erschienen: *Grün, O.; Heinrich, L.J.* (Hrsg): Empirische Forschung in der Wirtschaftsinformatik, Springer, Wien - New York 1997, 105-118.

Die Gewinnung korrekter Daten aus manuellen Aufschreibungen - Empirische Ermittlung eines Erfassungssystems

Thorsten Spitta

Universität Bielefeld, Fakultät für Wirtschaftswissenschaften
Postfach 10 01 31, D-33505 Bielefeld

Inhalt

Zusammenfassung

Die korrekte Erfassung von Aufwandsdaten ist ein stark vernachlässigtes betriebliches und forschungsmethodisches Problem. Die Entwicklung, das Design und das Einsatzkonzept eines Systems für die manuelle Erfassung von Aufwandsdaten werden dargestellt. Forschungsmethodisch ist ein solches System ein Meßinstrument, das die Korrektheitsanforderungen an die zu erfassenden Daten umsetzen muß. Das System wurde evolutionär unter Auswertung der damit erfaßten Daten entwickelt. Es werden Auswertungsbeispiele aus der Datenbasis über 6½ Jahre gezeigt, die durch Kontierungen von 40 Entwicklern an fünf Standorten entstanden ist.

Abstract

Collecting effort data is a strongly neglected problem of every day's practice and research. The development, design and application concept of a system for manual effort collection is presented. Research methodology calls such a system a measurement tool, which has to realize the correctness requirements on the data. Examples of results from a database produced by 40 developers in five locations over 6½ years are showed.

1 Gegenstand und Stand der Forschung

Manuell erfaßte Daten sind ein wichtiges Problem der Forschung und der betrieblichen Praxis. So legen drei der acht faktenerhebenden Untersuchungen des vorliegenden Bandes[1] Aufwandsdaten zugrunde, die sich nur durch Selbstaufschreibungen gewinnen lassen. Geschäftsprozesse der Art „Einzelfertigung" (bestimmte Produktionen, Entwicklung, Marktforschung, Softwareerstellung und -pflege) lassen sich nur beherrschen, wenn man die einzelnen Objekten, Vorgängen und Zeiträumen zugeordneten Personalaufwände kennt. Forschung und tatsächlicher Betrieb interferieren sogar bei solchen Fragen, die nicht aus primär für die Forschung erhobenen Daten beantwortet werden können. Viele Daten wissenschaftlicher Sekundärerhebungen wurden primär für Zwecke des Controlling erhoben.

Zum Thema *manuelle Datenaufschreibungen* gibt es nur eine Veröffentlichung von Basili und Weiss [BaWe84], deren Schwerpunkt die Validität von Änderungsmeldungen bei der Wartung von Software der NASA und der US-Marine ist. Manuell erhobene Daten aus dem IS-Bereich von Unternehmen werden allgemein als unsystematisch, fehlerbehaftet und firmenspezifisch bemängelt [Seli86; Grie87; Lehn91].

Der folgende Beitrag will zwei methodische Fragen beantworten:

- Wie kann man ein verläßliches und praktisch verwendbares Erfassungssystem entwickeln? Die Frage der explorativen Entwicklung eines solchen wird knapp behandelt.
- Wie muß das Design eines solchen Systems aussehen, damit man korrekte Daten erhält, die in Selbstaufschreibungen entstehen? Den Schwerpunkt des Aufsatzes bilden das Datendesign und das Einsatzkonzept mit Anwendungsbeispielen.

Der Beitrag beleuchtet die forschungsmethodischen Fragen desselben Systems, das in [Spit96] mit dem Schwerpunkt *Softwaremethodik* vorgestellt wird. Auf Details wird hier mehrfach verwiesen, um Überschneidungen zu vermeiden.

2 Forschungsmethode

Der Ablauf einer empirischen Untersuchung kann in fünf Stufen gesehen werden, wenn man von einer vorhandenen Fragestellung ausgeht [Atte75, Laat93]:

1. *Modellbildung* mit Ziel- und Hypothesenfindung,
2. Bildung und Validierung von Variablen, die die Dimensionen des Modells beschreiben und die Hypothesen quantifizieren (*Überprüfungsstrategie*),

[1] Vorläufig der Tagungsband vom März 1996 in Linz: „3"= Witte.., Gresse.., Hierholzer.

3. *Datenerhebung*,
4. Auswertung
5. *Interpretation* und *Folgerungen*.

Die Schritte laufen keineswegs streng sequentiell ab. Vielmehr besteht das Problem häufig darin, erst durch mehrfache Wiederholung der Schritte eins bis drei zu einem validen Modell mit sinnvollen Hypothesen und einem Meßinstrument für dessen Variable zu gelangen. In der einschlägigen sozialwissenschaftlichen Literatur wird von vielen Untersuchungen berichtet, bei denen das Modell erst explorativ entwickelt wurde, bevor die eigentliche Untersuchung durchgeführt werden konnte [Kerl73, Laat93, Roth93]. Dies ist auch hier der Fall. Es wird nicht eine Untersuchung aus bereits erhobenen Daten, sondern die empirische Entwicklung des Meßinstrumentes zu deren Erfassung dargestellt. Wegen des praxisbezogenen Vorgehens wird von einem *Erfassungssystem* gesprochen.

2.1 Modellentwicklung

Bei einem Datenerfassungssystem ist das Datendesign entscheidend, nicht die relativ triviale Implementierung. Dieses Design wurde Ende 1986 erstellt und als Prototyp implementiert. Es ist in [Spit89, 206ff] publiziert. Ab 2.1.1987 wurden damit Aufwandsdaten für alle Softwareprojekte und Wartungsaktivitäten eines Industrieunternehmens mit 200 MioDM Jahresumsatz erfaßt. Nach drei Monaten Pilotbetrieb durch 9 Entwickler an *einem* Standort wurde das System überarbeitet, indem präzisierte Integritätsbedingungen implementiert wurden. Die Korrektheit der Aufschreibungen wurde durch Wartungs- und Projektleiter ständig stichprobenartig überprüft. Die Struktur der Datenbasis wurde in den folgenden sechs Jahren mehrfach durch Attribute und Schlüsselwerte ergänzt, aber nicht strukturell geändert. Die in [Spit96] und hier gezeigten Daten haben diese ursprüngliche Struktur. Durch die Fusion dreier Firmen zu einem Konzern kam das System an anderen Standorten und in neuen Firmenkulturen zum Einsatz. Bis Mitte 1993 wurde es an *fünf* Standorten (drei Werke und zwei Softwarehäuser) von ca. 40 Entwicklern benutzt. Dies ist zwar keine zufällig ausgewählte Stichprobe, aber immerhin eine gewisse Streuung an empirischer Erprobung. Basierend auf den Erfahrungen der Benutzung (qualitative Schlüsse) und der Auswertung erfaßter Daten (quantitative Besetzung von Variablen) wurde ein endgültiges Design erstellt, das über die Fallanwendung hinaus allgemein empfohlen werden kann [Spit96].

Man kann die Entwicklungsmethodik als *Forschung durch Entwicklung* [SzMB81] ansehen oder ganz einfach als *inkrementelle Softwareentwicklung*, häufig irreführend *evolutionäres Prototyping* genannt. Der Unterschied der hier vorgestellten Studie zu einer sonstigen Systementwicklung ist, daß das System anhand der erfaßten Daten ständig daraufhin untersucht wurde, ob und wie es die Erfassung *korrekter* Daten unterstützt. Dies führte zur Veränderung von Varia-

blen und Wertebereichen (Modellgültigkeit) und zur Definition von Integritätsbedingungen (Datenkorrektheit). Insofern wurde oben von einem *Meßinstrument* gesprochen.

Die Entwicklung ist ein Beispiel dafür, wie bestimmte Anwendungen in realen Unternehmen, insbesondere im Bereich des Controlling, und wissenschaftliche Feldforschung interferieren können. Anliegen des Controlling im Betrieb wie eines Forschungsprojektes sind verläßliche Daten. Bei Längsschnittuntersuchungen, bei denen eine Datenerhebung für die wissenschaftliche Untersuchung zu aufwendig ist, wird man auf Sekundäranalysen angewiesen sein. Man möchte im hier vorliegenden Fall langfristig stabile Aussagen über Aufwandsverteilungen und Ereignisse wie Fehler oder Wartungsanforderungen gewinnen. Die betriebliche Berichterstattung für ein langjähriges Großprojekt benötigt ebensolche Daten.

2.2 Korrektheit von Daten

Korrektheit ist eine komplexe und kaum vollständig erfüllbare Eigenschaft empirischer Daten. Sie zerfällt in die Gütekriterien *Gültigkeit* (validity), *Zuverlässigkeit* (reliability) und *Repräsentativität* [Laat93, 56ff]. Letztere betrifft nur die Auswahl von Stichproben und spielt hier keine Rolle. **Gültig** sind die Variablen eines Modells, wenn sie messen, was sie messen sollen, also das Sachproblem adäquat abbilden. Dies spielt im nachfolgenden Design eine große Rolle bei dessen Struktur (*Ist die Beziehung Projekt - Teilprojekt hierarchisch?*) und den Wertebereichen (*Gibt es eine Phase Einführung oder nicht?*). **Zuverlässig** sind Daten streng genommen nur, wenn bei der Wiederholung eines Ereignisses unter kontrollierten Bedingungen identische Daten entstehen. Bei Aufwandsdaten bedeutet dies, daß Entwickler B ein gleichartiges Ereignis identischen Kategorien zuordnen muß, wie es A bereits getan hat. Zuverlässigkeit bedingt aber auch langfristige Stabilität. Ein gleichartiger Wartungsfall muß auch nach 5 Jahren als Wartung und nicht als Projekt eingestuft werden.

2.3 Datenerhebungsmethoden

Von den grundsätzlichen Datenerhebungsmethoden *(technische) Messung*, *Befragung*, *Inhaltsanalyse* und *Beobachtung* interessiert hier nur die letzte. Wenn ein Entwickler erfaßt, welchen Aufwand er für eine Aktivität benötigt hat, führt er eine *Selbstbeobachtung* durch, die wegen der damit verbundenen Manipulationsmöglichkeiten besonders kritisch zu sehen ist und in den Sozialwissenschaften auch so gesehen wird [Roth93, 127ff]. Aufgrund der hohen Zahl von Ereignissen (s. u.) bleibt jedoch keine andere Möglichkeit als *diese* Erhebungsmethode und die Hoffnung auf einen statistischen Ausgleich von Fehlern. Dazu müssen das Design und der Einsatz eines Erfassungssystems die Fehlerquellen der Methode berücksichtigen.

2.4 Fehlermöglichkeiten

Grundsätzlich sind zwei Fehlermöglichkeiten zu unterscheiden, *Meßfehler* und *Auswahlfehler*. Beide Fälle können als *Zufallsfehler* oder als *systematische Fehler* vorkommen [Laat93, 63ff]. **Zufallsfehler** lassen sich nie ausschließen, besonders nicht bei Zeitaufschreibungen. Allein in dem in [Spit96] betrachteten Zeitraum von 37 Monaten wurde die Verwendung von über 130.000 Stunden durch 25 Entwickler festgehalten. Der Zufallsanteil von Fehlern gleicht sich durch das Gesetz der großen Zahl aus, sowohl bei den erfaßten *Aufwänden in Stunden* (Meßfehler) als auch in den Zuordnungen zu Dimensionen, etwa *Projekt* oder *Wartung* (Auswahlfehler).

Problematischer gerade bei der Selbstbeobachtung sind **systematische Fehler**. Sie können zwei Ursachen haben:

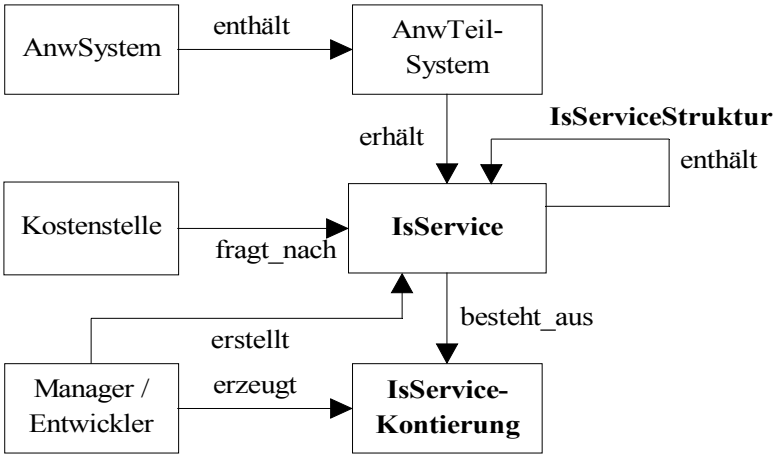
1. Dimensionen des Modells sind unscharf oder „zu scharf“. Dies muß das Design berücksichtigen. Typisch wären etwa eine falsche Zuordnung des Einführungsaufwandes von Projekten zur Wartung oder eine Differenzierung des Aufwandes in Verfahrensschritte.
2. Es wird bewußt falsch kontiert. Diese Fehlerart sei hier *interessegeleiteter Fehler* genannt. So etwa will ein Projektleiter im Rahmen seiner Schätzung bleiben und „verschiebt“ Aufwände in die Wartung oder ein im Vertrieb besonders „beliebter“ Entwickler belastet die Produktion mit Wartungsaufwand. Diese Fehlerart ist wegen der mangelnden Nachvollziehbarkeit der Daten bei Selbstbeobachtungen die eigentlich kritische. Ihr kann man nur bedingt durch das Design und das Einsatzkonzept für die Datengewinnung begegnen. Es darf jedoch vermutet werden, daß sich auch diese Fehler bei einem überlegten Einsatz ab ca. 10 Entwicklern statistisch ausgleichen.

Das folgende Design des Systems wird vor allem unter dem Blickwinkel betrachtet, ob und wie es die methodischen Aspekte umsetzt.

3 Design

Ein Datenerfassungssystem wird am günstigsten als Datenmodell entworfen. Mit *Datenmodell* ist jedoch nicht nur die recht plakative Ebene eines Entity-Relationship-Modells gemeint, sondern vor allem der Entwurf von Objekttypen auf der Ebene der Attribute mit exakten Integritätsbedingungen. Letztere stellen die formal definierbaren Korrektheitsanforderungen der Daten sicher, indem sie Wertebereiche und erlaubte Zustandsübergänge festschreiben. Die *Funktionen* für ein Erfassungssystem sind trivial und bedürfen keines expliziten Entwurfs. Hier genügt eine Aufzählung der datenändernden Funktionen. Im Gegensatz zu [Spit96] wird hier nur die minimale Fassung des Systems ohne Plankomponente gezeigt, die zur Erfassung der reinen Aufwandsdaten benötigt wird.

3.1 Struktur



Legende: → : 0, 1 : N - Beziehung; **Fettdruck** : Kern des Systems; <text> : Beziehungstyp, zu lesen in Pfeilrichtung *Abkürzungen*: Is : Informationssystem; Anw : Anwendungs~

Abb 1: Strukturmodell manueller Aufwands- und Ereigniserfassung

Die Semantik der Struktur des Designs ist folgende: *Is-Service(-Vorgänge)* {Projekt, Wartung, Betreuung} können beliebig tief hierarchisch gegliedert sein. Dies dient z. B. der Abbildung von Projekten und Teilprojekten, der Zuordnung von Ereignissen wie *Fehler* zu (Dauer-)Vorgängen wie etwa der Wartung eines Anwendungssystems. Die Hierarchie *Anwendungssystem* → *Anwendungs-Teilsystem* ist dagegen genau einstufig. Eine tiefere Schachtelung hat sich als unübersichtlich erwiesen. Zu einem Is-Service-Vorgang, den nur ein Manager anlegen kann, gibt es *Kontierungen*. Dies sind die eigentlichen Zeitaufschreibungen.

Eine betriebswirtschaftlich neutrale Klassifikation in *Anwendungssysteme* war im ursprünglichen Entwurf nicht enthalten. Sie ist in [Spit96] ausführlich begründet. Es hat sich gezeigt, daß die für die innerbetriebliche Leistungsverrechnung notwendige Kostenstellenstruktur langfristig nicht stabil ist.

3.2 Objekttypen

Die Präzisierung der im Strukturmodell gezeigten Objekttypen erfolgt nur für den in Abbild 1 fettgedruckten Kern des Systems. Zusätzlich erscheint eine Relation *Phasenübergang*, die zum Zeitpunkt einer Änderung des Attributes IsService.EntwSchr (*EntwicklungsSchritt*) automatisch erzeugt wird. Sie hält fest, wann ein Phasenübergang stattgefunden hat. Die Abkürzungen der Attributnamen am Schluß kennzeichnen Datentypen und damit Wertebereiche. Im einzelnen:

Relation	***Attribute***
IsService	(IsService#, VorgangsArt, Kostenstelle#, AnwTeilsystemCode#, Beschreibung, geschätzteAufwandsZeit, tatsAufwandsZeit, EntwicklSchr, strategRelevanz?, EingangsDat, WunschTermin, ZusageTermin, FertigstellTermin)
IsServiceStruktur	(Ober.IsService#, Unter.IsService#)
Phasenübergang	(IsService#, BeginnDat#, EntwicklSchr)
IsServiceKontierung	(IsService#, EntwicklerName#, Datum#, AufwandsZeit, TätigkeitsBes, EntwicklSchr, Kostenstelle#)

Syntax: <string># : Primärschlüssel; <string># : Fremdschlüssel; <string>? : Datentyp Bool
Abkürzungen (unterstrichen): Zeitraum; Schritt; Datum, synonym Termin; Beschreibung.

Tab. 1: Relationenmodell zur manuellen Aufwandserfassung

Für unser Thema spielen die Objekttypen *IsService* und *IsServiceKontierung* eine unterschiedliche Rolle. Abbild 1 zeigt, daß der erste von einem Manager gepflegt und nur der zweite vom Entwickler erzeugt wird. Dies ist wichtig für das Korrektheitproblem: Die Masse der von Entwicklern erzeugten Daten umfaßt nur ganz wenige Attribute, die verfälscht werden könnten, und zwar *IsService#, AufwandsZeit*, *TätigkeitsBeschreibung*, *EntwicklungsSchritt* und *Kostenstelle#*.

Die in Abschnitt 3.3 erläuterten Attribute des Objekttyps *IsService* werden nur von wenigen Managern (Bereichs- oder Hauptabteilungsleiter, Abteilungsleiter) gepflegt, die ein Interesse an korrekten Daten haben. Einem Projektleiter sollte in diesem System möglichst keine Manager-Rolle zugestanden werden. Insgesamt hat es in 6½ Jahren ca. 900 Vorgänge vom Typ *IsService* und ca. 115.000 Kontierungen gegeben. Der für die Datenkorrektheit wichtigste Objekttyp ist also *IsServiceKontierung*.

3.3 Integritätsbedingungen

Die Integritätsbedingungen werden in Tabelle 2 durch Wertebereiche und Ergänzungen halbformal dargestellt, wobei nur erläuterungsbedürftige Attribute des Objekttyps *IsService* und die Attribute des Typs *IsServiceKontierung* betrachtet werden müssen.

Die Datendefinition hat jetzt eine Genauigkeit erreicht, die eine Beurteilung des Aspektes Korrektheit zuläßt. Es sollen überwiegend systematische Fehler betrachtet werden. Das einzige Attribut, an dessen Korrektheit ein **Manager** nicht selbst interessiert sein könnte, ist die *Kostenstelle*. Wenn eine innerbetriebliche Kostenverrechnung betrieben wird, sind die **Anwender** ein Regulativ für korrekte Belastungen. Manipulationsmöglichkeiten gibt es kaum. Das Korrektiv *Anwender* würde übrigens bei einer wissenschaftlichen Primärerhebung fehlen.

Attribut	*Wertebereich*	*Ergänzungen*
Vom **Manager** *gepflegt* (= ändern erlaubt):		
IsService#	**Integer**	maschinelle Vergabe
VorgangsArt	{Projekt, Wartung, Betreuung, Fehler, Kauf}	K ist ein Ereignis ohne Kontierungen; F *kann* Kontierungen haben
AnwSystemCode#	{Logistik, Produktion, Vertrieb, Administration, Information}	ist Bestandteil von AnwTeilSystem-Code#
EntwicklSchr	**{'0'..'6',** und andere**}**	ausführlich siehe [Spit96, Abschn. 3.4]
strategRelevanz?	**Bool**	nur für Projekte und Kaufsoftware
Kostenstelle#	**[Kostenstelle]**	Prüfung gegen Schlüsselverzeichnis
Vom **Entwickler** *erzeugt* (= ändern nicht möglich):		
IsService#	**[IsService#]**	Kontierung nur einer vorhandenen Nummer.
EntwicklerName#	**[UserID]**	maschinelle Eingabe
Datum#	**Datum**	default Tagesdatum, überschreibbar mit Datum < Tagesdatum
AufwandsZeit	**[Zeitdauer]**	[Stunde mit 1 Dezimale]
TätigkeitsBes	**Text**	Pflichteingabe ≥ 5 Zeichen
EntwicklSchr	**{'1'..'6'}**	*eingeschränkter* Wertebereich!
Kostenstelle#	**[Kostenstelle]**	Prüfung gegen Schlüsselverzeichnis; Eingabe nur bei VorgangsArt = B

Legende: schattiert: fälschungsgefährdete Attribute; **Fettdruck**: Basistyp; [**Fettdruck**]: problemspezifischer Basistyp; { }: Aufzählungstyp

Tab. 2: Integritätsbedingungen der wichtigsten Vorgangsdaten

Der **Entwickler** kontiert eine Kostenstelle nur bei Dauertätigkeiten wie *Betreuung*, bei denen es ex definitionem keine beauftragende Kostenstelle geben kann. Hier gilt entsprechendes wie beim Manager. Ebenso wird der Entwickler durch die Anwender „überwacht", so daß er kaum erfolgreich in größerem Umfang Aufwände den falschen Projekten oder Wartungsaufträgen (*IsService#*) zuordnen kann. Wenn man von täglich 6 bis 7 Stunden produktiver und damit weiterverrechenbarer Arbeitszeit ausgeht, gehört schon eine erhebliche Energie dazu, *AufwandsZeit* in größeren Dimensionen systematisch zu manipulieren. Da heutige Softwareentwicklung überwiegend mit Anwenderbeteiligung abläuft, ist es unwahrscheinlich, daß größere systematische Verfälschungen unentdeckt bleiben. Gegen *Zufallsfehler* beim Eingeben der *IsService#* (vertippen) hilft das bewährte Konzept einer *Prüfziffer*. Dies wurde im hier berichteten Fall versäumt und läßt sich nachträglich kaum in ein System einfügen.

Ein weiterer Korrektheitsaspekt betrifft das *Fehlen* eines Attributes. Es gibt keinen *TätigkeitsCode* [s. Boeh81, 691ff], etwa um Verfahrensschritte wie *Codierung* oder *Test* zu erfassen. Solche übergenauen Attribute erhöhen nicht etwa die Präzision der Daten, sie verringern sie, weil neben dem erhöhten Aufwand zu viele Auswahlfehler zu erwarten sind. Der erfaßte Text soll eine freizügige Protokollierung erlauben. Dies hat sich als nützlich für das Projektmanagement und für *Inhaltsanalysen* erwiesen. „Fehler“ einer *Tätigkeitsbeschreibung* sind also bewußt nicht definiert.

An der Verfälschung des Attributes *Entwicklungsschritt* dürften weder ein Manager noch ein Entwickler ein Interesse haben. Zum methodischen Unterschied zwischen den beiden Attributen siehe [Spit96, Abschn. 3.4].

3.4 Funktionen und Implementierungsfragen

Bereits im ursprünglichen Entwurf war das System bezüglich der Kontierungen als *Buchhaltung* vorgesehen, in der es *keine Änderung*soperation für die Datenerzeuger gibt. Eine Falschbuchung kann nur *storniert* werden, bleibt also als inverse Buchung in der Datenbank erhalten. Hierdurch ist jede Änderung nachvollziehbar (einer der Grundsätze ordnungsgemäßer Buchführung).

Ein **Manager** kann bestimmte Felder *ändern*, solange es noch keine abhängigen Kontierungen gibt. Er und auch die **Projektleiter** verfügen über eine Funktion *umbuchen*, die im Batchbetrieb abläuft und jede Veränderung in der Datenbank protokolliert. Außerdem wird automatisch eine Liste für das Management mit den Änderungen gedruckt. Die Umbuchungsfunktion wurde notwendig, um z. B. unnötig komplexe Projekt-/Teilprojektstrukturen wieder zu verschmelzen. Dabei wird angenommen, daß unübersichtliche Strukturen Auswahlfehler begünstigen.

Eine Funktion *löschen* gibt es für keine Benutzergruppe. Wird ein *IsService* angefordert, aber nie bearbeitet, wird er *storniert*. Wird ein Projekt oder ein Wartungsfall nicht beendet, wird er als *abgebrochen* gekennzeichnet.

Ein System für Selbstaufschreibungen von Softwareentwicklern darf auf keinen Fall ohne eine abgesichterte Datenbank installiert werden (also auf vielen PC-Plattformen *nicht*!). Es darf nicht möglich sein, mit selbstgeschriebenen Programmen die Datenbasis zu verändern. Dies wurde im vorliegenden Fall durch das System NATURAL-Security auf Basis von ADABAS sichergestellt. Entwickler und Manager können für datenerzeugende oder -ändernde Operationen nur die vorgesehenen Funktionen benutzen.

Die Entwurfsentscheidung für ein *Dialogsystem*, mit dem jeder Entwickler seinen Aufwand *selbst* erfaßt, ist kein Zugeständnis an den Zeitgeist, sondern Konzept. Eine möglichst zeitnahe Erfassung der Daten ohne Medienbruch geschieht im Interesse ihrer Korrektheit. Aus sozialwissenschaftlichen Untersuchungen ist

bekannt, daß Aufschreibungen ereignisnah stattfinden sollten, um verfälschende Einflüsse der Erinnerung auszuschließen [Roth93, 133; Laat93, 68]. Abbild 2 zeigt die Erfassungsmaske mit den Kontierungen eines in der Wartung tätigen Entwicklers für einen Arbeitstag. Es illustriert die auf Effizienz zielende Maskengestaltung und die offen gehaltene Tätigkeitsbeschreibung.

```
                        ZEITERFASSUNG - kurz
     MeierC                                                22.05.89

Sv   Sv   Kst  Entw  Zeit     Tätigkeit
Art  Nr        Schr _____________________________________________
 f   71   ___   _    1,5      fehlersuche doppelbuchung ad_____
 f   71   ___   _    0,5      sortierfehler wbs_______________
 w   70   ___   _    1        allgem. probleme________________
 w   72   812   _    1        div. wg kommissionierliste______
 w   264  ___   3    0,5      besprechung mob.auftr.erf_______
 w   315  ___   1    0,5      div. wg inventur________________
 w   331  ___   4    2,5      tourenplanung/kundenstammkarten__
 _   ___  ___   _    ____     _________________________________
 _   ___  ___   _    ____     _________________________________

____ ----------------------------------------------------------
 PF1= Hilfe   PF3= Zurück        PF12= Ende   DatFreig= WEITER
```

Legende: **Fettdruck** : Vom System angezeigte Daten. Nach Datenfreigabe werden Default-Daten (Kostenstelle und Entwicklungsschritt) angezeigt.

Abb. 2: Erfassungsmaske für einen Tagesbericht

Die Maske läßt übrigens erkennen, daß Kostenstellen über alle Mandanten und Werke disjunkt sein sollten. Andernfalls werden Erfassungsaufwand und Fehlerquellen vergrößert.

Da sich nicht alle korrektheitsfördernden Maßnahmen konstruktiv im Werkzeug abbilden lassen, muß auch das Einsatzkonzept betrachtet werden, das sich aus der evolutionären Entwicklung ergeben hat.

4 Einsatzkonzept

Eine Aufwandserfassung darf auf keinen Fall zur Kontrolle oder Beurteilung der Mitarbeiter eingesetzt werden, wie dies Selig in einigen von 36 untersuchten Unternehmen festgestellt hat [Seli86, 232]. Wenn Mißtrauen herrscht, steigt das Interesse der Mitarbeiter, systematisch falsch zu kontieren.

Daher wird ein offenes Konzept empfohlen, bei dem die Mitarbeiter das System als *Hilfe* bei der Projektabwicklung und Wartung empfinden. Im einzelnen:

- Jeder Entwickler kann alle Kontierungen online lesen und Auswertungen benutzen. Es wurden 8 online- und 13 batch-Auswertungen eingesetzt.
- Jeder Entwickler und Manager kann Aufwände nur für die eigene UserID erfassen, ist also für „seine“ Daten allein verantwortlich.
- Niemand wird nach den erfaßten Daten beurteilt, weder durch Vergleich der kontierten Arbeitszeiten mit einer Gleitzeiterfassung, noch durch Plan-/Istvergleiche von Projektschätzungen.
- Neu eingestellte Mitarbeiter kontieren in der Einarbeitungszeit *nicht*. Allein wegen der Auswahlfehler würde dies zu Beginn einer Tätigkeit in einem noch unbekannten Umfeld die Korrektheit der Datenbasis verringern.

An einer Stelle wird die Offenheit zugunsten korrekter Daten durch das Prinzip der *Vollständigkeit der Erfassung* auch eingeschränkt. Dies bedeutet, daß ein Mitarbeiter entweder alle Tätigkeiten erfaßt, insbesondere Projektarbeit *und* Wartung, oder gar keine. Letzteres gilt für internen Service, also Systemprogrammierung, Operating und Jobvorbereitung. Hierdurch wird die Möglichkeit verringert, Zeiten gar nicht zu kontieren.

5 Anwendungsbeispiele

Die folgenden beispielhaften Auswertungen zeigen Möglichkeiten der Verwendung der Aufwandsdaten vor allem für das Informationsmanagement:

- Tabelle 3 zeigt die Verteilung der IS-Service-Vorgänge auf Anwendungsgebiete über 6½ Jahre. In dieser Zeit wurden 271.077 Stunden in 837 Vorgängen kontiert, das sind bei 1.600 Std/MJ 169 Mitarbeiter-Jahre inclusive Betreuung.
- In Abbild 3 werden zeitliche Aufwandsverläufe für Berichterstattung und Projektmanagement dargestellt,
- Abbild 4 dient der Langzeitbeobachtung von Aufwänden, insbesondere dem Verhältnis zwischen Entwicklung und Wartung.

Anzahl IsService-Fälle in 6,5 Jahren: 1.1.1987 - 30.6.1993

SvArt	*Wartung*			*Projekt*			*Gesamt*			
AnwGebiet	S+A	"+"	i.A.	S+A	"+"	i.A.	S+A	"+"	i.A.	
Admin.	16	64	6	8	25	8	24	89	14	127
Information	5	48	9	6	23	5	11	71	14	96
Logistik	8	48	7	26	24	19	34	72	26	132
Produktion	4	26	6	8	19	31	12	45	37	94
Vertrieb	51	202	19	37	51	28	88	253	47	388
Gesamt	84	388	47	85	142	91	169	530	138	837

Legende:

S+A:	storniert oder abgebrochen
"+":	fertig
i.A.:	in Arbeit
	im Text diskutiert

Tab. 3: Anzahl IsService-Vorgänge je Anwendungsgebiet in 6½ Jahren

Tabelle 3 ist eine stark verdichtete Sicht auf die Daten für Zwecke der Berichterstattung und des Controlling. Danach sind im Mittel 20% der Vorgänge wieder *storniert* worden (kein Aufwand) oder *abgebrochen*. Auffällig ist der hohe Anteil im Gebiet Logistik, der zu hinterfragen wäre. Bemerkenswerter ist jedoch die große Anzahl *laufender* Aktivitäten in Produktion und Vertrieb mit genau gegensätzlichen Erklärungen. Von den 28 Projekten im **Vertrieb** sind 18 *angefordert*, 2 *in Einführung* und nur 8 wirklich *in Arbeit*. Außerdem sind seit Jahresbeginn 19 Wartungsaktivitäten und 9 Projekte *fertig* geworden. In der **Produktion** sind nur 3 Projekte *angefordert*, 3 *in Einführung* und 25 (!) *in Arbeit*, deren Erfolgsaussichten gering sind. Dies zeigt die „Erfolgsquote“ von nur 3 beendeten Wartungsvorgängen und 3 *fertig* gewordenen Projekten im letzten Halbjahr.

Das Beispiel zeigt die Verwendung ereignisbezogener Sichten für das Informationsmanagement und dessen starke Abhängigkeit von der Führungsorganisation: Die Unternehmensleitung hatte die Schließung eines der vier inländischen Produktionsstandorte zum Jahresende beschlossen.

Abbild 3 zeigt Aufwände im Zeitablauf:

Aufwand [Std] **je Anwendungsgebiet**

	Quartal						***Projekte***
AnwGebiet	IV/90	I/91	II/91	III/91	IV/91	I/92	= fehlt in der Grafik
Vertrieb	456	1557	1611	1682	1292	1745	Aufträge, Versand
Logistik	2070	1787	1533	2258	2548	2786	Bestände, Artikeldaten
Produktion	2511	2102	1916	1664	1362	839	PPS-System
Inform.	421	441	227	97	195	136	90/91: PC-Vernetzung
Admin.	107	82	195	150	76	127	überwiegend SAP
Wartung*	90	29	78	12	50	180	*= ad hoc - Wartung

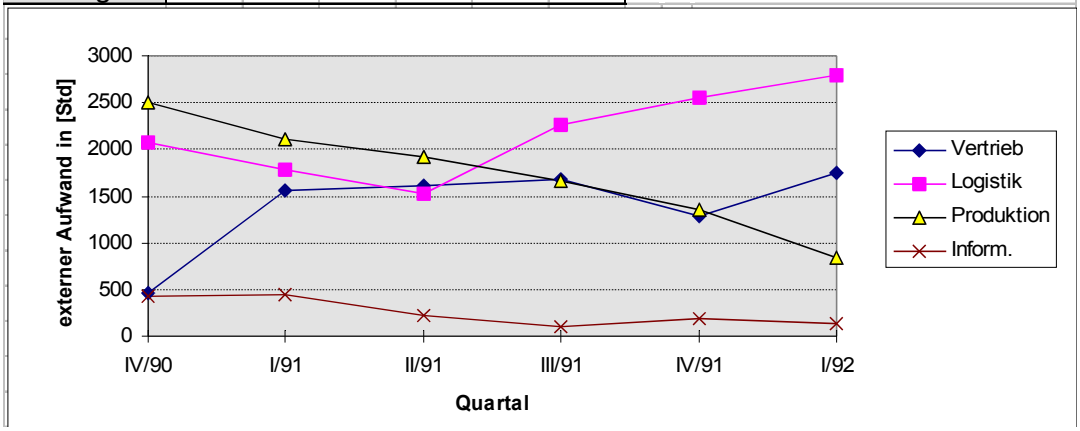


Abb. 3: Personaleinsatz Externe über 6 Quartale 1990 - 1992 je Anwendungsgebiet

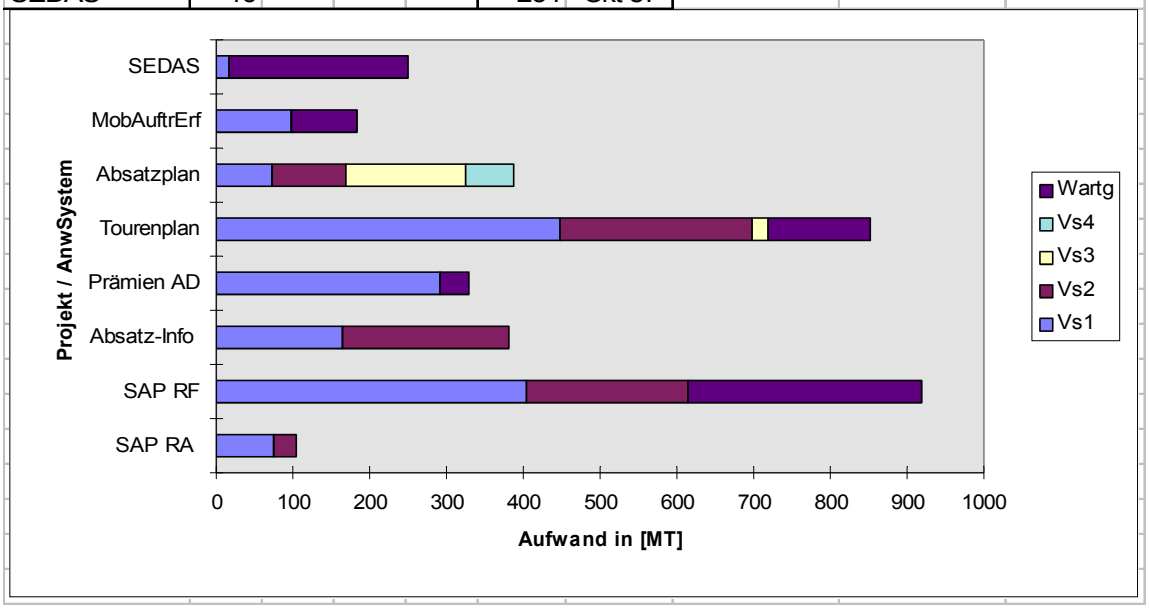
Während der Einsatz der Externen pro Quartal relativ konstant ist, geben die Aufwandsverläufe der Anwendungsgebiete interessante Signale: Die Vertriebsprojekte befinden sich unmittelbar vor der Einführung und zeigen einen relativ normalen Verlauf. Der Anstieg durch die Abschlußtests ist noch tolerierbar. Das

Produktionsprojekt *PPS-1* befindet sich ab I/92 in der Einführung. Der Verlauf ist fast ideal: Der externe Aufwand geht stetig zurück. Dagegen ist der steile Anstieg in der Logistik ein Alarmsignal. Er wird durch das „notleidende“ Projekt *Bestände* verursacht.

Das dritte Beispiel betrachtet einige Aufwände langfristig: Abbild 4 zeigt die dauerhafte Verwendbarkeit der erfaßten Daten. Die Systeme sind evolutionär in bis zu vier Versionen entwickelt oder stufenweise eingeführt (Fertigsoftware). Die letzte Datenspalte zeigt die Wartungsaufwände bis 30.6.1993, soweit sie separat erfaßt wurden. Den langfristigen Nutzen einer Aufwandserfassung kann man besonders deutlich an den Projekten *SEDAS* und *MobAuftrErf* sehen.

Nach dem Datenaustausch-Standard im Handel „SEDAS“ werden Rechnungsdaten vom Lieferanten zum Kunden per Leitung oder Datenträger übermittelt.

Aufwand [MT] **pro Version u. Wartung**							1 MitarbeiterTag = 8 Stunden
Projekt	Vs_1	Vs_2	Vs_3	Vs_4	Wartg	seit	
SAP RA	74	30			0		0 = Wartung nicht erfaßt
SAP RF	404	211			304	Okt 91	Wartung für RF *und* RA
Absatz-Info	165	217			0		
Prämien AD	291				39	Jul 89	
Tourenplan	447	250	21		134	Dez 89	
Absatzplan	72	97	157	61	0		Vs_4 zum Stichtag noch nicht beendet
MobAuftrErf	97				87	Aug 87	
SEDAS	16				234	Okt 87	



Abkürzungen: Datentransfer nach dem Standard SEDAS, Mobile Auftragserfassung, Prämiensystem Außendienst, Finanz- und Anlagenbuchhaltung des Systems R/2

Abb. 4: Entwicklungs- und Wartungsaufwand mittelgroßer Projekte 1.1.1987 bis 30.6.1993

Das geschäftspolitisch äußerst wichtige Projekt (Attribut *strategisch?* = true) wurde unter hohem Zeitdruck besonders „billig“ eingeführt. Der dauerhaft hohe Wartungsaufwand ist der „Preis“ für diese Vorgehensweise. Im Gegensatz dazu erscheint das Verhältnis zwischen Entwicklung und Wartung bei der mobilen Auftragserfassung durch 40 Außendienst-Mitarbeiter ausgewogen.

6 Befunde

Durch die empirisch gestützte Entwicklung des Erfassungssystems stellen die Kapitel drei und vier bereits Befunde dar. Es erscheint jedoch zulässig, auch aus der Datenbasis Aussagen als Hypothesen zu machen. Zunächst einige **qualitative Aussagen**:

- Wenn eine langfristig stabile Klassifikation der Anwendungssysteme existiert, dann lassen sich sowohl die *evolutionäre* (Weiter-)*Entwicklung* von Software als auch der Wartungsaufwand gut beobachten (vgl. Abb. 4).
- Es erscheint möglich, das Theorem einer *Lebenszyklus-Kurve* in der Nutzungsphase von Software (vgl. z.B. [Schw95, 49f]) aus den vorliegenden Daten zu falsifizieren. Eine gleichgerichtete Vermutung äußert Lehner [Lehn91, 68ff].
- *Phasenverläufe* in der Softwareentwicklung sind beobachtbar, müssen aber angesichts der Ergebnisse von Witte über die Sequenz von Entscheidungsverläufen [Witt68] noch differenziert untersucht werden. Dies ist mit dem vorliegenden Material möglich. Wenn es diese Phasen gibt, dann ist eine Phase *Einführung* als Abschluß der Entwicklung beobachtbar (vgl. auch [Spit93]). Sie darf weder fehlen [PaSi94, 65ff] noch der Wartung zugerechnet werden [Wall90, 5].
- Neben Entwicklung und Wartung muß eine Serviceart *Betreuung* treten, die nicht der Wartung zugerechnet werden darf. Die Betreuungsart *PC-Service* weist einen hohen Verwaltungsaufwand für die Anlagenbuchhaltung auf.

Einige **quantitative Aussagen** sind zunächst nur fallorientiert zu sehen:

- Etwa ein Viertel des Wartungsaufwandes ist *ungeplante Wartung*, die direkt auf Kostenstellen zu verrechnen ist (über 37 Monate in [Spit96]: 23,5%; über 78 Monate hier: 22,8%).
- Die Struktur und *Kostenentwicklung der Wartung* ist Ansatzpunkt für Einsparungspotentiale. Sie erlaubt Schlüsse über die Qualität von Systemen (vgl. Abb. 4: SEDAS) oder über die Wünsche der Anwender (vgl. Abb. 3: Vertrieb mit 52% der Wartungsanforderungen).
- Die *Aufwandsverteilung* zwischen Entwicklung und Wartung+Betreuung beträgt hier 66% : 34%. Der hohe Entwicklungsanteil ist Kennzeichen einer Investitionsentscheidung der Unternehmensleitung, die Synergien zwischen 11

operativen Gesellschaften des Konzerns voranzutreiben. Die Hypothese hieraus lautet: Aufwandsverteilungen zwischen Entwicklung und Wartung sind Ausdruck situativer Unternehmensentscheidungen. Sie kennen keine Gesetzmäßigkeiten (s. Abb. 4) und sind daher für wissenschaftlich-prognostische Zwecke oder als Zielgrößen unbrauchbar (s. auch [Lehn91, 26]).

7 Schlußfolgerungen

Es wurde gezeigt, *daß* man ein Meßinstrument für korrekte Selbstaufschreibungen im Betrieb entwerfen kann, und *wie* dies möglich ist. Es wurde weiterhin die praktische Verwendbarkeit unter betrieblichen Bedingungen demonstriert. Wenn die erfaßten Daten zeitnah benutzt werden, ist ihre Korrektheit sogar höher einzuschätzen als die von Daten, die nur für Forschungszwecke erhoben wurden. Zeitnahe Verwendungen sind das Projekt- und Qualitätsmanagement und die innerbetriebliche Kostenverrechnung. In beiden Fällen sind die Benutzer das Korrektiv, das systematische Verfälschungen der Daten bereits im Ansatz verhindert.

Ich danke Peter Wolf für kollegiale Hinweise und wertvolle Anregungen.

Literatur

[Atte75] Atteslander, P.: Methoden der empirischen Sozialforschung. 4. Aufl., DeGruyter, Berlin - New York et al. 1975.

[BaWe84] Basili, V.R., Weiss, D.M.: A Methodology for Collecting Valid Software Engineering Data. In: IEEE Transactions on Software Engineering 10(1984) 6, 728-738.

[Boeh81] Boehm, B.W.: Software Engineering Economics. Prentice Hall, Englewood Cliffs 1981.

[Grie87] Griese, J.; Obelode, G.; Schmitz, P.; Seibt D.: Ergebnisse des Arbeitskreises Wirtschaftlichkeit der Informationsverarbeitung. In: ZfbF 39(1987) 7, 515-551.

[Kerl73] Kerlinger, F. N.: Foundations of Behavioral Research. 2nd ed. Holt, Reinhart & Winston, London et al. 1973.

[Laat93] Laatz, W.: Empirische Methoden. Harry Deutsch, Thun - Frankfurt/M 1993.

[Lehn91] Lehner, F.: Softwarewartung. Hanser, München - Wien 1991.

[PaSi94] Pagel, B.-U.; Six, H.-W.: Software Engineering, Add.-Wesley, Bonn - Reading/Mass. et al. 1994.

[Roth93] Roth, E. (Hrsg.): Sozialwissenschaftliche Methoden. 3. Aufl., Oldenbourg, München - Wien 1993.

[Seli86] Selig, J.: EDV-Management. Springer, Berlin - Heidelberg et al. 1986.

[Schw95] Schwarze, J.: Systementwicklung. Neue Wirtschaftsbriefe, Herne - Berlin 1995.

[Spit89] Spitta, Th.: Software Engineering und Prototyping, Springer, Berlin - Heidelberg et al. 1989.

[Spit93] Spitta, Th.: Aufwandschätzung und Produktivität in Softwareprojekten. In: Softwaretechnik-Trends, 13(1993) 3 (Proceedings Softwaretechnik ‘93), 17-24.

[Spit96] Spitta, Th.: Die Aufwandserfassung von IS-Dienstleistungen. Erscheint in: Wirtschaftsinformatik 38 (1996) 5, 473-484.
[SzMB81] Szyperski, N.; Müller-Böling, D.: Zur technologischen Orientierung der empirischen Forschung. In: Witte, E.: Der praktische Nutzen empirischer Forschung, Mohr/Siebeck, Tübingen 1981, 159-188.
[Wall90] Wallmüller, E.: Software-Qualitätssicherung in der Praxis, Hanser, München - Wien 1990.
[Witt68] Witte, E.: Phasen-Theorem und Organisation komplexer Entscheidungsverläufe. In: ZfbF 20(1968) 10, 625-647.