ECONSTOR
Make Your Publications Visible.

A Service of
ZBW
Leibniz-Informationszentrum Wirtschaft
Leibniz Information Centre for Economics

Jung, Reinhard; Rimpler, Andreas; Schnieder, Thomas; Teubner, Alexander

Working Paper

Eine empirische Untersuchung von Kosteneinflußfaktoren bei integrationsorientierten Reengineering-Projekten

Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 37

Provided in Cooperation with:
University of Münster, Department of Information Systems

Suggested Citation: Jung, Reinhard; Rimpler, Andreas; Schnieder, Thomas; Teubner, Alexander (1995) : Eine empirische Untersuchung von Kosteneinflußfaktoren bei integrationsorientierten Reengineering-Projekten, Arbeitsberichte des Instituts für Wirtschaftsinformatik, No. 37, Westfälische Wilhelms-Universität Münster, Institut für Wirtschaftsinformatik, Münster

This Version is available at:
https://hdl.handle.net/10419/59369

Standard-Nutzungsbedingungen:

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

Terms of use:

Documents in EconStor may be saved and copied for your personal and scholarly purposes.

You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.

If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.

WWW.ECONSTOR.EU

Mitglied der Leibniz-Gemeinschaft

Arbeitsberichte des Instituts für Wirtschaftsinformatik
Herausgeber: Prof. Dr. J. Becker, Prof. Dr. H. L. Grob, Prof. Dr. K. Kurbel,
Prof. Dr. U. Müller-Funk, Prof. Dr. R. Unland, Prof. Dr. G. Vossen

Arbeitsbericht Nr. 37

Eine empirische Untersuchung von Kosteneinflußfaktoren bei integrationsorientierten Reengineering-Projekten

Reinhard Jung, Andreas Rimpler,
Thomas Schnieder, Alexander Teubner

Institut für Wirtschaftsinformatik der Westfälischen Wilhelms-Universität Münster,
Grevener Str. 91, 48159 Münster, Tel. (0251) 83-9750, Fax (0251) 83-9754
März 1995

Inhalt

1 Integrationsorientiertes Reengineering 3
1.1 Wirtschaftliche Integration 3
1.2 Einflußfaktoren auf die Kosten bei Integration durch Reengineering 4
1.3 Produktmerkmale 6

2 Durchführung der empirischen Untersuchung 7
2.1 Untersuchte Zusammenhänge 7
2.2 Forschungsform 9
2.3 Meßvorschriften 12
2.3.1 Unabhängige Variablen 12
2.3.2 Abhängige Variablen 15

3 Datenauswertung und Interpretation 16
3.1 Statistische Auswertung 16
3.1.1 Das Modell 16
3.1.2 Überprüfung der Annahmen 17
3.1.3 Schätzung der Einflüsse 19
3.1.4 Testen auf signifikante Unterschiede 19
3.1.5 Korrelation zwischen Zeitdauer und Vollständigkeit der Lösung 21
3.2 Interpretation der Ergebnisse 21

4 Ausblick 23

Literatur 24

Anhang 26

Zusammenfassung

Die Zusammenhänge zwischen Merkmalen eines Software-Altsystems und den daraus resultierenden Kosten für bestimmte Reengineering-Aktivitäten sind nahezu unerforscht. Der vorliegende Arbeitsbericht schildert die experimentelle Untersuchung von drei ausgewählten Einflußfaktoren bei einer für das integrationsorientierte Reengineering typischen Aufgabenstellung. Untersucht wurden die Wirkungen der Komplexität des Quelltexts, der Dokumentiertheit der Datenstrukturen sowie der Verfügbarkeit von Tages- bzw. Testdaten.

1 Integrationsorientiertes Reengineering

Die Integration vorhandener und neuer Anwendungssysteme ist in den letzten Jahren zu einer der wesentlichen Herausforderungen für die betriebliche Informationsverarbeitung geworden. Durch Datenintegration können hohe Kosten im Bereich der Datenerfassung (z.B. durch Mehrfacheingaben) und Datenpflege (z.B. durch Aktualisierung redundanter Daten) vermieden werden. Darüber hinaus bietet die Datenintegration die technischen Voraussetzungen sowohl für die softwaretechnische Funktionsintegration als auch für die aufgabenorientierte Funktionsintegration mit Hilfe von Anwendungssystemen (z.B. durchgängig unterstützte Arbeitsabläufe). Die heute zum Teil seit mehr als 10 Jahren in Betrieb befindlichen Anwendungssysteme wurden allerdings meist als Insellösungen konzipiert. Eine vollständige Ersetzung durch neue, miteinander integrierte Systeme stellt oft eine zu hohe Investition dar.

Das *integrationsorientierte Reengineering* ist ein Ansatz, der die Integration von Software-Altsystemen und Neuentwicklungen ermöglicht[1)]. Als vorrangiges Ziel wird die Integration der Daten verfolgt. Im Verlauf des Integrationsprozesses, der sich aus Kosten- und Kapazitätsgründen über einen längeren Zeitraum erstrecken kann, werden jeweils in Integrationsprojekten einzelne Anwendungssysteme in ein integriertes verteiltes Anwendungssystem (IVAS) eingebunden. Das Ergebnis ist i.d.R. ein verteiltes System, da die zu integrierenden Altsysteme oft nicht aus ihrer veralteten Laufzeitumgebung „herausgelöst“ werden können, während die neueren Systeme auf modernerer Hardware bzw. Software aufsetzen.

1.1 Wirtschaftliche Integration

Wenn im Rahmen eines Integrationsprojekts ein Altsystem integriert werden soll, stellt sich die Frage, ob es durch Reengineering integriert werden kann oder ob eine Ersetzung (Kauf von Standardsoftware oder Individualsoftware bzw. Eigenentwicklung) günstiger ist. Letztendlich handelt es sich bei der Beantwortung der Frage um die Suche nach der wirtschaftlicheren Alternative. Voraussetzung für einen Wirtschaftlichkeitsvergleich ist neben der Kenntnis eventuell unterschiedlicher Leistungen (bzw. Nutzen) der Alternativen die Kenntnis der zu erwartenden Kosten[2)]. Bezogen auf Neuentwicklungen ist bereits eine Vielzahl von Methoden und Verfahren zur Kosten- bzw. Aufwandsschätzung verfügbar (Cocomo, Function point, SLIM usw.). Entsprechende Methoden für Reengineering-Projekte sind dagegen nicht bekannt.

---

1) Vgl. Eicker et al. (1992); Eicker et al. (1993); Kurbel (1994).
2) Vgl. Eicker, Jung (1994); Jung (1995).

Der Ansatz des integrationsorientierten Reengineering ist aus einem von der Deutschen Forschungsgemeinschaft geförderten Projekt hervorgegangen, das am Institut für Wirtschaftsinformatik der Universität Münster durchgeführt wird. Neben der Erarbeitung von softwaretechnischen Konzepten wurden auch ökonomische Fragestellungen untersucht. Um erste grundlegende Erkenntnisse für die Kostenschätzung zu erhalten, wurde im Rahmen des Projekts eine empirische Untersuchung mit dem Ziel durchgeführt, Zusammenhänge zwischen Einflußfaktoren und resultierenden Kosten zu erkennen und einschätzen zu können.

1.2 Einflußfaktoren auf die Kosten bei Integration durch Reengineering

Um Einflußfaktoren zu erhalten, deren Betrachtung im Rahmen einer empirischen Untersuchung besonders erfolgversprechend ist, wurden in einem ersten Schritt Faktoren zusammengetragen, die einen signifikanten Einfluß auf die Kosten einer Reengineering-Maßnahme bzw. eines Reengineering-Projekts haben könnten. Grundsätzlich lassen sich drei verschiedene Klassen von Faktoren unterscheiden[3)]:

1. *Projektziele*: Im Rahmen des integrationsorientierten Reengineering ist die Integration das vorrangige Projektziel. Häufig kommen aber weitere Ziele hinzu, weil durch die Berücksichtigung in *einem* Projekt Kosten für mehrfache Einarbeitung in das Altsystem und für andere Mehrfacharbeiten eingespart werden können. Beispiele für weitere Ziele sind eine neue Benutzeroberfläche, eine Portierung in eine modernere Umgebung oder die Verbesserung der Wartbarkeit.

2. *Produktmerkmale*: Die „Beschaffenheit“ des vorliegenden Software-Altsystems ist eine wesentliche Determinante der Kosten für ein Reengineering-Projekt. Eine Vielzahl von Eigenschaften des Altsystems (z.B. Strukturiertheit, Selbstdokumentiertheit, Verfügbarkeit von Dokumentationen) beeinflussen die verschiedenen Aktivitäten eines Reegineering-Projekts und damit die Kosten.

3. *Projektrandbedingungen*: Den Projektrandbedingungen kommt ebenfalls eine große Bedeutung zu. Neben typischen Eckdaten wie zum Beispiel Terminvorgaben sind auch die technischen Ressourcen zu beurteilen. Eine komfortable Reengineering- bzw. Entwicklungsumgebung kann zur Senkung der Kosten beitragen. Darüber hinaus sind die Personalverfügbarkeit sowie die Personalqualifikation und -motivation von besonderer Bedeutung.

---

3) Vgl. Jung (1995).

Neben den genannten Klassen existiert ein Einflußfaktor, der keiner der Klassen zugeordnet werden kann. Der *Integrationsfortschritt* bzw. die *Modellüberschneidung* zwischen dem (immer größer werdenden) IVAS-Datenmodell und dem Altsystem-Datenmodell, die mit dem Integrationsfortschritt zunimmt, beeinflußt ebenfalls die Kosten. Der Grund liegt darin, daß die Identifizierung von synonymen bzw. redundanten Datenelementen bei größeren Modellen aufwendiger wird. Zudem steigt mit hoher Wahrscheinlichkeit die Anzahl der gefundenen Synonyme, und damit steigen auch die Kosten für die logische und physische Integration dieser Datenelemente. Die grundlegenden Zusammenhänge zwischen Einflußfaktoren und den Kosten eines Integrationsprojekts auf der Basis von Reengineering sind in Abbildung 1 dargestellt.

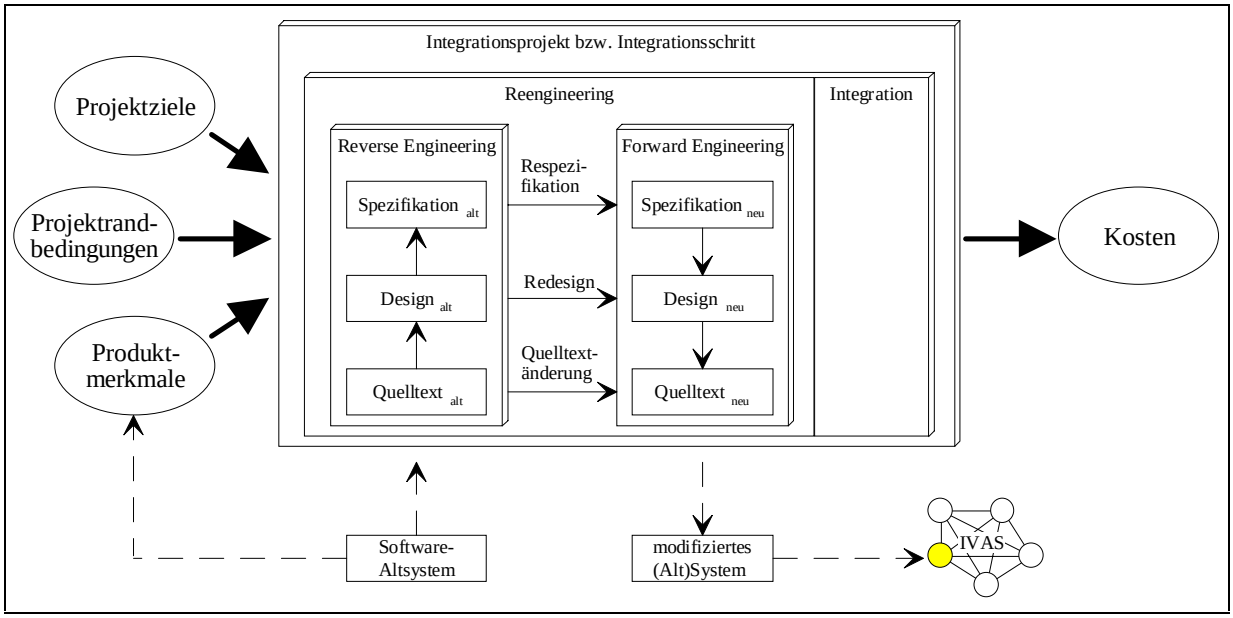

Abb. 1: Zusammenhänge zwischen Einflußfaktorklassen und Kosten eines Integrationsprojekt

Aus den genannten Klassen wurde schließlich diejenige für eine detaillierte Untersuchung ausgewählt, von der besonders wichtige Erkenntnisse für das integrationsorientierte Reengineering zu erwarten waren. Während *Projektziele* und *Projektrandbedingungen* in gleicher oder zumindest ähnlicher Weise Einfluß auf Neuentwicklungsprojekte und auf Reengineering-Projekte haben, sind *Produktmerkmale* nur in Reengineering-Projekten relevant. Darüber hinaus fehlen selbst im Bereich der Software-Wartung - einem für wissenschaftliche Maßstäbe eher alten Forschungsgebiet - Erkenntnisse über die genauen Zusammenhänge zwischen Produktmerkmalen und Wartungsaufwand[4)].

4) Vgl. Lehner (1991), S. 15.

1.3 Produktmerkmale

Die Produktmerkmale eines Software-Altsystems, die vermutlich Auswirkungen auf ein Reengineering-Projekt haben, betreffen hauptsächlich die *Verständlichkeit* und die *Änderbarkeit* eines Software-Altsystems. Die identifizierten Merkmale beziehen sich zum einen auf den Quelltext des Software-Altsystems und zum anderen auf weitere Komponenten des Systems „Software“ (Dokumentation, Tages- bzw. Testdaten usw.). Die folgenden, direkt quantifizierbaren Produktmerkmale des Quelltexts bzw. Einflußfaktoren wurden identifiziert (sprachspezifische Merkmale von Cobol sind durch „*“ gekennzeichnet)[5]:

- Anteil des Quelltexts, der mit Sicherheit nicht durchlaufen wird (sog. „toter Code“);
- Anzahl der Accepts (ohne Date, Time, Day)*, d.h. Benutzereingaben;
- Anzahl der vom Programm verarbeiteten Dateien und Tabellen;
- Anzahl der Datenelemente in der File-Section*;
- Anzahl nicht referenzierter Datenelemente
- Anzahl der Dateizugriffe;
- Anzahl von Displays/Literalen*;
- Länge des Quelltexts (Lines of Code; LOC);

Darüber hinaus wurden folgende Produktmerkmale des Quelltexts bzw. Einflußfaktoren identifiziert, die entweder nur indirekt quantifizierbar oder aber nur qualifizierbar sind:

- Art der Dateizugriffe (lesend und/oder schreibend);
- Art der verarbeiteten Dateien (sequentiell, wahlfrei, dynamisch*);
- Kommentiertheit (Menge und Qualität der Kommentare im Quelltext);
- Komplexität des Quelltexts;
- Konsistenz des Programmierstils;
- Einfachheit der Algorithmen;
- Modularisiertheit (Strukturiertheit im Großen);
- Modulfestigkeit;
- Verwendung parametrisierter Unterprogramme;
- Datenstrukturiertheit (Typisierung, Verwendung von Datenstrukturen);
- verwendete Programmiersprache;
- Selbstdokumentation (sprechende Bezeichner, Einhaltung von Namenskonventionen);
- Dokumentiertheit der Datenstrukturen, die für die Dateiein- und -ausgabe benutzt werden;
- Ablaufsteuerung über Werte, die erst zur Laufzeit des Programmes ermittelt werden;

---

5) Zu den Merkmalen vgl. auch Jung (1993), S. 16 ff.

- Verwendung sinnvoller Steuerkonstrukte (auch z.B. Vermeidung von Stop runs, Gotos *);
- Verwendung „undurchsichtiger“ Sprachkonstrukte (z.B. Rename, Redefine*).

Weitere Einflußfaktoren bzw. Produktmerkmale, die sich nicht auf den Quelltext beziehen, sind:

- Verfügbarkeit von Entwurfsdokumenten (z.B. Spezifikation, Design, Modulspezifikationen);
- Verfügbarkeit von Beschreibungen, die in einem Data Dictionary abgelegt sind;
- Qualität des Programmierhandbuchs bzw. der Systemdokumentation (Dokumentation der verarbeiteten Daten und Dokumentation der Programmlogik);
- Verfügbarkeit von Tages- und Testdaten;
- Verfügbarkeit von Testprotokollen.

Es muß davon ausgegangen werden, daß die Aufstellung der Einflußfaktoren bzw. der Produktmerkmale noch unvollständig ist. Nach dem aktuellen Erkenntnisstand sind die Autoren jedoch überzeugt, zumindest die Haupteinflußgrößen benannt zu haben. In welcher Weise sich diese Größen allerdings auf die Kosten für eine Reengineering-Maßnahme auswirken und welche Interdependenzen bestehen, ist nicht bekannt. Um diesen Zusammenhang zumindest partiell aufzudecken, wurde im Rahmen der Projektarbeit die Wirkung ausgewählter Produktmerkmale auf die Kosten für das Reengineering empirisch untersucht.

2 Durchführung der empirischen Untersuchung

2.1 Untersuchte Zusammenhänge

Für eine empirische Untersuchung der Wirkungszusammenhänge ergeben sich zwei wichtige Restriktionen:

- Nach aktuellem Forschungsstand kann - wie oben bereits erwähnt - nicht ausgeschlossen werden, daß relevante Einflußfaktoren noch unberücksichtigt sind. Auf Grundlage der bisher identifizierten Faktoren kann das Problem deshalb nur unvollständig beschrieben werden.

- Auch mit einer vollständigen Problembeschreibung ist es wegen der Vielzahl der Einflußfaktoren und -kriterien nahezu unmöglich, die Beeinflussungsstruktur vollständig in einem Untersuchungsdesign zu erfassen. Die Hauptschwierigkeit liegt in der Erfassung der Wechsel-

wirkungen, die zwischen den einzelnen Einflußfaktoren bestehen (z.B. zwischen Dokumentations- und Verständlichkeitskriterien). Sie können für eine größere Anzahl von untersuchten Einflußfaktoren praktisch nicht erfaßt werden, da die Anzahl der Wechselwirkungen exponentiell mit der Anzahl der Faktoren wächst.

Unter diesen Bedingungen schien es sinnvoll, einige wenige, für die Fragestellung des integrationsorientierten Reengineering jedoch besonders interessierende Faktoren auszuwählen und deren Auswirkungen auf die Kosten von Reengineering-Projekten zu untersuchen. Da das integrationsorientierte Reengineering eine sehr weitgehender Ansatz ist, wurde eine Einschränkung auf die entscheidende Teilaufgabe bzw. Phase im Reengineering-Prozeß, das *Reverse engineering*, vorgenommen. Hiermit wird der wichtige Schritt bezeichnet, in dem aus einem vorliegenden Altsystem Entwurfsentscheidungen extrahiert (das Programm wird verstanden) und u.U. für eine folgende Veränderung auf abstrakterem Niveau (z.B. als Design) dargestellt werden[6)].

Das Herauslösen einzelner Faktoren aus ihrem Wirkungskontext führt jedoch dazu, daß *Wechselwirkungen mit nicht untersuchten Faktoren* aus der Untersuchung ausgeblendet werden. Daher lassen sich umfassende Erklärungszusammenhänge für die Wechselwirkungen im Rahmen dieser Untersuchung nicht ermitteln.

Unter Berücksichtigung der für die empirische Untersuchung zur Verfügung stehenden Ressourcen wurden drei Faktoren für die empirische Untersuchung ausgewählt:

1. Dokumentiertheit der Datenstukturen (D),
2. Komplexität des Quelltexts (K),
3. Verfügbarkeit von Tages- bzw. Testdaten (T),

Auf eine Berücksichtigung der *Interdependenzen zwischen den untersuchten Faktoren* zugunsten einer genaueren Erfassung der Einzelwirkungen wurde verzichtet (monovariate Analyse). Es ist jedoch nicht auszuschließen, daß solche Effekte in der Realität auftreten. Denkbar ist z.B., daß sich die Verfügbarkeit von Tages- bzw. Testdaten bei einem Quelltext mit geringer Komplexität stärker (positiver im Sinne von geringeren Kosten) auswirkt als bei hoher Komplexität, weil das Nachvollziehen von Datenfluß und Datenmanipulation „im“ Programm natürlich mit steigender Komplexität schwieriger wird.

---

6) Vgl. Chikofsky, Cross (1993), S. 15; Lehner (1991), S. 234 f.

Als Kostenindikatoren für das Reverse engineering wurden der für die Lösung der Aufgabe notwendige *Personalaufwand* und die dabei erzielte Qualität bzw. die *Vollständigkeit der Lösung* herangezogen.

Der Personalaufwand wird als besonders wichtiger Kostenfaktor in IV-Projekten eingeschätzt[7]. Andere Produktionsfaktoren wie beispielsweise Betriebsmittel (Hard- und Software) haben dagegen einen vergleichsweise geringen Anteil an den Kosten. Weniger offenkundig ist dagegen die Bedeutung der Vollständigkeit der Lösung für die Kosten eines Reengineeringprojekts. Mit der getrennten Beobachtung von Zeit und Vollständigkeit wird dem realen Phänomen Rechnung getragen, daß die Gewinnung konzeptioneller Informationen aus Altsystemen typischerweise nicht in einem Arbeitsgang vollständig gelingt und in mehreren Iterationen vollzogen werden muß. Das bedeutet jedoch, daß man die Effektivität eines Reverse engineering-Arbeitsgangs nicht alleine an der benötigten Zeit messen kann, sondern gleichzeitig berücksichtigen muß, wie umfassend die benötigten konzeptionellen Informationen erfaßt wurden.

Abbildung 2 stellt in einem groben Modell die Wirkungszusammenhänge zwischen Produktmerkmalen eines Altsystems und den Kosten dar. Während die reale Wirkung der einzelnen Faktoren über ein Netzwerk von Beziehungszusammenhängen vermittelt wird, erfaßt die Untersuchung diese Einzelwirkungen jeweils als einen Gesamteinfluß. Die Betrachtung der jeweiligen Gesamteinflüsse erfolgt unabhängig voneinander.

2.2 Forschungsform

Als Vorgehensweise zur empirischen Untersuchung des interessierenden Zusammenhangs wurde das (Labor-)Experiment gewählt. Als Experiment wird eine wiederholbare Forschungsanordnung bezeichnet, bei der unter kontrollierbaren Bedingungen die unabhängigen Variablen so manipuliert werden, daß Kausalaussagen über die abhängige Variable möglich werden[8].

Bei der Untersuchung der Merkmale von Software-Altsystemen auf die Vollständigkeit der Lösung und auf den Personalaufwand (Zeit) des Reverse engineering bietet es sich an, unterschiedlichen Versuchspersonen (Vpn) Altsysteme mit unterschiedlichen Merkmalen zu präsentieren und anschließend zu ermitteln, wie sich die Vpn hinsichtlich der Lösung der gestellten Reverse engineering-Aufgabe verhalten. Wenn Gruppen von Vpn mit jeweils (in bestimmten Merkmalen) unterschiedlichen Altsystemen unterschiedliche Erfolge (Zeit und Vollständigkeit)

---

7) Vgl. Boehm, Papaccio (1988), S. 1468.
8) Vgl. Müller-Böling, Klandt (1994) S. 83.

in der Lösung der Aufgabenstellung erzielen, so kann von einem kausalen Zusammenhang zwischen unabhängigen und abhängigen Variablen ausgegangen werden.

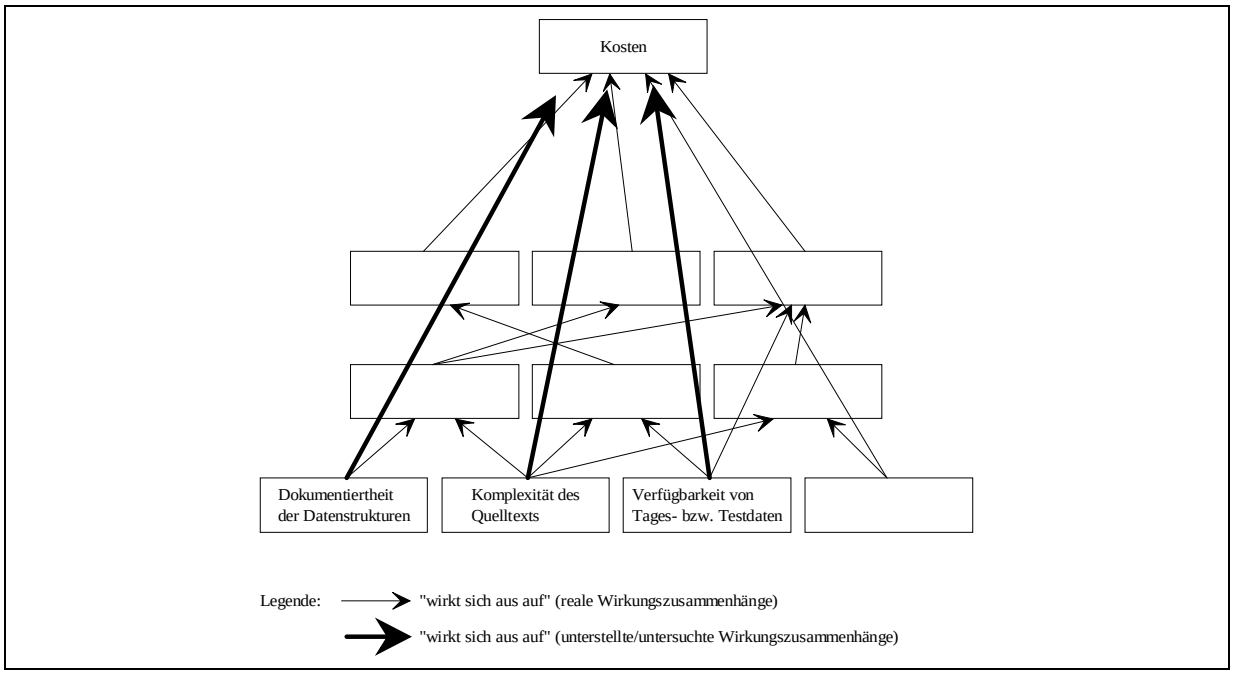

Abb. 2: Wirkungszusammenhänge zwischen den Einflußfaktoren und den Kosten in einer vereinfachten Darstellung

Zwei wichtige Merkmale der Forschungsform „(Labor-)Experiment“ müssen hinsichtlich der Validität der Forschungsergebnisse näher untersucht werden: die *Kontrollierbarkeit* und die *Künstlichkeit* der Versuchsanordnung.

Kontrollierbarkeit

Zentrales Merkmal der Forschungsform „Experiment“ ist die Kontrollierbarkeit der Bedingungen. Alle Faktoren, die sonst einen Einfluß auf die abhängige Variable haben, müssen konstant gehalten oder zumindest in ihrem Einfluß kontrolliert (d.h. bestimmt) werden. Die wichtigste Voraussetzung hierfür ist die Gleichheit der Gruppen bezüglich der nicht untersuchten unabhängigen Variablen. Im vorliegenden Fall war zu vermuten, daß die Qualifikation und Erfahrung der Vpn (in Abschnitt 1.3 unter Projektmerkmalen erfaßt) einen Einfluß auf die Vollständigkeit der Lösung sowie auf die benötigte Zeit haben, der die Einflüsse der untersuchten Produktmerkmale überlagert. Die absolute Gleichheit der Gruppen kann in den seltensten Fällen erreicht werden, da meist nicht alle Einflußfaktoren bekannt sind. Um dennoch die Prämisse gleicher Gruppen aufrechterhalten zu können, werden typischerweise Gruppenaufteilungen

nach dem Randomizing- oder Matching-Verfahren vorgenommen[9]. In unserem Fall wurde eine Kontrolle der Erfahrungen der Vpn durch Matching erreicht. Entsprechend ihrer individuellen Erfahrung mit prozeduralen Programmiersprachen wurden dieVpn so auf die Gruppen verteilt, daß jede Gruppe sowohl aus erfahrenen als auch weniger erfahrene Vpn bestand.

Künstlichkeit

Ein Problem der Forschungsform „Laborexperiment“ ist die Künstlichkeit der Situation. Mit Bezug auf die Künstlichkeit mußten für die Durchführung der Experimente folgende Probleme berücksichtigt bzw. bewältigt werden:

- Die maximale Dauer des Experiments mußte auf einen Tag begrenzt werden, um zu verhindern, daß bei längerer Dauer zunehmend nicht kontrollierbare Faktoren (z.B. die Beschäftigung mit der Aufgabe während der Unterbrechungen) auf das Ergebnis einwirken. Der Umfang der Aufgabenstellung konnte deshalb nicht realistisch gewählt werden, sondern es war eine Teilaufgabe auszuwählen.

- Die Arbeitssituation (Arbeitsumgebung, Motivation usw.) entsprach nicht der betrieblichen Realität.

- Aus Gründen der Verfügbarkeit wurden Studenten als Vpn gewählt.

Aus diesen Restriktionen resultiert unweigerlich ein Validitätsproblem, daß die Übertragbarkeit der Erkenntnisse aus den Experimenten auf die betriebliche Realität einschränkt. Um diesem Problem entgegenzuwirken, wurde versucht, die einzelnen Restriktionen zu entschärfen.

Bei der *Formulierung der Aufgabenstellung* wurde darauf geachtet, daß die gestellte Aufgabe typisch für das Reverse engineering ist. Die Vpn sollten die Datenelemente der File Section eines vorgegebenen Cobol-Anwendungsprogramms nachdokumentieren und Redundanzen zwischen Datenelementen unterschiedlicher Dateien erkennen. Dieser Arbeitsschritt spielt bei der Integration von Software-Altsystemen eine zentrale Rolle, wenn es darum geht, vorhandene Redundanzen zu identifizieren und korrekt zu behandeln[10]. Der Arbeitsschritt ist erforderlich, um das Datenmodell eines Altsystems zu rekonstruieren und um es später in das IVAS-Datenmodell integrieren zu können.

---

9) Vgl. Müller-Böling, Klandt (1994) S. 85.
10) Vgl. Eicker, Schnieder (1993); Eicker et al. (1993).

Bei dem Programm handelte es sich um ein sogenanntes „Real word“-System, d.h. um ein Programm, das in der ursprünglichen Form in einem Unternehmen eingesetzt wurde. Das Programm hatte einen Umfang von 1260 Lines of code (LOC) in der Programmiersprache Cobol und lief auf einer IBM-370, es handelte sich somit um eine für den betrieblichen Gebrauch typische Kombination von eingesetzter Software (bzw. Programmiersprache) und Hardware[11)].

Bei der *Gestaltung der Arbeitssituation* wurden den Vpn Dokumente und Hilfsmittel zur Verfügung gestellt, wie sie auch in der betrieblichen Praxis typischerweise verfügbar sind:

- IBM-kompatibler PC 80486 mit Drucker und Maus,
- Microsoft Cobol 5.0 (einschließlich des Debuggers) als „Reengineering-Werkzeug“,
- das zu untersuchende Programm (als Datei und als Ausdruck).

Als *Versuchspersonen* wurden - wie oben erwähnt - aus Gründen der Verfügbarkeit Studenten der Wirtschaftsinformatik gewählt. Es wurde jedoch darauf geachtet, daß die Vpn über praktische Erfahrungen mit der eingesetzten Entwicklungsumgebung und der Programmierprache Cobol verfügten. Darüber hinaus waren auch praktische Erfahrungen mit anderen prozeduralen Programmiersprachen vorhanden. In einer den Experimenten vorausgehenden Schulung wurden die Vpn anhand eines praktischen Beispiels (vgl. das Beispielprogramm im Anhang) in die Aufgabenstellung eingearbeitet.

2.3 Meßvorschriften

Für die zu untersuchenden Einflußfaktoren bzw. unabhängigen Variablen *Dokumentiertheit*, *Komplexität des Quelltexts* und *Verfügbarkeit von Tages- bzw. Testdaten* (vgl. Abschnitt 2.1) sowie für die abhängige Variable *Kosten* waren Meßvorschriften zu formulieren, die im folgenden erläutert sind.

2.3.1 Unabhängige Variablen

Das Experiment als Forschungsform ist dadurch gekennzeichnet, daß der Forscher die unabhängigen Variablen „manipuliert“, um zu erkennen, wie sich dieses auf die abhängige Variable auswirkt. Das bedeutet, daß die Ausprägungen der unabhängigen Variablen schon vor der Durchführung des Experiments festgelegt werden.

---

11) Vgl. Banker et al. (1993), S. 87.

Ausgangspunkt für die Bildung der Ausprägungen der unabhängigen Variablen ist das erwähnte Cobol-Programm. Die ursprüngliche Version des Programms stellt jeweils eine Ausprägung in jeder experimentellen Untersuchungsreihe, d.h. für jede Untersuchung einer unabhängigen Variablen, dar. Weitere Ausprägungen der Merkmale wurden durch gezielte Veränderungen im Programm selber bzw. durch die Erstellung zusätzlicher Dokumente oder Daten erreicht. Das bedeutet, daß weitere Versionen künstlich entstanden.

Dokumentiertheit der Datenstukturen (D)

Für die Variable „Dokumentiertheit der Datenstukturen (D)“ wurden drei Programmversionen konstruiert, in denen jeweils die Beschreibungen der Datensatzstrukturen in der File Section variiert wurden. Zur Veranschaulichung der einzelnen Versionen ist ein bestimmter Ausschnitt aus den Datensatzstrukturen in seinen verschiedenen Ausprägungen angegeben:

- Ausprägung 1: In der Beschreibung der Datensatzstruktur wurden *ausschließlich nichtsprechende Bezeichner* für die Datenelemente verwendet. Beispiel:

```
05  FS-SATZ.
    10  FS204                PIC  X(30).
    10  FS205                PIC  X(30).
```

- Ausprägung 2: In dieser Ausprägung wurden zur Beschreibung der Datensatzstruktur *zum Teil sprechende Bezeichner* für die Datenelemente verwendet. Die Bezeichner in der Datensatzstruktur der ersten Datei wurden wie unter „Ausprägung 1“ gewählt, die Bezeichner der anderen Datei wie unter „Ausprägung 3“.

- Ausprägung 3: In dieser Ausprägung wurden zur Beschreibung der Datensatzstruktur *ausschließlich sprechende Bezeichner* für die Datenelemente verwendet. Beispiel:

```
05  FS-SATZ.
    10  TitelNamenHinweise1   PIC  X(30).
    10  TitelNamenHinweise2   PIC  X(30).
```

- Ausprägung 4: Die Bedeutung der einzelnen Datenelemente wurde *in einer externen Dokumentation* für jedes Datenelement beschrieben, die wie folgt aufgebaut war (Beispiel):

  FS204: Anrede, Vorname, Vorname Ehegatte, Nachname (Kontoinhaber);
  alternativ: kann Anrede oder Hinweis enthalten;
  FS205: Anrede, Vorname, Vorname Ehegatte, Nachname (Kontoinhaber);
  alternativ: Ansprechpartner in einer Firma, die Depotinhaber ist.

Komplexität des Quelltexts (K)

Als dritte unabhängige Variable wurde die Komplexität des Quelltexts variiert. Gemessen wurde die Komplexität mit McCabe´s Essential Complexity ist ein häufig benutztes Komplexitätsmaß, auf dessen Basis Aussagen über den Aufwand für die Durchführung von Wartungsaktivitäten gemacht werden können.

Für das Experiment wurde die McCabe Essential Complexity (EC) des Originalsystems durch Änderung der Anzahl der Sprungbefehle variiert. Zusätzlich zu der Ausgangsvariante mit einer Essential Compexity von EC = 7 (Ausprägung 2) entstanden so eine Variante mit der Komplexität EC = 2 (Ausprägung 1) und eine zweite mit der Komplexität EC = 95 (Ausprägung 3).

Verfügbarkeit von Tages- bzw. Testdaten (T)

Für die Untersuchung des Einflusses von Tages- bzw. Testdaten (T) auf Aufwand und Vollständigkeit wurden neben der ursprünglichen Version (ohne Daten) drei Varianten:

- Ausprägung 1: Die ursprüngliche Version ohne die Bereitstellung irgendwelcher Daten wurde als erste Ausprägung gewählt.

- Ausprägung 2: Zusätzlich zu dem Programm wurden den Vpn von jeder Datei vier Datensätze aus den *Tagesdaten*[12)] als Ausdruck zur Verfügung gestellt. Beispiel (für die Datenelemente FS204 und FS205):

  ```
  Sparkauf
  Z.HD.FRAU Maier
  ```

- Ausprägung 3: Den Vpn wurden dieselben Daten wie bei Ausprägung 2 zur Verfügung gestellt, allerdings in Dateiform. So war es möglich, das Programm mit Hilfe der Tagesdaten und des Debuggers schrittweise ablaufen zu lassen, um Informationen zu erhalten.

- Ausprägung 4: Den Vpn wurden *Testdaten*[13)] in Dateiform zur Verfügung gestellt. Die Testdaten waren so konstruiert, daß eine *minimale Mehrfachbedingungsüberdeckung* von

---

12) Unter *Tagesdaten* wird eine Menge von Datensätzen aus dem Stammdatenbestand einer Unternehmung verstanden, die "zufällig", also ohne Anwendung festgelegter Kriterien, aus dem Datenbestand gezogen werden. Es handelt sich demnach um Datensätze, die vom Programm vermutlich korrekt verarbeitet werden.

13) Unter *Testdaten* wird eine Menge von Datensätzen verstanden, die im Hinblick auf die Verarbeitung durch ein Programm konstruiert wurden. Testdaten werden konstruiert, um eine gewissen Art der Ausführung eines Programms analysieren zu können.

89,4 % erreicht wurde. Diese Quote wurde mit 13 bzw. 14 Datensätzen für die beiden Dateien erreicht.

Die Konstruktion von Testdaten gemäß dem Kriterium „Minimale Mehrfachbedingungsüberdeckung“ ergibt eine Menge von Testdatensätzen. Erlaubte Kombinationen von Datenwerten und damit auch eine Menge von Integritätsregeln sowie auch Zusammenhänge zwischen Daten unterschiedlicher Dateien sind schon zu einem beachtlichen Anteil aus den Daten selbst ermittelbar. Nur für einen kleineren Rest der Aufgaben und für die Verifikation der aus den Datenwerten abgeleiteten Hypothesen über die Bedeutung von Datenfeldern ist es noch erforderlich, den Quelltext zu analysieren.

2.3.2 Abhängige Variablen

Ziel der Experimente war es, den Einfluß ausgewählter Produktmerkmale (von Altsystemen) auf die Kosten für das Reverse engineering zu messen. Zur Erfassung der Kosten wurden zwei Indikatoren herangezogen: Zum einen die *Zeit*, die zur Durchführung der Aufgabe notwendig ist und zum anderen die *Vollständigkeit der Lösung*, die ein Indikator für den Umfang notwendiger Nacharbeiten in eventuellen Folgeprojekten ist. Die getrennte Erfassung von Zeit und Vollständigkeit hat neben der inhaltlichen (vgl. Abschnitt 2.1) auch eine meßtheoretische Bedeutung:

Für die Formulierung der Meßvorschrift zur Erfassung der Zeit war es notwendig, den Zeitpunkt des Abschlusses eines Experiments zu bestimmen. Für einen 100m-Lauf ist die Aufstellung einer Meßvorschrift vergleichsweise trivial. Sie könnte lauten: Wenn ein Läufer die Ziellinie überschreitet, so ist diesem Ereignis als Zahlenwert die gestoppte Zeit in Sekunden zuzuordnen. Hieraus ergibt sich die meßtheoretisch notwendige eindeutige Zuordnung eines numerischen Relativs (Laufzeiten in Sekunden) zu dem empirischen Relativ (Rangfolge einer Anzahl von Läufern beim Zieleinlauf). Im Falle des Reverse engineering ist es jedoch weitaus schwieriger, den Zeitpunkt der Erfüllung der Aufgabe bzw. in der Untersuchung den Abschluß des Experiments genau festzulegen.

Damit die Vpn das Ende des Experiments „erkennen“ konnten, wurde ihnen eine auf Formularen basierende Vorgehensweise zur Lösung der Aufgabe zur Verfügung gestellt (vgl. Anhang). In die Formulare waren alle konzeptionellen Informationen einzutragen, die gemäß der Aufgabenstellung aus dem Altsystem extrahiert werden sollten. Damit war den Vpn eine Einschätzung möglich, wann die Aufgabe erfüllt war. Jedoch konnte dadurch nicht sichergestellt werden, daß das inhaltliche Ziel tatsächlich vollständig erreicht wurde. Es bestand die Möglichkeit, daß einzelne Vpn die Unvollständigkeit ihrer Lösung erkannten, sich jedoch nicht in der Lage

sahen, eine vollständige Lösung zu erreichen, oder daß sie ihre Lösung irrtümlich für vollständig hielten.

Die Meßvorschrift mußte deshalb wie folgt formuliert werden: Das Experiment gilt als beendet, wenn die Vpn nach subjektiver Einschätzung davon ausgeht, keine Verbesserung der Lösung mehr erzielen zu können. Die Zeit vom Beginn bis zum Abschluß des Experiments ist in Minuten zu messen.

Diese Meßvorschrift stellt jedoch keine eindeutige Zuordnung eines numerischen zu einem empirischen Relativ dar. So beschreibt die Aussage „Experiment A konnte in 250 Minuten abgeschlossen werden während für Experiment B 290 Minuten benötigt wurden“ die Realität nur unvollständig, wenn im Experiment A die Aufgabe nur zu 70% gelöst wurde, in B aber zu 100%. Eine meßtheoretisch korrekte Vorschrift muß also zusätzlich die Vollständigkeit der Lösung erfassen. Dies war anhand der vorgegebenen Formulare auch praktikabel. Die Vollständigkeit der Lösung wurde in Prozent gemessen. In *Pre-Tests* wurde vor den auszuwertenden Experimenten ermittelt, für welchen Anteil der Gesamtlösung wieviel Zeit benötigt wurde. Anhand dieser Werte wurde die Vollständigkeit der von den Vpn erarbeiteten Lösungen ermittelt.

Die abhängigen Variablen - die Zeit gemessen in Minuten und die Vollständigkeit gemessen in Prozent - wurden für die Auswertung als metrisch-skaliert interpretiert. Die Ausprägungen der unabhängigen Variablen wurden als nominal-skaliert interpretiert[14]. Mit Hilfe varianzanalytischer statistischer Auswertungen war es dann möglich, den Einfluß der nominal-skalierten unabhängigen Variablen auf die metrisch-skalierten abhängigen Variablen quantitativ zu bestimmen.

3 Datenauswertung und Interpretation

3.1 Statistische Auswertung

3.1.1 Das Modell

Bei dem hier betrachteten Experiment wurde die Abhängigkeit zweier Größen (Zeitdauer, Vollständigkeit der Lösung) von mehreren Einflußfaktoren (Komplexität des Quelltexts, Dokumentiertheit der Datenstrukturen, Verfügbarkeit von Tages- bzw. Testdaten) monovariat

---

14) Zur Bestimmung des Meßniveaus vgl. Müller-Böling, Klandt (1993), S. 21.

untersucht. Dabei wurde der Einfluß sowohl auf die Zeitdauer als auch auf die Vollständigkeit der Lösung für jeden Faktor getrennt betrachtet, so daß insgesamt sechs verschiedene Abhängigkeiten zu untersuchen waren. Außerdem interessierte die Korrelation zwischen Zeitdauer und Vollständigkeit. Auf die Untersuchung möglicher Wechselwirkungen der Einflußfaktoren wurde verzichtet (siehe Abschnitt 2.1). Die Modellierung solcher Abhängigkeiten geschieht üblicherweise mit Hilfe von linearen Modellen. Dabei wird angenommen, daß die Effekte der Einflußfaktoren linear in die abhängige Größe eingehen und das Versuchsergebnis von einem zufälligen, additiven Störterm beeinflußt wird, der aufgrund des Einflusses anderer, nicht betrachteter Faktoren sowie aufgrund der zufälligen Personenauswahl und Messungenauigkeiten auftritt (vgl. Abschnitt 2.2). Da die untersuchten Einflußfaktoren als qualitative Größen behandelt wurden, hat man in allen sechs Fällen die gleiche Struktur, nämlich eine einfaktorielle Varianzanalyse:

$$X_{ij} = v_i + U_{ij} \qquad i=1, ..., k, \;\; j = 1, ..., n_i \; .$$

Dabei bezeichnet X_{ij} den jeweiligen Stichprobenwert der abhängigen Größe (Zeitdauer bzw. Vollständigkeit) und v_i den Effekt der i-ten Stufe bzw. Ausprägung des jeweiligen Einflußfaktors. Die Konstante k repräsentiert die Anzahl der Stufen der einzelnen unabhängigen Variablen; die Komplexität hat drei Stufen, die beiden anderen Variablen jeweils vier Stufen. Die Gruppengröße wird mit n_i (hier $n_i = 5$ für alle i) bezeichnet und U_{ij} ist der zufällige Störterm, der üblicherweise als normalverteilt mit Erwartungswert 0 und unbekannter Varianz σ^2 angenommen wird. Man ist nun vor allem an folgenden Erkenntnissen über die unbekannten Parameter $v_1, ..., v_k$ interessiert:

1. Einflüsse v_i auf die abhängige Größe;
2. Signifikanz der Unterschiede zwischen den Auswirkungen der Stufen eines Einflußfaktors auf die beiden abhängigen Variablen.

Die üblicherweise hierfür verwendeten statistischen Verfahren erfordern die Normalität der Fehlerverteilung und die Homogenität der Gruppenvarianzen.

3.1.2 Überprüfung der Annahmen

Die Prüfung der Nullhypothese, daß eine Normalverteilung vorliegt, erfolgte mit dem Lilliefors-Test[15)] und ergab die in Tabelle 1 dargestellten P-Werte[16)]. Vertikal sind die abhängigen

---

15) Vgl. Sachs (1992).
16) Die P-Werte geben die Wahrscheinlichkeit an, die Nullhypothese zu verwerfen, obwohl sie richtig ist.

Variablen eingetragen, wobei „V“ für „Vollständigkeit der Lösung“ und „Z“ für „Zeit“ steht. Horizontal sind die unabhängigen Variablen Komplexität, Dokumentiertheit und Test- bzw. Tagesdatenverfügbarkeit (symbolisiert durch „K“, „D“ und „T“) eingetragen. Im folgenden werden die genannten Kurzformen verwendet. Buchstabenkombinationen kennzeichnen einzelne Exprimente. Das Symbol „ZD“ bezeichnet z.B. das Experiment, in dem die Zeit („Z“) in Abhängigkeit von der Dokumentiertheit („D“) untersucht wurde.

	K	D	T
V	> 0.2	> 0.2	> 0.2
Z	> 0.2	0.036	> 0.2

Tab. 1: Ergebnisse des Lilliefors-Tests auf Normalverteilung

Die Ergebnisse wurden durch Normalverteilungsplots gestützt, so daß lediglich im Modell ZD die Normalverteilungsannahme nicht haltbar erscheint.

Die Prüfung auf Homogenität der Gruppenvarianzen erfolgte mit dem Cochran-Test[17)], bei stärkeren Abweichungen von der Normalverteilung, insbesondere bei der Wölbung[18)], in den Modellen ZD und ZT mit dem Levene-Test[19)]. Auf dem 5%-Niveau lauten die kritischen Werte für den Cochran-Test: $c_{3;4;0.05} = 0.7457$, $c_{4;4;0.05} = 0.6287$. Die Ergebnisse des Cochran-Tests sind in Tabelle 2 dargestellt.

	K	D	T
V	0.416	0.443	0.411
Z	0.693	*	*

Tab. 2: Werte der Cochran-Teststatistik für H_0: $\sigma_1 = ... = \sigma_k$
gegen K_0: es gibt $i \neq j$ mit $\sigma_i \neq \sigma_j$

Die Annahme der Homogenität ist also in den geprüften Modellen (VK, VD, VT und ZK) haltbar. Die P-Werte des Levene-Tests lauten $P_{ZD} = 0.021$ und $P_{ZT} = 0.006$, so daß in diesen beiden Modellen die Gruppenvarianzen als signifikant verschieden zu betrachten sind.

17) Vgl. Sachs (1992).
18) Vgl. Lindman (1991).
19) Vgl. Sachs (1992).

3.1.3 Schätzung der Einflüsse

Die Parameter $v_1, ..., v_k$ wurden durchweg mit Hilfe der Least Squares-Methode geschätzt, die selbst unter Nichtnormalität und Heterogenität gute Eigenschaften besitzt. Bei der einfaktoriellen Varianzanalyse ergeben sich als Schätzer gerade die jeweiligen Gruppenmittelwerte der Stichproben (vgl. Tabellen 3 und 4).

Faktoren\Stufen	1	2	3	4
K	256.4 (60.3)	240.0 (29.6)	250.6 (26.8)	*
D	245.4 (29.0)	240.0 (29.6)	328.2 (84.6)	214.8 (23.8)
T	240.0 (29.6)	244.8 (25.9)	287.6 (55.8)	321.6 (18.6)

Tab. 3: Geschätzte Mittelwerte v_i in Bezug auf die Zeitdauer (Standardabweichung in Klammern)

Faktoren\Stufen	1	2	3	4
K	0.527 (0.215)	0.478 (0.288)	0.531 (0.265)	*
D	0.546 (0.323)	0.478 (0.288)	0.896 (0.124)	0.697 (0.182)
T	0.478 (0.288)	0.574 (0.180)	0.552 (0.202)	0.560 (0.215)

Tab. 4: Geschätzte Mittelwerte v_i in Bezug auf die Vollständigkeit (Standardabweichung in Klammern)

3.1.4 Testen auf signifikante Unterschiede

Mit Hilfe geeigneter Testverfahren wurde anschließend untersucht, ob die beobachteten Unterschiede zwischen den Stufen v_i signifikant sind oder nur auf zufälligen Schwankungen beruhen. Dabei wurde zunächst die Hypothese

$$H_0: v_1 = ... = v_k \qquad \text{gegen} \qquad K_0: \text{es gibt } i, j, i \neq j \text{ mit } v_i \neq v_j$$

getestet. Bei Ablehnung von H_0 wurde genauer untersucht, zwischen welchen zwei Stufen ein signifikanter Unterschied besteht, indem man die Kontraste $v_i - v_j$ betrachtete, was auf Hypothesen der Form $H: v_i = v_j$ gegen $K: v_i \neq v_j$ führte.

Der unter den Standardannahmen verwendete F-Test für (H_0, K_0) wurde auch im Modell ZT verwendet, da er bei gleicher Gruppengröße hinreichend robust gegen heterogene Varianzen ist[20)].

	K	D	T
V	0.937	0.067	0.906
Z	0.820	*	0.006

Tab. 5: P-Werte des F-Tests für (H_0, K_0)

Dies deutet darauf hin, daß in den Modellen ZT und QD signifikante Unterschiede bestehen, die im folgenden noch genauer untersucht werden.

Das Modell ZT

Heterogene Varianzen können beim Testen von Kontrasten im Gegensatz zu obiger Situation den F-Test verfälschen. Daher wurde hier ein modifizierter F-Test verwendet, der nur auf den geschätzten Gruppenvarianzen beruht, die miteinander verglichen werden. Außerdem wird die Zahl der Freiheitsgrade angepaßt[21)].

Hypothese	H_1: $v_1 = v_3$	H_2: $v_1 = v_4$	H_3: $v_2 = v_3$	H_4: $v_2 = v_4$	H_5: $v_3 = v_4$
P-Wert	0.142	0.001	0.174	0.001	0.254

Tab. 6: P-Werte des modifizierten F-Tests für Kontraste im Modell ZT

Dies deutet auf signifikante Unterschiede zwischen den Stufen 1 und 4 bzw. 2 und 4 hin.

Das Modell VD

Der übliche F-Test für Kontraste ergab folgendes Bild:

Hypothese	H_1: $v_1 = v_3$	H_2: $v_2 = v_3$	H_3: $v_2 = v_4$	H_4: $v_3 = v_4$	
P-Wert	0.037	0.015	0.174	0.214	

Tab. 7: P-Werte des F-Tests für Kontraste im Modell VD

---

20) Vgl. Lindman (1991).
21) Vgl. Lindman (1991).

Dies deutet auf signifikante Unterschiede zwischen den Stufen 1 und 3 bzw. 2 und 3 hin.

Das Modell ZD

Da hier die beiden Standardannahmen nicht erfüllt sind und die Abweichung von der Normalverteilung speziell in der Wölbung ($\gamma = 4.245$), gegen die der F-Test relativ empfindlich ist[22)], gravierend ist, wurde das entsprechende nichtparametrische Verfahren, der Kruskal-Wallis-Test, verwendet. Er ergab für (H_0, K_0) den P-Wert 0.0382, was auf signifikante Unterschiede zwischen den Stufen schließen läßt. Bei der Untersuchung der Kontraste mit dem Tukey-Kramer-Verfahren[23)] zeigte sich, daß auf dem 5%-Niveau nur die Stufen 3 und 4 signifikant verschieden sind (kritischer Wert: 3.628, Teststatistik: 4.082).

3.1.5 Korrelation zwischen Zeitdauer und Vollständigkeit der Lösung

Der Scatterplot (vgl. Abbildung 3) zeigt, daß keine starke Korrelation existiert und man sich auf die Untersuchung linearer Zusammenhänge beschränken kann. Der Pearson-Korrelationskoeffizient ergab $\rho_1 = 0.2495$. Der Spearman-Rangkorrelationskoeffizient als nicht-parametrische Vergleichsgröße hat den Wert $\rho_2 = 0.2082$. Dies deutet auf einen leichten, positiven linearen Zusammenhang hin. Die Frage, ob dieser als signifikant anzusehen ist, führt auf das Testen der Hypothesen H_i: $\rho_i = 0$ gegen K_i: $\rho_i \neq 0$, $i = 1, 2$. Da die beiden Größen Zeitdauer und Vollständigkeit als normalverteilt angesehen werden können (die P-Werte der jeweiligen Lilliefors-Tests sind größer als 0.2), läßt sich der entsprechende t-Test für (H_1, K_1) anwenden. Er liefert den P-Wert 0.098. Der zugehörige nichtparametrische Spearman-Test für (H_2, K_2) hat den P-Wert 0.17. Der positive lineare Zusammenhang läßt sich daher als signifikant ansehen, wenngleich er eher schwach ist.

3.2 Interpretation der Ergebnisse

Mit Hilfe der im vorhergehenden Abschnitt beschriebenen statistischen Auswertungen konnten einige Aussagen über die Zusammenhänge zwischen den unabhängigen Variablen (Einflußfaktoren) und den abhängigen Variablen (Zeit bzw. Vollständigkeit der Lösung) gewonnen werden.

---

22) Vgl. Lindman (1991).
23) Vgl. Sachs (1992).

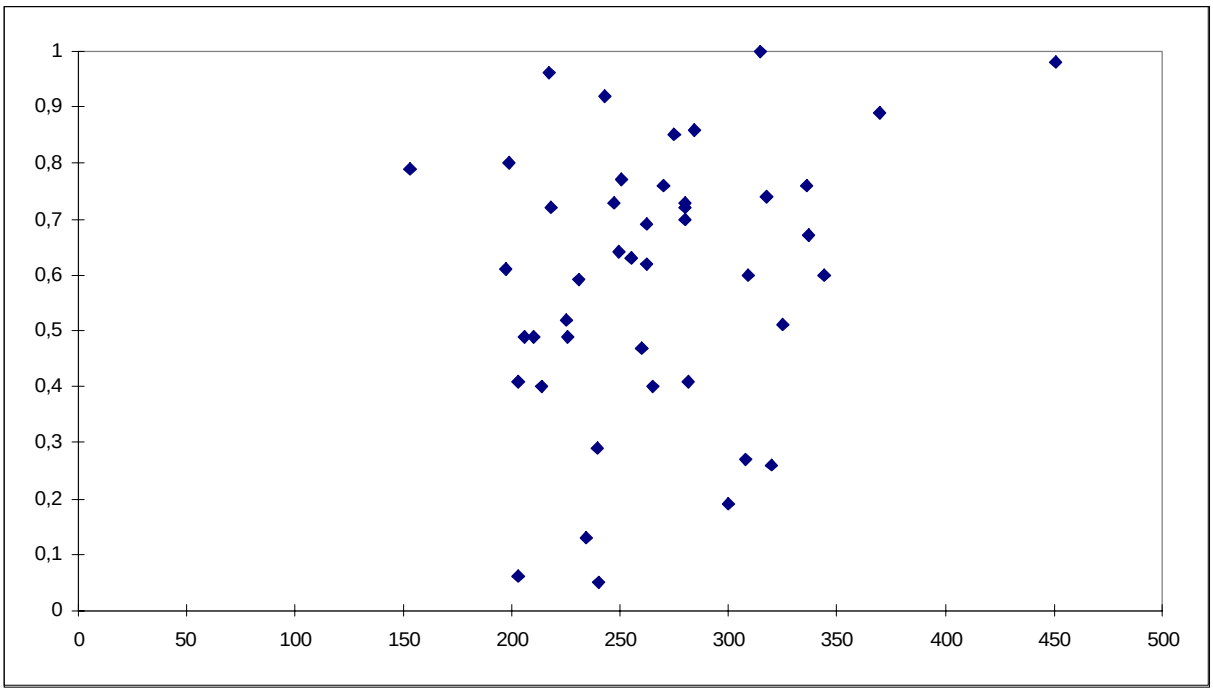

Abb. 3: Zusammenhang zwischen Bearbeitungszeit [min] (horizontal) und Vollständigkeit der Lösung [%] (vertikal)

Auswirkungen der Verfügbarkeit von Tages- bzw. Testdaten

Zwischen Ausprägung 1 („keine“) und Ausprägung 4 („Testdaten mit minimaler Mehrfachbedingungsüberdeckung von 89,4 %“) sowie zwischen Ausprägung 2 („vier Datensätze aus den Tagesdaten als Ausdruck“) und Ausprägung 4 des Merkmals Verfügbarkeit von Tages- bzw. Testdaten wurden bezüglich der Wirkung auf die Zeit (für die Lösung der Aufgabe) signifikante Unterschiede erkannt. Allerdings wurde der zuvor angenommene Zusammenhang einer Verkürzung der Bearbeitungszeit durch die Bereitstellung von Testdaten (im Fall von Ausprägung 4) nicht nachgewiesen, sondern der gegenteilige Effekt einer Verlängerung der Bearbeitungszeit. Die Auswirkungen auf die Vollständigkeit der Lösung sind nicht signifikant und konnten somit nicht belegt werden.

Die Vpn haben sich zwar offensichtlich mit den Testdaten „auseinandergesetzt“, konnten dadurch allerdings keine verwertbaren Informationen erhalten. Hypothetisch könnte man aus dieser Erkenntnis folgern, daß die Verfügbarkeit von Tages- bzw. Testdaten nur dann einen Produktivitätszuwachs hervorruft, wenn Werkzeuge zur komfortablen Nachverfolgung des Datenflusses innerhalb der Programme zur Verfügung stehen. Der in den Experimenten verwendete Debugger ermöglicht zwar den schrittweisen Ablauf eines Programms bei gleichzeitiger Kontrolle der Inhalte zuvor selektierter Variablen, ist allerdings unkomfortabel und stößt bei der Darstellung größerer Recordvariablen an die Grenzen der Übersichtlichkeit. Die Vpn wurden in der den Experimenten vorangehenden Schulung in die Benutzung des Werkzeugs eingear-

beitet, es ist aber nicht auszuschließen, daß noch mehr Erfahrung im Umgang mit dem Debugger letztendlich doch zu Produktivitätssteigerungen bei Verfügbarkeit von Tages- bzw. Testdaten führt.

Auswirkungen der Dokumentiertheit der Datenstrukturen

Zwischen Ausprägung 1 („ausschließlich nichtsprechende Bezeichner“) und Ausprägung 3 („ausschließlich sprechende Bezeichner“) sowie zwischen Ausprägung 2 („zum Teil sprechende Bezeichner“) und Ausprägung 3 des Merkmals Dokumentiertheit der Datenstrukturen wurden bezüglich der Wirkung auf die Vollständigkeit der Lösung signifikante Unterschiede erkannt. Die Vollständigkeit der Lösung stieg erwartungsgemäß mit einer besseren Dokumentiertheit der Datenstrukturen. Die Auswirkungen einer externen Dokumentation der Datenstrukturen (Ausprägung 4) zeigen sich in einer deutlichen Verkürzung der Bearbeitungszeit gegenüber Ausprägung 3. Obwohl nicht alle Einzelvariationen einen signifikanten Effekt zeigten, läßt sich zusammenfassen, daß sich Dokumentiertheit der Datenstrukturen in der erwarteten Weise ausgewirkt hat. Eine bessere Dokumentiertheit führt zu einer kürzeren Bearbeitungszeit und zu einer vollständigeren Lösung.

Auswirkungen der Quelltext-Komplexität

Die Tatsache, daß signifikante Auswirkungen der Quelltext-Komplexität weder auf die Zeit noch auf die Vollständigkeit der Lösung nachgewiesen werden konnten, deutet darauf hin, daß die Nachdokumentation der Datenelemente von diesem Einflußfaktor unabhängig ist. Ein Grund könnte darin liegen, daß die Versuchspersonen den Quelltext mit Hilfe des zur Verfügung gestellten Werkzeugs nach einzelnen Datenelementen durchsucht haben und auf diese Weise die Komplexität nahezu „überlesen“ haben. Um Redundanzen zwischen einzelnen Datenelementen, die von einem Programm verarbeitet werden, erkennen zu können, reicht grundsätzlich das Auffinden des Bezeichners in einem Vergleich (z.B. „IF FS211 = DPKN“). Es ist jedoch möglich, daß sich die Komplexität signifikant auswirkt, wenn die untersuchten Datenelemente bzw. die verarbeiteten Daten im Quelltext häufiger verändert bzw. zerlegt und in Hilfsvariablen weiterverarbeitet werden und somit eine detaillierte Untersuchung des Quelltexts unumgänglich ist.

Zusammenhang zwischen Zeit und Vollständigkeit der Lösung

Zwischen den beiden abhängigen Variablen wurde vor Durchführung der Experimente ein positiver Zusammenhang erwartet, der (zumindest schwach ausgeprägt) auch nachgewiesen werden konnte. Die Überlegung war, daß eine Vpn, die die Aufgabe vergleichsweise früh als erfüllt ansieht, eine unvollständigere Lösung erarbeitet als eine Vpn, die mehr Zeit aufwendet.

Eventuell ist der erwartete Zusammenhang deshalb nur schwach ausgeprägt, weil insbesondere bei den etwas sorgfältigeren Vpn (d.h. vollständigere Lösung) beobachtet werden konnte, daß sie nach Bearbeitung der Aufgabe noch einige Zeit mit der Überprüfung ihrer Ergebnisse verbracht haben.

4 Ausblick

Die im vorliegenden Aufsatz beschriebenen Experimente stellen einen ersten Versuch dar, Zusammenhänge zwischen Einflußfaktoren und den Kosten von Reengineering-Projekten im Rahmen des integrationsorientierten Reengineering herzustellen. Die gewonnenen Erkenntnisse sind auch auf das „konventionelle“ Reengineering übertragbar, da das Daten-Reengineering auch dort eine Rolle spielt. Bezogen auf ein Wirkungsmodell, daß die Zusammenhänge zwischen allen Einflußfaktoren und den Kosten enthält, konnten die Experimente nur einen sehr kleinen Teil abdecken. In dieser Richtung müßten zum einen mehr Experimente der geschilderten Art durchgeführt werden, zum anderen müßten aber auch empirische Untersuchungen durchgeführt werden, die auf Erfahrungswerten aus der betrieblichen Praxis basieren (Feldstudien). Voraussetzung für umfassende empirische Studien ist allerdings ein hinreichend detailliertes Modell, das alle zu messenden Einflußfaktoren enthält, damit die Datenerhebung strukturiert und zielgerichtet erfolgen kann.

Literatur

Banker, R.D., Datar, S.M., Kemerer, C.F., Zweig, D.: Software Complexity and Maintenance Costs; Communications of the ACM 36 (1993) 11, S. 81-94.

Boehm, B.W., Papaccio, P.N.: Understanding and Controlling Software Costs; IEEE Transactions on Software Engineering 14 (1988) 10, S. 1462-1477.

Chikofsky, E.J., Cross , J.H.: Reverse Engineering and Design Recovery: A Taxonomy; IEEE Software 7 (1990) 1, S. 13-17.

Eicker, S., Jung, R.: Wirtschaftlichkeit von Software-Reengineering-Projekten; in: v. Dobschütz, L., Kisting, J., Schmidt, E. (Hrsg.): IV-Controlling in der Praxis, Wiesbaden 1994, S. 113-131.

Eicker, S., Jung, R., Kurbel, K.: Anwendungssystem-Integration und Verteilungsarchitektur aus der Sicht des Reengineering, Informatik - Forschung und Entwicklung 8 (1993) 2, S. 70-78.

Eicker, S., Kurbel, K., Pietsch, W., Rautenstrauch, C.: Einbindung von Software-Altlasten durch integrationsorientiertes Reengineering, Wirtschaftsinformatik 34 (1992) 2, S. 137-145.

Eicker, S., Schnieder, T.: Integrationsorientiertes Reengineering von Cobol-Altsystemen, Praxis der Informationsverarbeitung und Kommunikation 16 (1993) 3, S. 162-168.

Jung, R.: Wirtschaftlichkeitsbeurteilung der Modifikation von Software-Altsystemen: Grundlagen und Grenzen; in: Pietsch, W., Steinbauer, D. (Hrsg.): Reengineering und Wirtschaftlichkeit; erscheint 1995.

Jung, R.: Wirtschaftlichkeitsfaktoren beim integrationsorientierten Reengineering - Verteilungsarchitektur und Integrationsschritte aus ökonomischer Sicht; Arbeitsbericht Nr. 16 des Instituts für Wirtschaftsinformatik, Münster 1993.

Kurbel, K.: Integration-oriented Data Reengineering Supporting Long-term Information Systems Evolution and Business Process Reengineering; in: Ng, P.A., et al. (Eds.): Proceedings of ICSI '94, Third International Conference on Systems Integration, São Paulo, Brazil; Los Alamitos, CA 1994, S. 466-475.

Lehner. F.: Softwarewartung - Management, Organisation und methodische Unterstützung; München, Wien 1991.

Lindmann, H.R.: Analysis of variance in experimental design; Berlin u.a. 1991.

Müller-Böling, D., Klandt, H.: Methoden der empirischen Forschung; Berlin u.a. 1994.

Sachs, L.: Angewandte Statistik, 7. Aufl.; Berlin u.a. 1992.

Anhang

Anhang A: Beispielprogramm

```
*********************************************************************
       IDENTIFICATION DIVISION.
*********************************************************************
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT s-dat ASSIGN TO "st.txt"
              ORGANIZATION LINE SEQUENTIAL.
           SELECT b-dat ASSIGN TO "be.txt"
              ORGANIZATION LINE SEQUENTIAL.

*********************************************************************
       DATA DIVISION.
       FILE SECTION.
       FD  s-dat.
       01  s-satz.
           05  artikel.
               10  art-name     PIC X(20).
               10  art-nr       PIC X(06).
           05  lagerinfo.
               10  lager-ort    PIC X(15).
               10  lager-best   PIC 9(04).
       FD  b-dat.
       01  b-satz            PIC X(15).
       WORKING-STORAGE SECTION.
       01  sq4500               PIC X(15).
       01  sf  REDEFINES  sq4500.
           05  FILLER           PIC X(06).
           05  sf002            PIC X(06).
           05  sf003            PIC 9(03).
       01  bs               PIC 9(01).
       01  tmp.
           05  gr               PIC X(03).
           05  FILLER           PIC X(01).
           05  be               PIC X(16).
       01  zerlegung.
           05  herkunft         PIC X(01).
           05  klartextname     PIC X(20).

*********************************************************************
       PROCEDURE DIVISION.
           OPEN I-O s-dat.
           OPEN INPUT b-dat.
           MOVE 1 TO bs.
           PERFORM testen UNTIL bs = 9.
           GO TO ende.
       testen.
           PERFORM b-lesen.
           IF bs = 1 PERFORM schreiben.
       schreiben.
           DISPLAY "Suche...".
           PERFORM s-lesen UNTIL art-nr = sf002.
           ADD sf003 TO lager-best.
           REWRITE s-satz.
       s-lesen.
           READ s-dat.
           MOVE art-name TO tmp.
           DISPLAY "Artikelgruppe: "  gr.
           UNSTRING be DELIMITED BY '-' INTO
              herkunft
              klartextname.
           IF herkunft = 'F' PERFORM fremd ELSE PERFORM eigen.
       b-lesen.
           READ b-dat INTO sq4500 AT END MOVE 9 TO bs.
           IF bs = 1 DISPLAY "Zugang: " b-satz.
       fremd.
           DISPLAY "fremdbezogenes Teil: " klartextname.
       eigen.
           DISPLAY "eigengefertigtes Teil: " be.
       ende.
           CLOSE s-dat.
           CLOSE b-dat.
           STOP RUN.
```

Anhang B: Vorgehensweise

Die Vorgehensweise, die aus insgesamt sieben Teilschritten besteht, zielt auf das sukzessive Verbessern von aus dem Programm ableitbaren Informationen über die Datenelemente ab. Zu diesem Zweck wurde die Vorgehensweise in Formulare abgebildet, die anhand von Fragen und Feldern für entsprechende Antworten die Lösung der Aufgabe strukturiert. Im folgenden werden die einzelnen Schritte der Vorgehensweise anhand des im Anhang A dargestellten Beispielsprogramms kurz erläutert.

1. *Grobe Kategorisierung der Programms*. In diesem ersten Schritt sollten die Testpersonen ermitteln, ob es sich bei dem vorliegenden Programm um ein Batch-Programm oder um ein Dialog-Programm handelt. Diese Unterscheidung ist erforderlich, da in den beiden Klassen unterschiedliche Schnittstellen zur „Außenwelt“ existieren (Benutzereingaben einerseits, Dateieingaben andererseits), die eine differenzierte Untersuchung des Datenflusses bedingen.

2. *Kategorisieren der Dateien*. Für jede Datei der File Section ist die Dateiorganisation, die Zugriffsart sowie die Zugriffsform zu bestimmen. Die Dateiorganisation ist dem File-Control-Eintrag der Environment Division zu entnehmen. Mögliche Ausprägungen sind „sequentiell“, „relativ“ und „index-sequentiell“. Die Zugriffsart gibt an, ob die Datei „lesend“, „schreibend“ oder „lesend/schreibend“ verarbeitet wird. Die Ermittlung der Zugriffsform („sequentiell“, „wahlfrei“ oder „dynamisch“), die die tatsächliche Nutzung der Dateien durch das Programm beschreibt, erfordert eine erste Untersuchung des Quelltextes. Es ist zum Beispiel denkbar, daß ein Programm eine index-sequentielle Datei lediglich sequentiell nutzt, indem beim ersten Datensatz begonnen und dann jeweils der nächste verarbeitet wird.

3. *Ermittlung der groben Datensatzstruktur*. In diesem Schritt werden die Informationen über die Datenelemente, die den Einträgen der File Section zu entnehmen sind, verfeinert, indem der Quelltext nach den Anweisungen READ INTO und WRITE FROM durchsucht wird. Sofern ein Datensatz beispielsweise mit Hilfe von READ INTO in eine Variable der Working-Storage Section gelesen wird und diese Variable eine feinere Struktur als die entsprechende Variable der File Section aufweist, kann durch Ersetzung des entsprechenden Deklarationsteils eine Verfeinerung der Deklaration erreicht werden.

4. *Grobe Interpretation der Datensatzstrukturen*. In diesem Schritt sind die bisher wiedergewonnenen Informationen über die Datensätze bzw über die Datenelemente zu interpretieren. Zu diesem Zweck können Bezeichner verwendet werden, sofern sie selbsterklärend sind.

5. *Ableitung der Grobstruktur des Programms*. In diesem Schritt wird die Grobstruktur des Programms ermittelt und mit Hilfe eines Programmablaufplans (PAP) abgebildet. Dieser Schritt dient dazu, den Testpersonen einen Überblick über das zu untersuchende Programm und die wichtigsten Abläufe zu geben. Er dient als Vorbereitung der restlichen Schritte.

6. *Untersuchung der Programmlogik bezüglich der Ein- und Ausgabedateien*. Anhand der Programmlogik sollen die Testpersonen untersuchen, ob eine Sortierung der Dateien vorausgesetzt wird und welches eventuell jeweils das Schlüsselfeld ist. Das im Anhang dargestellte Programm setzt beispielsweise eine Sortierung der Datei *s-dat* nach dem Datenelement *art-nr* und eine gleichartige Sortierung der Datei *b-dat* nach dem Datenelement *sf002* (als Verfeinerung von *b-satz*) voraus, wobei die beiden Datenelemente Schlüssel der jeweiligen Datei sind. Darüber hinaus muß zu jedem Datensatz der Datei *b-dat* ein entsprechender Datensatz (Schlüsselgleichheit) in der Datei *s-dat* existieren. Diese Informationen kann aus der Programmlogik gewonnen werden.

7. *Verbesserung der Datensatzbeschreibungen*. In diesem letzten und wichtigsten Schritt werden die Datensatzbeschreibungen weiter verfeinert. Zu diesem Zweck muß der Datenfluß - bezogen auf jedes Datenelement der verfeinerten File Section - „durch“ den Quelltext genau untersucht werden. Wichtig sind in diesem Zusammenhang zum Beispiel Vergleiche mit anderen Variablen oder auch Bildschirmausgaben, die Rückschlüsse auf die Semantik der Datenelemente zulassen.

Anhang C: Lösungsformulare

Vorgehensweise
für die Ableitung von detaillierten Datensatzbeschreibungen
aus einem Cobol-Altsystem

Schritt 1: Grobe Kategorisierung des Programms

Handelt es sich um ein Batch- oder um ein Dialogprogramm? ❑ Batch ❑ Dialog

(Suche nach dem Wort ACCEPT, um Eingaben von der Tastatur zu identifizieren)

Schritt 2: Kategorisieren der Dateien: Dateiorganisation, Zugriffsform, Zugriffsart

Welche Dateien werden von dem Programm verwendet? Beschreiben Sie jeweils die Dateiorganisation (sequentiell/relativ/index-sequentiell), die Zugriffsform (sequentiell/wahlfrei/dynamisch) sowie die Zugriffsart (lesend/ schreibend/beides)! Dokumentieren Sie die Ergebnisse in der folgenden Tabelle!

(Untersuchung des FILE CONTROL-Eintrags in der INPUT-OUTPUT SECTION; untersuchen Sie die OPEN-Statements, um jeweils die Zugriffsart zu bestimmen)

Dateiname (logisch)	Dateiorganisation	Zugriffsform	Zugriffsart
Beispiel: *s-dat* *b-dat*	*sequentiell* *sequentiell*	*sequentiell* *sequentiell*	*lesend/schreibend* *lesend*

Schritt 3: Ermitteln der tatsächlichen Datensatzstruktur

Ermitteln Sie die tatsächlichen Datensatzstrukturen, die den Dateien zugrundeliegen. Untersuchen Sie dazu zunächst den FD-Eintrag in der FILE SECTION. Weitere Hinweise auf die tatsächliche Datensatzstruktur können auch in der WORKING-STORAGE SECTION zu finden sein. Die Verbindungen zwischen den Datensatzbeschreibungen aus der FILE SECTION und Records aus der WORKING-STORAGE SECTION können im Programmtext anhand der folgenden Statements identifiziert werden: READ INTO, WRITE FROM. Sofern Testdaten bzw. Tagesdaten verfügbar sind, können auch diese Hinweise auf die tatsächliche Datensatzstruktur enthalten. Beispiel:

```
01 s-satz
   05  artikel.
       10  art-name     PIC  X(20).
       10  art-nr       PIC X(06).
   05  lagerinfo.
       10  lager-ort    PIC X(15).
       10  lager-best   PIC 9(04).

01 b-satz
   05  FILLER           PIC X(06).
   05  sf002            PIC X(06).
   05  sf003            PIC 9(03).
```

Dokumentieren Sie das Ergebnis in einer eigenen Datei, die die tatsächlichen Datensatzstrukturen enthält. Sie können zu diesem Zweck mit Hilfe von Copy+Paste Teile des Programms einfügen. Sehen Sie zwischen den einzelnen Zeilen der Beschreibung mindestens vier Leerzeilen vor, da dort später (in Schritt 7) noch handschriftliche Eintragungen vorgenommen werden. Drucken Sie die Beschreibung anschließend aus!

Schritt 4: Grobe Interpretation der Datensatzstrukturen

Geben Sie eine erste grobe Interpretation der extrahierten Datensatzstrukturen an. Schätzen Sie insbesondere die Bedeutung der Datensätze im Anwendungsumfeld ab. Werten Sie dazu die Kommentare und eventuell vorliegende weitere Dokumentationen der Datensatzbeschreibungen sowie Test- bzw. Tagesdaten aus.

Beispiel: Die Datei s-dat enthält offensichtlich Artikeldaten, die aus Informationen über einen Artikel und den zugehörigen Lagerort bestehen. Die Datei b-dat besteht aus Datensätzen mit drei Feldern, deren Verwendung aus den Deklarationen und Kommentaren nicht ersichtlich ist.

Schritt 5: Ableitung der Grobstruktur des Programms

Ermitteln Sie die grobe Struktur des Programms und stellen Sie diese in Form eines Programmablaufplans (PAP) dar.

Schritt 6: Untersuchung der Programmlogik bzgl. der Ein- und Ausgabedateien

Untersuchen Sie, welche Anforderungen durch die Programmlogik an die Beschaffenheit der Ein- und Ausgabedateien gestellt werden.

Wird z.B. eine Sortierung der Sätze vorausgesetzt? Welches ist ggf. das Schlüsselelement? Dokumentieren Sie Ihre Ergebnisse in der folgenden Form:

Datei _____ Sortiert ? ❑ Nein ❑ Ja, Schlüsselfeld(er) _____

Datei _____ Sortiert ? ❑ Nein ❑ Ja, Schlüsselfeld(er) _____

Datei _____ Sortiert ? ❑ Nein ❑ Ja, Schlüsselfeld(er) _____

Datei _____ Sortiert ? ❑ Nein ❑ Ja, Schlüsselfeld(er) _____

Datei _____ Sortiert ? ❑ Nein ❑ Ja, Schlüsselfeld(er) _____

In welcher Beziehung stehen die einzelnen Dateien zueinander? Dokumentieren Sie Ihre Ergebnisse in der folgenden Form (auf der nächsten Seite):

Beispiel: Zu jedem Satz aus der Datei b-dat muß genau ein Satz in der Datei s-dat vorhanden sein, wobei korrespondierende Elemente durch identische Werte für artnr und sf002 identifiziert werden. Für den Fall, daß zu einem Satz aus b-dat kein Satz in der Datei s-dat existiert, ist keine Behandlung implementiert, d.h., die Programmausführung würde mit einer Fehlermeldung abbrechen.

Schritt 7: Verbesserung der Datensatzbeschreibungen

Bestimmen Sie die Bedeutungen sowie - wenn möglich - die Wertemengen der Datenelemente aus der FILE SECTION, die durch das Programm auch tatsächlich benutzt werden (Ergebnis aus Schritt 3)! Verfolgen Sie dazu mit Hilfe der Microsoft Cobol-Umgebung oder auch anhand des Listings den Steuerfluß jeweils bezogen auf ein Datenelement. Benutzen Sie, sofern vorhanden, auch Tages- und Testdaten. Ermitteln Sie dabei auch semantisch redundante Datenelemente, d.h. Elemente die Daten mit identischer Bedeutung aufnehmen.

Dokumentieren Sie Ihre Ergebnisse in dem Ausdruck, den Sie in Schritt 3 erstellt haben. Beispiel:

```
01 s-satz
   05  artikel.
       10  art-name     PIC  X(20).
          Name eines Artikels, der evtl. am Lager vorrätig ist;
          dieses Feld spaltet sich weiter auf in:
          gr       PIC X(03)    enthält die Artikelgruppe
          FILLER   PIC X(01)    unbekannt (vermutlich Trennung)
          BE       PIC X(16)    wird wie folgt belegt:
                                [F-] klartextname
                                Der Prefix wird nur bei fremd-
                                bezogenen Teilen benutzt. Bei
                                eigengefertigten Teilen enthält
                                BE nur den Klartextnamen.
       10  art-nr       PIC X(06).
          Nummer eines Artikels, der evtl. am Lager vorrätig
          ist; redundant zu sf002
   05  lagerinfo.
       10  lager-ort    PIC X(15).
          Lagerort eines Artikels
       10  lager-best   PIC 9(04).
          Lagerbestand eines Artikels

01 b-satz
   05  FILLER           PIC X(06).
      Verwendung unbekannt
   05  sf002            PIC X(06).
      Nummer eines Artikels, der dem Lager zugegangen ist;
      redundant zu art-nr
   05  sf003            PIC 9(03).
      Höhe des Lagerzugangs mit der Nummer sf002
```

Arbeitsberichte des Instituts für Wirtschaftsinformatik

Nr. 1 Bolte, Ch., Kurbel, K., Moazzami, M., Pietsch, W.: Erfahrungen bei der Entwicklung eines Informationssystems auf RDBMS- und 4GL-Basis; Februar 1991.

Nr. 2 Kurbel, K.: Das technologische Umfeld der Informationsverarbeitung - Ein subjektiver 'State of the Art'-Report über Hardware, Software und Paradigmen; März 1991.

Nr. 3 Kurbel, K.: CA-Techniken und CIM; Mai 1991.

Nr. 4 Nietsch, M., Nietsch, T., Rautenstrauch, C., Rinschede, M., Siedentopf, J.: Anforderungen mittelständischer Industriebetriebe an einen elektronischen Leitstand - Ergebnisse einer Untersuchung bei zwölf Unternehmen; Juli 1991.

Nr. 5 Becker, J., Prischmann, M.: Konnektionistische Modelle - Grundlagen und Konzepte; September 1991.

Nr. 6 Grob, H.L.: Ein produktivitätsorientierter Ansatz zur Evaluierung von Beratungserfolgen; September 1991.

Nr. 7 Becker, J.: CIM und Logistik; Oktober 1991.

Nr. 8 Burgholz, M., Kurbel, K., Nietsch, Th., Rautenstrauch, C.: Erfahrungen bei der Entwicklung und Portierung eines elektronischen Leitstands; Januar 1992.

Nr. 9 Becker, J., Prischmann, M.: Anwendung konnektionistischer Systeme; Februar 1992.

Nr. 10 Becker, J.: Computer Integrated Manufacturing aus Sicht der Betriebswirtschaftslehre und der Wirtschaftsinformatik; April 1992.

Nr. 11 Kurbel, K., Dornhoff, P.: A System for Case-Based Effort Estimation for Software-Development Projects; Juli 1992.

Nr. 12 Dornhoff, P.: Aufwandsplanung zur Unterstützung des Managements von Softwareentwicklungsprojekten; August 1992.

Nr. 13 Eicker, S., Schnieder, T.: Reengineering; August 1992.

Nr. 14 Erkelenz, F.: KVD2 - Ein integriertes wissensbasiertes Modul zur Bemessung von Krankenhausverweildauern - Problemstellung, Konzeption und Realisierung; Dezember 1992.

Nr. 15 Horster, B., Schneider, B., Siedentopf, J.: Kriterien zur Auswahl konnektionistischer Verfahren für betriebliche Probleme; März 1993.

Nr. 16 Jung, R.: Wirtschaftlichkeitsfaktoren beim integrationsorientierten Reengineering: Verteilungsarchitektur und Integrationsschritte aus ökonomischer Sicht; Juli 1993.

Nr. 17 Miller, C., Weiland, R.: Der Übergang von proprietären zu offenen Systemen aus Sicht der Transaktionskostentheorie; Juli 1993.

Nr. 18 Becker, J., Rosemann, M.: Design for Logistics - Ein Beispiel für die logistikgerechte Gestaltung des Computer Integrated Manufacturing; Juli 1993.

Nr. 19 Becker, J., Rosemann, M.: Informationswirtschaftliche Integrationsschwerpunkte innerhalb der logistischen Subsysteme - Ein Beitrag zu einem produktionsübergreifenden Verständnis von CIM; Juli 1993.

Nr. 20 Becker, J.: Neue Verfahren der entwurfs- und konstruktionsbegleitenden Kalkulation und ihre Grenzen in der praktischen Anwendung; Juli 1993.

Nr. 21 Becker, K., Prischmann, M.: VESKONN - Prototypische Umsetzung eines modularen Konzepts zur Konstruktionsunterstützung mit konnektionistischen Methoden; November 1993

Nr. 22 Schneider, B.: Neuronale Netze für betriebliche Anwendungen: Anwendungspotentiale und existierende Systeme; November 1993.

Nr. 23 Nietsch, T., Rautenstrauch, C., Rehfeldt, M., Rosemann, M., Turowski, K.: Ansätze für die Verbesserung von PPS-Systemen durch Fuzzy-Logik; Dezember 1993.

Nr. 24 Nietsch, M., Rinschede, M., Rautenstrauch, C.: Werkzeuggestützte Individualisierung des objektorientierten Leitstands ooL; Dezember 1993.

Nr. 25 Meckenstock, A., Unland, R., Zimmer, D.: Flexible Unterstützung kooperativer Entwurfsumgebungen durch einen Transaktions-Baukasten; Dezember 1993.

Nr. 26 Grob, H. L.: Computer Assisted Learning (CAL) durch Berechnungsexperimente; Januar 1994.

Nr. 27 Kirn, St., Unland, R. (Hrsg.): Tagungsband zum Workshop "Unterstützung Organisatorischer Prozesse durch CSCW". In Kooperation mit GI-Fachausschuß 5.5 "Betriebliche Kommunikations- und Informationssysteme" und Arbeitskreis 5.5.1 "Computer Supported Cooperative Work", Westfälische Wilhelms-Universität Münster, 4.-5. November 1993

Nr. 28 Kirn, St., Unland, R.: Zur Verbundintelligenz integrierter Mensch-Computer-Teams: Ein organisationstheoretischer Ansatz; März 1994.

Nr. 29 Kirn, St., Unland, R.: Workflow Management mit kooperativen Softwaresystemen: State of the Art und Problemabriß; März 1994.

Nr. 30 Unland, R.: Optimistic Concurrency Control Revisited; März 1994.

Nr. 31 Unland, R.: Semantics-Based Locking: From Isolation to Cooperation; März 1994.

Nr. 32 Meckenstock, A., Unland, R., Zimmer, D.: Controlling Cooperation and Recovery in Nested Transactions; März 1994.

Nr. 33 Kurbel, K., Schnieder, T.: Integration Issues of Information Engineering Based I-CASE Tools; September 1994.

Nr. 34 Unland, R.: TOPAZ: A Tool Kit for the Construction of Application Specific Transaction; November 1994.

Nr. 35 Unland, R.: Organizational Intelligence and Negotiation Based DAI Systems - Theoretical Foundations and Experimental Results; November 1994.

Nr. 36 Unland, R., Kirn, S., Wanka, U., O'Hare, G.M.P., Abbas, S.: AEGIS: AGENT ORIENTED ORGANISATIONS; Februar 1995.

Nr. 37 Jung, R., Rimpler, A., Schnieder, T., Teubner, A.: Eine empirische Untersuchung von Kosteneinflußfaktoren bei integrationsorientierten Reengineering-Projekten; März 1995.