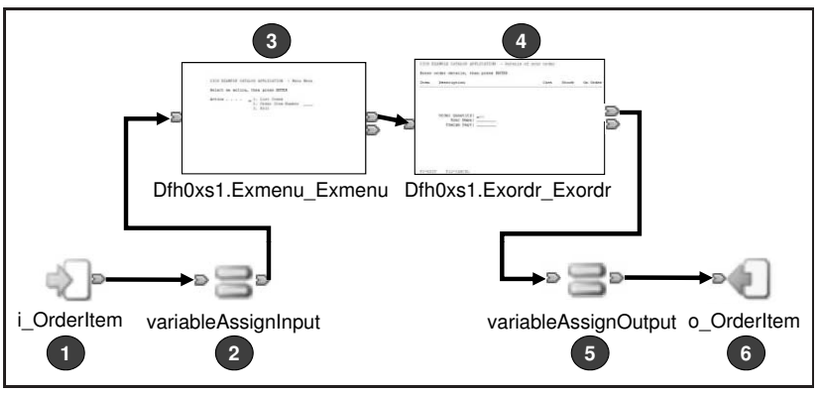

Bild 2 Beispiel eines Service Flows, bei dem zwei CICS-Bildschirme eines Kataloges durchlaufen werden.

oder Entfernen von Komponenten bzw. Pfeilen lässt sich die gesamte Logik des Geschäftsprozesses umgestalten.

4 Die Service Flow Runtime

Nach der Modellierung erstellen die Generatoren des RDz eine Laufzeitkomponente, die auf CICS-spezifische Möglichkeiten wie beispielsweise CICS Business Transaction Services und die Link3270-Bridge zurückgreifen. Erstere sorgen für eine hochoptimierte Implementierung in der CICS-Umgebung, die die von den integrierten CICS-Anwendungen bereitgestellte Quality of Service (QoS) sicherstellt. Sobald die Laufzeitkomponente erzeugt wurde, wird sie vom SFM zum Host übertragen und kann von nun an verwendet werden.

Die SFR erweitert CICS TS um Komponenten und Konfigurationen zur Ausführung von Service Flows. Weiterhin stellt die SFR die Adapter für den Zugriff auf CICS-Transaktionen und Schnittstellen von Legacy-Anwendungen zur Verfügung. Da bei der Geschäftsprozessmodellierung nicht-invasive Techniken zum Einsatz kommen, muss der Quelltext der zugrundeliegenden Anwendungen nicht verändert werden, womit eine schnelle und effiziente Wiederverwendung gewährleistet ist. In Bild 3 ist der Ablauf in der SFR nach dem Deployment eines Webservices dargestellt. Im ersten Schritt (1) installieren die vom SFM generierten JCL-Installationsskripte den neuen Webservice im CICS. Die JCLs werden mit Hilfe der initial erzeugten Template-Dateien erstellt und somit an die nötigen systemspezifischen Parametern (2) angepasst und an die entsprechend des modellierten Webservices nötigen Compiler und Interpreter (3) geschickt. Nun ist der Service Flow zur Laufzeit verfügbar und es kann auf ihn zugegriffen werden.

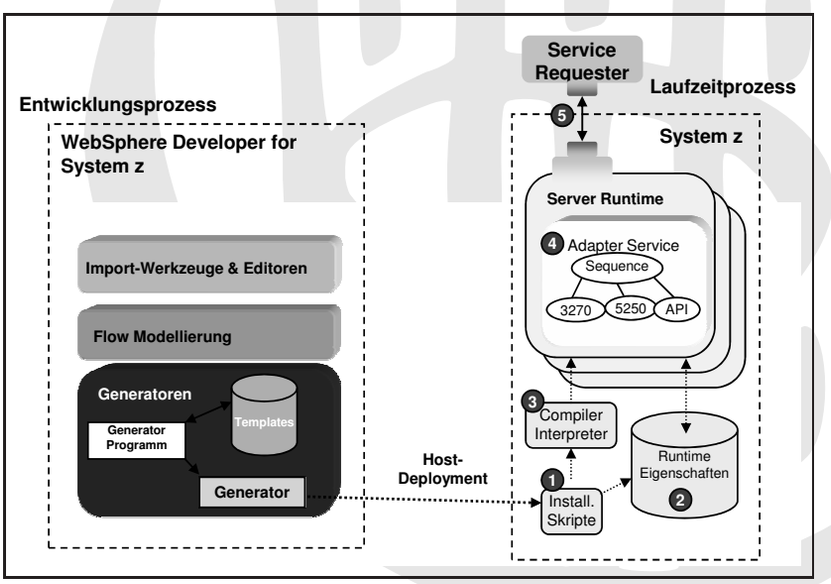

Bild 3 Die Verarbeitung des Service Flows auf dem Host zur Laufzeit.

Jedes Mal, wenn ein Webservice veröffentlicht wird, müssen dem CICS einige Parameter über die zu installierende Anwendung mitgeteilt werden. Dies sind beispielsweise die benötigten Ressourcen oder eventuelle Interaktionen zwischen verschiedenen Ressourcen. Aus diesem Grund muss der Webservice in einer Reihe von Schritten lauffähig gemacht werden. Allerdings wird dem Anwender diese Arbeit von den vom SFM automatisch erzeugten JCLs abgenommen. Wurde ein Webservice erfolgreich erzeugt und auf den Host exportiert, muss er noch manuell kompiliert werden, wozu nur ein einziger Befehl „SUBMIT“ nötig ist. Anschließend ist der Webservice einsatzfähig und kann über einen Service Requester (5) benutzt werden [6].

5 Ausblick

In diesem Beitrag wurde eine von IBM verfolgte Vorgehensweise zur Modernisierung von CICS-Anwendungen näher erläutert. Es wurde ein Weg aufgezeigt, um die oft über Jahrzehnte hinweg gewachsene Anwendungen, die ein Höchstmaß an Robustheit, Sicherheit, Stabilität und Verlässlichkeit bieten, in eine moderne und flexible IT-Landschaft zu überführen.

Für einen flexiblen Geschäftsbetrieb, sowie eine anpassungsfähige Informationstechnologie, sind die zuvor beschriebenen Modernisierungsmaßnahmen äußerst effizient. Um die zuvor aufgezeigten Werkzeuge zu verwenden, ist jedoch die Installation und Konfigura-

tion des RDz und der zugehörigen Host-Komponente notwendig. Beides erfordert einen spezifischen finanziellen (Software-Preis) und personellen (Installation der Host-Komponenten) Aufwand. Dieser Aufwand amortisiert sich jedoch ab einem gewissen Projektumfang.

Um möglichst viele Anwendungen auf diesem Wege zu erneuern, ist dies eine äußerst wirtschaftliche, performante und zuverlässige Lösung[4]. Insbesondere für große Firmen (Banken, Versicherungen, Versandthäuser) bildet die Modernisierung von Legacy CICS-Anwendungen eine gewaltige Herausforderung, die mit dem CICS SFF nachhaltig gelöst werden kann.

Literatur

[1] I. Arnold, C. Backhouse, L. Compton et al.: Application Development for CICS Web Services; IBM Redbook SG24-7126-00; May 2006.

[2] P. Herrmann, U. Kebschull, W. G. Spruth: Einführung in z/OS und OS/390; Oldenbourg Verlag; 2. Auflage; 2004.

[3] P. Herrmann, M. Meinhold: Java Server Pages als Präsentationslogik für CICS-Anwendungen mit Datenbankzugriff; München; PIK-Magazin; ISSN: 0930-5157; 02/2008.

[4] IBM: CICS Service Flow Runtime User's Guide, Version 3 Release 1; Seventh edition; IBM 5655-M15; November, 22 2005.

[5] C. Backhouse, J. Hollingsworth, S. Hurst, M. Pocock: Architecting Access to CICS within an SOA; IBM Redbook SG24-5466-05; October 2006.

[6] M. Keen, A. Acharya, S. Bishop et al.: Patterns: Implementing an SOA Using an Enterprise Service Bus Extend the value of your core business systems; IBM Red Book SG24-6346-00; July 2004.

[7] A. Landenberger: Java und CICS – Schrittweise Modernisierung auf dem Mainframe; JavaSPEKTRUM magazine 5/2007; May 2007.

[8] B. Moore, D. Dean, A. Gerber et al.: Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework; IBM Redbook SG24-6302-00; February 2004.

[9] F. Stefan: Aggregation of CICS Transactions with the Service Flow Feature; University of Leipzig; Master Thesis; November 2007.

1

2

3

1 M.Sc. Fred Stefan studierte an der Universität Leipzig Informatik und stellte 2008 seine Masterarbeit über die Anwendungsmodernisierung auf Mainframe-Rechnern mit dem CICS SFF in Zusammenarbeit mit IBM fertig. Seit Januar 2008 ist er externer Doktorand in der Abteilung Betriebliche Informationssysteme am Institut für Informatik der Universität Leipzig. Seine Interessen liegen im Bereich Integration Engineering, mit besonderem Fokus auf agile, leichtgewichtige Integrationstechniken.
Adresse: Universität Leipzig, Johannisgasse 26, 04103 Leipzig, Germany,
E-Mail: stefan@informatik.uni-leipzig.de

2 Dr. Paul Herrmann promovierte 1973 im Bereich Physik. Von 1968 bis 1989 war er Abteilungsleiter am Rechenzentrum der Universität Leipzig. Seit 2000 ist er Verantwortlicher für das Mainframe-Projekt am Institut für Informatik an der Universität Leipzig. In Forschung und Lehre widmet er sich Mainframe-und Rechnersystem-Technologien.
Adresse: Universität Leipzig, Johannisgasse 26, 04103 Leipzig, Germany,
E-Mail: paul@informatik.uni-leipzig.de

3 Prof. Dr.-Ing. Wilhelm G. Spruth studierte Elektrotechnik an der RWTH Aachen. Er ist Honorarprofessor am Institut für Informatik der Universität Leipzig, sowie am Institut für Informatik der Universität Tübingen. Vorher war er mehrere Jahre Entwicklungsleiter im IBM Entwicklungslabor in Böblingen. Herr Spruth ist Mitglied in der ACM, GI und der Computer Society des IEEE.
Adresse: Eberhard-Karls-Universität Tübingen, Sand 13, 72076 Tübingen, Germany,
E-Mail: spruth@informatik.uni-tuebingen.de

[4] Im Vergleich zur „*rip and rewrite*"-Lösung