Metriken für Eigenentwicklungen in SAP ERP Systemen

Andreas Mielke

VMS AG
Burgstraße 61
D-69121 Heidelberg
am@vms.net

Inst. f. Theoretische Physik
Univ. Heidelberg
Philosophenweg 19
D-69120 Heidelberg

Abstract: Für den Betrieb von großen SAP ERP-Systemen spielen Eigenentwicklungen eine wichtige Rolle. Sie tragen zwischen 15% und 35% der kompletten Betriebskosten der SAP-Systeme. In Upgrade-Projekten entfallen zwischen 60% und 80% der Kosten auf das Testen von Eigenentwicklungen. Von daher spielen Mengengerüste für Eigenentwicklungen eine wichtige Rolle in der Beurteilung von SAP-Kosten. Entsprechende Metriken werden hier dargestellt, basierend auf Auswertungen von Nutzungsdaten aus SAP-Systemen von jeweils mindestens einem Jahr. Wir liefern Metriken zur Nutzung von Eigenentwicklungen, zu den Kosten von Eigenentwicklungen und zu ihrer Performance.

1 Einleitung

ERP-Systeme gehören in Großunternehmen in vielen Fällen zu den zentralen IT-Systemen. Kosten von ERP-Systemen oder ERP-Projekten spielen damit für die IT in Großunternehmen eine wesentliche Rolle. Obwohl ERP-Systeme seit mehr als 20 Jahren eingesetzt werden, werden ihre Kosten erst seit etwa 10 Jahren systematisch untersucht (siehe [Ste01, Fra01] und die Referenzen darin). Dabei liegt der Schwerpunkt auf den Kosten für ERP Einführungsprojekte. In [Ste01, Fra01] wird der Begriff ERP-Projekt synonym mit ERP Einführungsprojekt verwendet. Es wurden eine Reihe von Metriken für die Kosten von ERP-Projekten entwickelt, einen Überblick liefert [DW08]. Andererseits stellen die Einführungskosten im Rahmen einer Gesamtkostenbetrachtung (total cost of ownership, TCO) nur einen geringen Teil dar [SAP05, Uel04]. Nach [ZSB04] tragen die Betriebskosten im Rahmen einer TCO-Betrachtung auf einer Basis von fünf Jahren etwa 80% der Gesamtkosten einer IT-Anwendung, nach [SK07] liegt der Anteil zwischen 60% und 80%.

Es gibt in der Praxis kaum ERP-Systeme, die vollständig ohne Eigenentwicklungen auskommen. Dafür gibt es verschiedene Gründe. Der wesentliche Grund ist sicher, dass bestimmte, für die Abbildung der Geschäftsprozesse des Unternehmens wichtige Funktionalitäten im Standard nicht vorhanden sind und daher nur durch Eigenentwicklungen realisiert werden können. Ein zweiter Grund sind Anforderungen an das Reporting, die im Standard nicht vorhanden sind. Ein dritter Grund sind schließlich Anforderungen einzelner Anwender, die

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

in Eigenentwicklungen realisiert werden.

Eigenentwicklungen tragen einen großen Anteil an den Gesamtkosten eines ERP-Systems. Nach Beatty und Williams [BW06] werden 60% bis 80% der Kosten bei Upgradeprojekten durch Testaufwände von Eigenentwicklungen verursacht. Typische Beispiele sind Releasewechsel, Unicode-Umstellungen, Migrationen. Hier müssen alle im System vorhandenen Eigenentwicklungen umgestellt und anschließend getestet werden, was zu einem deutlich erhöhten Aufwand führt. Dies bestätigen frühe Fallstudien von Ng [Ng01] und Light [Lig01]. Gründe für diesen hohen Kostenanteil werden in den genannten Arbeiten nicht präzise untersucht. Ein wichtiger Aspekt sind aber eine häufig im Vergleich zum Standard niedrigere Code-Qualität, die dazu führt, dass häufiger Fehler oder Abbrüche auftreten, dass die Softwarepflege aufwendiger ist oder dass häufiger Anpassungen vorgenommen werden.

In der Praxis typische Fragestellungen zu Beginn von Upgradeprojekten betreffen die Abschätzung des Aufwandes. Häufig ist nicht bekannt, wieviele Eigenentwicklungen in einem System tatsächlich vorhanden sind, so dass auch der Umfang von Anpassungen und der Testaufwand nicht präzise abgeschätzt werden können. Hat man den Aufwand abgeschätzt, ist die zweite Frage, wie er sich reduzieren lässt. Ein erster Ansatz dazu ist festzustellen, welche im System vorhandenen Eigenentwicklungen überhaupt noch benötigt werden. In einem zweiten Schritt kann dann die Frage gestellt werden, ob ein Teil der nötigen Anpassungen automatisiert vorgenommen werden kann. Häufig werden diese Fragen nur bei größeren Projekten gestellt, letztlich müsste aber schon bei jedem Patch, den der Hersteller des ERP-Systems zur Verfügung stellt, geklärt werden, ob dieser Patch Einfluss auf die Eigenentwicklungen hat. In diesem Zusammenhang sind z. B. auch Eigenentwicklungen wichtig, die durch die Modifikation einer Funktionalität aus dem Standard entstanden sind. Betrifft dieser Patch die entsprechende Standardfunktionalität, so wird die nötige Änderung durch den Patch nur dort aber nicht in der Eigenentwicklung durchgeführt.

Die im Vergleich zum Standard niedrigere Code-Qualität von Eigenentwicklungen führt oft auch zu einer schlechteren Performance von Eigenentwicklungen [Boh06]. Diese wirkt sich im SAP-TCO-Modell [SAP05, Uel04] bei den Endnutzerkosten aus. Um Aufwandsabschätzungen für ein Projekt zur Performanceoptimierung zu bekommen, ist es wichtig zu wissen, wieviele Eigenentwicklungen betroffen sind. Aussagen dazu sind oft nicht vorhanden und auch nicht leicht zu bekommen.

Ziel dieses Artikels ist es, Mengengerüste zu den oben genannten Fragestellungen zur Verfügung zu stellen. Mit diesen Mengengerüsten lassen sich in typischen SAP ERP-Systemen zumindest die Aufwände präzise abschätzen. Die gezeigten Mengengerüste stammen aus der VMS Benchmarkbase, der Datenbank der VMS AG, in der sich Daten von mehr als 2.400 (Stand März 2011) vermessenen SAP-Systemen befinden.

Die Arbeit ist wie folgt gegliedert: Im folgenden Abschnitt gehen wir zunächst auf den Stand der Forschung ein. Der dritte Abschnitt beschreibt die Methode, dazu gehört insbesondere eine genaue Beschreibung der verwendeten Stichprobe und der Datenerfassung. Im vierten Abschnitt werden die Resultate dargestellt, der fünfte Abschnitt enthält Zusammenfassung und Ausblick.

ISBN 978-3-88579-286-4

2 Stand der Forschung

Die in der Einleitung erwähnten Arbeiten betrachten fast ausschließlich Einführungsprojekte von ERP-Systemen. In der vorliegenden Arbeit geht es dagegen um den Betrieb dieser Systeme und um Upgradeprojekte. Bei der Durchsicht der Literatur stellt man fest, dass es nur wenige empirisch abgesicherte, quantitative Analysen zu Menge und Performance von Eigenentwicklungen in laufenden ERP-Systemen gibt. Generelle Kennzahlensysteme des IT-Controlling [Küt03] beinhalten keine Kennzahlen, die Eigenentwicklungen in SAP-Systemen betreffen. Ansätze für Kennzahlen, die Eigenentwicklungen in SAP-Systemen betreffen, finden sich in [Boh06, SK07, SK09].

Bohr [Boh06] führt eine Vielzahl von Kennzahlen ein und wertet diese nach Branchen aus. Die Arbeit verwendet keine anerkannten statistischen Methoden für diese Analyse, insbesondere fehlt eine Analyse der Frage, für welche Kennzahlen die Branche überhaupt ein signifikanter Faktor ist. Die untersuchte Stichprobe enthält nur für drei der betrachteten Branchen mehr als fünf Systeme. Die Datenbasis ist also nicht gesichert.

Sekatzek und Krcmar [SK07] schlagen ein Kennzahlensystem zur Optimierung von SAP-Systemen vor, mit dem Kosten, Lizenznutzung, Hardwarenutzung und Standardisierungsgrad charakterisiert werden. Die Autoren machen auf die Bedeutung von Eigenentwicklungen und Modifikationen in ERP-Systemen aufmerksam, sie stellen einen wichtigen Kostentreiber dar. In [SK09] untersuchen diese Autoren die Standardnähe von ERP-Systemen genauer. Sie definieren Kennzahlen zur Beurteilung der Standardnähe, beschreiben Verfahren zur Bestimmung dieser Kennzahlen und eine Implementierung dieses Verfahrens und wenden diese Implementierung im Rahmen einer Fallstudie auf Systeme eines Automobilkonzerns an. Doch auch hier bleiben Fragen offen, insbesondere Aufwandsabschätzungen von Upgradeprojekten oder Performance betreffend (Kapitel 5 in [SK09]). Insbesondere ist unklar, ob und wie sich die Ergebnisse auf andere Fälle übertragen lassen. Einem Teil dieser Fragen wird in der vorliegenden Arbeit nachgegangen.

3 Methode und verwendete Stichprobe

Für diese Arbeit wurden 118 verschiedene SAP ERP-Systeme vermessen. Für diese Vermessung wurde der VMS data collector der VMS AG verwendet. Es handelt sich dabei um ein in ABAP geschriebenes Programm, das in ein SAP-System transportiert und eingeplant wird und dann im Batchbetrieb täglich Meßdaten liefert. Es kommen die gleichen oder zumindest ähnliche Methoden und Funktionsbausteine zum Einsatz wie in [SK09]. Folgende Daten werden durch diese Vermessung erhoben:

Nutzungsdaten: Für jede Transaktion und jeden Report wird pro Stunde die Zahl der Steps getrennt nach Tasktypen (Dialog, Batch, etc.) erfasst, dazu Ressourcenverbräuche wie CPU-Last, Datenbankzugriffe und die Laufzeit. Diese Daten sind in einem SAP-System im CCMS[1] vorhanden und z. B. auch über den Systemlastmonitor des

[1] Das SAP Computing Center Management System, enthält Komponenten zum Monitoring des Systems.

Systems (Transaktion ST03N) zugänglich. Die Daten können also prinzipiell auch ohne Vermessung erhoben werden.

Statische Daten: Dazu gehören insbesondere Information aus der Tcode-Tabelle des SAP-Systems, wo verzeichnet ist, welche Transaktion welches Programm aufruft. Weiter wurden Master-Include Beziehungen in den SAP-Systemen ausgelesen um festzustellen, welches Programm andere Programme oder Includes verwendet. Diese Daten sind in Tabellen des SAP-Systems gespeichert und können ebenfalls prinzipiell auch ohne Vermessung erhoben werden. Alternativ können diese Informationen auch über eine automatisierte Code-Analyse bestimmt werden.

Zusätzlich zu diesen Meßdaten wurde für alle Systeme eine vollständige Erfassung der Betriebskosten mit Hilfe der VMS-TCO-Workbench[2] durchgeführt. Die Betriebskosten enthalten laufende Kosten und Abschreibungen aus dem Rechenzentrumsbetrieb, für die Datenbank, die SAP-Basis, den vollständigen Applikationsbetrieb inkl. Support, Incident Management, Change Management, Benutzerverwaltung, Administration, etc. Enthalten sind alle Kosten für Anschaffung und Wartung von Hard- und Software, die Personalkosten sowie Kosten für Raum, Strom, Klima, etc. Die Vollständigkeit der Kostenerfassung wird mit Hilfe des SAP TCO-Modells geprüft [SAP05, Uel04]. Es werden alle Kosten dieses Modells erfasst außer Einführungskosten, Projektkosten und Endnutzerkosten.

In der Stichprobe sind nur ERP-Systeme berücksichtigt, die mindestens zwei Module intensiv verwenden. Ausgeschlossen sind reine HR-Systeme oder auch BW-Systeme, CRM-Systeme, APO-Systeme, etc. Ausgeschlossen wurden auch Systeme, die durch spezielle Branchenlösungen dominiert werden, z. B. IS-U. Für alle Systeme liegen Meßdaten von mindestens 12 Monaten vor.

Die Analyse der Daten erfolgt mit Hilfe von Verteilungen, Korrelationen und Varianzanalysen der betrachteten Größen und mit parametrischen Regressionen [FKS07]. Für die Regressionen werden lineare Modelle verwendet. Für die konkrete Durchführung der Analyse kommt das Statistikpaket R (siehe z.B. [Dal08] und die darin zitierte Literatur) zum Einsatz.

4 Resultate

Wenn ein Benutzer in einem SAP System arbeitet, verwendet er Transaktionen und Reports. Wir betrachten deshalb zunächst Transaktionen und Reports. Eine wichtige Kenngröße, die schon bei einer kurzen Vermessung eines Systems hinreichend präzise bestimmt werden kann, ist die Zahl unterschiedlicher, pro Tag verwendeter Eigenentwicklungen (nur Transaktionen und Reports). Die Schwankungen dieser Messgröße in einem System sind von Tag zu Tag so gering, dass in fast allen Fällen ein Mittelwert von einer Woche ausreichend ist. Da die Nutzungsdaten in der Regel für mindestens eine Woche im CCMS des SAP-Systems gespeichert werden, kann diese Größe mit Hilfe des Systemlastmonitors

[2] Werkzeug zur Modellierung von Kostenstrukturen und zur Kostenerfassung von SAP Betriebskosten der VMS AG.

(Transaktion ST03N) sofort ermittelt werden.

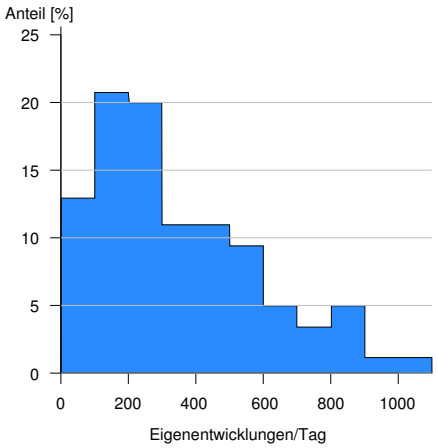

Abbildung 1: Verteilung der Zahl der pro Tag verwendeten unterschiedlichen Eigenentwicklungen in der Stichprobe der 118 vermessenen SAP-Systeme. Eigenschaften der Verteilung: Erstes Quartil: 170, Median: 283, Drittes Quartil: 453, Mittelwert: 308.

Abbildung 1 zeigt eine Verteilung dieser Größe für die verwendete Stichprobe. Da diese Größe für SAP-Systeme ohne großen Aufwand bestimmt werden kann, verwenden wir sie im folgenden als unabhängige Variable. Ob eine Variable in statistischen Modellen als unabhängige Variable geeignet ist, kann erst a posteriori gezeigt werden. In allen weiter unten betrachteten Modellen erweist sie sich als geeignet.

4.1 Metriken zur Verwendung von Eigenentwicklungen

Eine wichtige Größe zur Beurteilung von Eigenentwicklungen in einem SAP-System ist die Zahl der pro Jahr verwendeten Eigenentwicklungen (Transaktionen und Reports). Dabei ist zu beachten, dass einige Programme nur an wenigen Tagen im Jahr verwendet werden, z.B. im Rahmen des Geschäftsjahreswechsels. Ein erstes Ziel ist es, für diese Größe eine Abschätzung zu gewinnen.

Das Ergebnis ist in Abbildung 2 dargestellt. Die Abschätzung (Fit und Streuung in der Abbildung) lautet

$$(\#\,\text{Eigenentwicklungen/Jahr}) = (5,23 \pm 0,81)(\#\,\text{Eigenentwicklungen/Tag})^{0.93} \quad (1)$$

Der F-Test liefert $F_{(1)} = 351 \gg F_{0.05,1,117} = 3.92$. Das Verhalten ist leicht sublinear. Der lineare Fit liefert

$$(\#\,\text{Eigenentwickl./Jahr}) = (3,0 \pm 0,48)(\#\,\text{Eigenentwickl./Tag}) + 162 \pm 112 \quad (2)$$

mit $F_{(2)} = 86.5$. Beide Modelle schätzen die Daten ähnlich gut ab. Die große Streuung

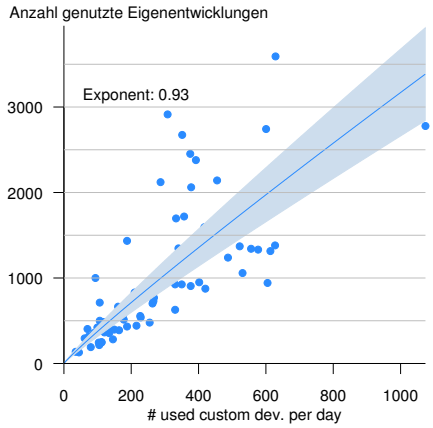

Abbildung 2: Aufgetragen ist die Zahl der pro Jahr verwendeten Eigenentwicklungen (nur Transaktionen und Reports) in Abhängigkeit von der Zahl der pro Tag verwendeten Eigenentwicklungen (nur Transaktionen und Reports). Die blaue Linie gibt einen Fit an, die hellblaue Fläche die Streuung. 60% der Systeme liegen in diesem Bereich.

im konstanten Term in (2) und der etwas niedrigere F-Wert führen dazu, das erste Modell vorzuziehen.

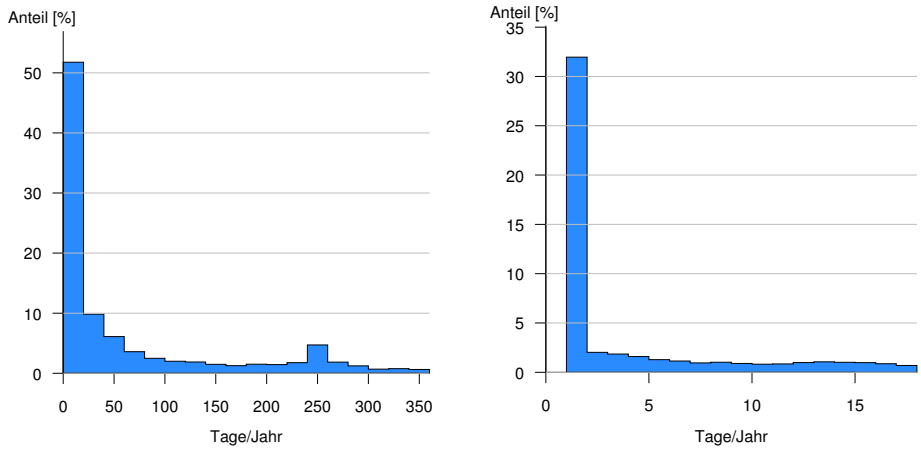

Abbildung 3: Das Histogramm links zeigt, wie groß der Anteil von selten oder häufiger genutzten Eigenentwicklungen ist. Aufgetragen ist der Anteil von Eigenentwicklungen in einem typischen System gegen die Zahl von Tagen pro Jahr, an denen die Eigenentwicklungen verwendet werden. Rechts ist die Verteilung für selten (an weniger als 20 Tagen pro Jahr) genutzte Eigenentwicklungen gezeigt.

Abbildung 3 zeigt, wie viele Eigenentwicklungen wie oft verwendet werden. Zunächst fällt auf, dass ein sehr hoher Anteil (mehr als 50%) nur an 20 oder weniger Tagen verwendet wird. Auffällig ist aber auch das Maximum bei 250 Tagen pro Jahr. Das sind Eigenentwicklungen,

die häufig, typischerweise jeden Werktag verwendet werden. Schaut man sich diese genauer an, so gibt es hier drei Klassen:

1. Eigenentwicklungen, die für einen Geschäftsprozess wichtig sind, der sich nicht durch Standardtransaktionen aus SAP abbilden ließ. Wenn dieser Geschäftsprozess täglich verwendet wird, werden auch diese Eigenentwicklungen täglich verwendet.
2. Eigenentwicklungen, die wichtige Reports enthalten, die täglich aufgerufen werden.
3. Eigenentwicklungen, die nur eine leichte Modifikation einer Standardtransaktion sind und häufig verwendet werden.

Eigenentwicklungen, die selten verwendet werden, analysieren wir in Abbildung 3 im rechten Histogramm etwas genauer. Es zeigt sich, dass etwa ein Drittel aller Eigenentwicklungen nur einmal pro Jahr verwendet werden. Ein geringer Teil davon sind Eigenentwicklungen, die für den Jahresabschluss oder ähnliche, einmal pro Jahr stattfindende Aktivitäten benötigt werden. Die meisten sind dagegen Programme, die für einen speziellen Zweck erstellt werden, für diesen Zweck an einem Tag verwendet werden und danach niemals wieder verwendet werden.

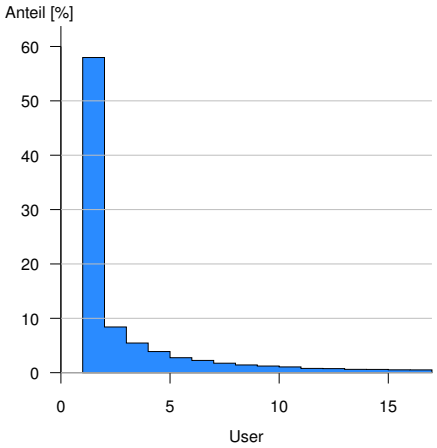

Abbildung 4: Das Histogramm zeigt, wie hoch der Anteil von Eigenentwicklungen in einem System typischerweise ist, der von einer bestimmten Anzahl von Benutzern verwendet wird.

Ein ähnliches Bild zeigt auch Abbildung 4. Sehr viele Eigenentwicklungen werden nur von sehr wenigen Nutzern verwendet. Fast 60% aller Eigenentwicklungen in einem SAP-System werden nur von einem Benutzer verwendet. Sie fallen in drei Klassen:

1. Eigenentwicklungen, die für einen speziellen Zweck für einen Benutzer erstellt werden und von diesem nur für kurze Zeit verwendet werden.
2. Eigenentwicklungen, die für einen speziellen Zweck für einen Benutzer erstellt werden und von diesem regelmäßig verwendet werden.

3. Eigenentwicklungen, die von einem technischen Benutzer regelmäßig im Batch ausgeführt werden, z. B. auch für eine Schnittstelle.

Die Analyse zeigt, dass ein großer Teil der Eigenentwicklungen selten oder nur zu einem bestimmten einmaligen Zweck verwendet wird. Es handelt sich gewissermaßen um „Sonderlocken“, die in vielen Fällen abgeschnitten werden können.

In diesem Zusammenhang ist noch interessant anzusehen, in welchen Tasktypen (Dialog, Batch, etc.) Eigenentwicklungen verwendet werden. Das Resultat zeigt Tabelle 1. Bei der Zahl der Eigenentwicklungen und bei den Steps dominiert Dialog, bei der Laufzeit dominiert Batch.

Tasktyp	# Eigenentwicklungen	# Steps	Laufzeit
Dialog	69%	84%	21%
Batch	20%	5%	77%
RFC	7%	4%	1%
Verbucher	4%	7%	1%

Tabelle 1: Verteilung der genutzten Eigenentwicklungen (Transaktionen und Reports) auf die verschiedenen Tasktypen. Dialog enthält HTTP und HTTPS. Verbucher fasst alle Verbuchertypen von SAP zusammen. Alle anderen Tasktypen haben Anteile unter 0,2%.

In den bisher vorgestellten Ergebnissen wurden Transaktionen und Reports betrachtet. In vielen Fällen ist es wichtig zu wissen, wieviele Programme verwendet werden, z.B. immer dann, wenn Codeanalysen nötig sind. Es gibt keine eineindeutige Abbildung von der Menge der Transaktionen und Reports zu der Menge der Programme. Da eine Transaktion in einem SAP-System nur ein Einstiegspunkt in ein Programm ist, können zu einem Programm mehrere Transaktionen gehören. Da andererseits ein Programm andere Programme aufrufen kann, kann der Aufruf einer Transaktionen zu dem Ablaufen von mehreren Programmen führen. Die Metrik, die die Anzahl der verwendeten, eigenen Programme liefert, ist einfach durch einen proportionalen Zusammenhang gegeben:

$$(\#\text{ eigene Programme/Jahr}) = (0,91 \pm 0,05)(\#\text{ Eigenentwicklungen/Jahr}) \tag{3}$$

In unseren Daten gibt es keinen signifikanten Zusammenhang zwischen dem Faktor Branche und der Zahl der pro Tag verwendeten Eigenentwicklungen. Tabelle 2 zeigt zwar Schwankungen von Branche zu Branche, die Schwankungen innerhalb der Branchen sind aber größer. Der F-Test im Rahmen einer Varianzanalyse liefert $F = 1.68 < F_{0.05,8,110} = 2.02$, die Nullhypothese kann damit nicht verworfen werden. Auch auf alle anderen in dieser Arbeit betrachteten Größen hat die Branche keinen signifikanten Einfluss. In [Boh06] wird ein solcher Zusammenhang suggeriert, aber nicht empirisch belegt.

Damit sind aber erst die genutzten Eigenentwicklungen analysiert. Im folgenden Abschnitt diskutieren wir vorhandene Eigenentwicklungen.

Branche	# Systeme	Mittelwert	Standardabw.
Automobil	7	417	69
Consumer Products	9	321	44
Energie	8	290	80
Finanz	11	202	53
Diskrete Fertigung	12	232	42
Pharmazie	14	302	49
Prozessfertigung	18	246	98
Handel	9	259	57
Services	20	346	30

Tabelle 2: Mittelwert und Standardabweichung der Zahl der pro Tag verwendeten Eigenentwicklungen für die verschiedenen Branchen.

4.2 Vorhandene Eigenentwicklungen

Die Stichprobe liefert gute Ergebnisse zu der Zahl der vorhandenen Programme. Das Ergebnis ist in Abbildung 5 dargestellt.

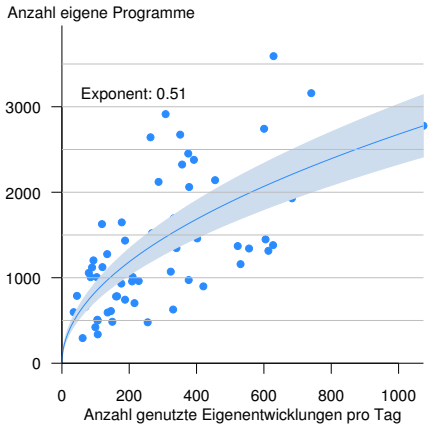

Abbildung 5: Aufgetragen ist die Zahl der vorhandenen Programme in Abhängigkeit von der Zahl der pro Tag verwendeten Eigenentwicklungen (nur Transaktionen und Reports). Die blaue Linie gibt einen Fit an, die hellblaue Fläche die Streuung. 60% der Systeme liegen in diesem Bereich.

Die Abschätzung (Fit und Streuung in der Abbildung) lautet

$$(\#\,\text{Programme}) = (81,1 \pm 10,6)(\#\,\text{Eigenentwicklungen/Tag})^{0.51} \tag{4}$$

Der F-Test liefert $F_{(4)} = 526 \gg F_{0.05,1,117} = 3.92$. Das Verhalten ist deutlich sublinear. Ein linearer Fit ergibt

$$(\#\,\text{Programme}) = (2,4 \pm 0,3)(\#\,\text{Eigenentwicklungen/Tag}) + 660 \pm 110 \tag{5}$$

mit $F_{(5)} = 55.2$. Auch hier liefern beide Modelle einen ähnlich guten Fit der Daten, auch hier ist das erste Modell wegen des höheren F-Werts zu bevorzugen. Auffällig ist, dass der Exponent in (4) deutlich niedriger ist als bei den verwendeten Eigenentwicklungen (Abbildung 2, Gleichung 1). Das bedeutet, dass der Anteil der nicht-verwendeten eigenentwickelten Programme in Systemen mit weniger Eigenentwicklungen höher ist. In einfachen Systemen liegt der Anteil bei 35% bis 45%, in komplexeren bei 30% bis 40%. Diese Zahlen liegen in der Größenordnung der Angaben in [Boh06].

Ein weiterer wichtiger Punkt ist, dass SAP-Systeme dynamisch sind. Es werden regelmäßig neue Programme in das System transportiert, die dann auch verwendet werden. Einige ältere Programme werden nicht mehr verwendet. Die Zahlen der täglich verwendeten Eigenentwicklungen und die Zahl der in einem Jahr genutzten Eigenenentwicklungen steigen dagegen kaum. Diese Zahlen verändern sich meist sprunghaft, wenn man einzelne Systeme ansieht, z. B. weil neue Funktionalität im System implementiert wird oder seltener weil eine Bereinigung (z. B. Prozessharmonisierung) vorgenommen wurde. Die Rate der in einem Monat neu in ein System transportierten Programme beträgt $(21-9+13)$/Monat, die Zahl der neu hinzukommenden Transaktionen und Reports ist höher. Damit kommen pro Jahr etwa 240 neue Programme in ein System. Diese Zahl liegt für einfache Systeme in der Schwankungsbreite in Abbildung 5, für komplexe deutlich darunter, so dass dieser Effekt für die Auswertung der Daten in Abbildung 5 statistisch nicht signifikant ist. Trotzdem ist dieser Effekt wichtig, er führt dazu, dass ältere Systeme von (4) eher nach oben, jüngere eher nach unten abweichen.

4.3 Funktionsgruppen, Modulpools, Includes

Für Funktionsgruppen, Modulpools und Includes streuen die Daten sehr, die Aussagen sind also erheblich weniger präzise.

Funktionsgruppen Die Zahl der Funktionsgruppen liegt typischerweise zwischen 10% und 40% der Zahl der Programme.

Modulpools Die Zahl der Modulpools liegt zwischen 2% und 4% der Zahl der Programme.

Includes Die Zahl der Includes liegt 2 bis 10 mal so hoch wie die Zahl der Programme.

Die Streuungen sind groß, im Einzelfall kann es deutliche Ausreißer nach oben, seltener auch Ausreißer nach unten geben. Wichtig ist aber, dass insbesondere die Includes mengenmäßig deutlich höher liegen als die Zahl der Programme.

4.4 Performance von Eigenentwicklungen im Dialog

Die Dialogperformance von Eigenentwicklungen ist in der Regel schlechter als die von Standardtransaktionen. Eine Analyse zeigt Abbildung 6. Für Standardtransaktionen liegt der Median in der betrachteten Stichprobe bei 530ms, der Mittelwert bei 810ms.

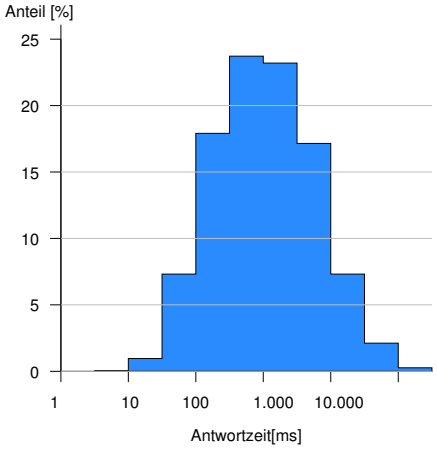

Abbildung 6: Verteilung der mittleren Antwortzeiten für häufig genutzte Eigenentwicklungen im Dialog. Berücksichtigt sind nur Eigenentwicklungen, die an mehr als 20 Tagen pro Jahr und mit mehr als 1.000 Dialogschritten pro Jahr verwendet werden. Eigenschaften der Verteilung: 1. Quartil: 300 ms, Median: 1.200 ms, 3. Quartil: 3.500ms, Mittelwert: 4.300 ms.

Die Verteilung ist eine log-normale Verteilung, was zu erwarten war [Mie06]. Mit einem Mittelwert von 4.300 ms und einem Median von 1.200 ms sind Eigenentwicklungen damit typischerweise einen Faktor 2 bis 4 langsamer als Standardtransaktionen. Da hier eine Verteilung von Mittelwerten betrachtet wird, bedeutet das auch, dass der hier gezeigte Median dem Mittelwert in einem System entspricht. Wir können also erwarten, dass in einem gegebenen System die mittlere Antwortzeit von Eigenentwicklungen bei 1.200ms liegen sollte, wobei längere Antwortzeiten häufiger vorkommen.

In Abbildung 7 wird die Performance der Nutzung gegenübergestellt. Typischerweise haben intensiv genutzte Eigenentwicklungen eine bessere Antwortzeit als solche, die weniger intensiv verwendet werden. Man sieht, dass lange Antwortzeiten häufig vorkommen. In Abbildung 7 werden im Vergleich zu Abbildung 6 nur Eigenentwicklungen mit mehr als einem Step pro Tag gezeigt, deshalb liegt hier der Median der Verteilung etwas unter 1s und damit niedriger als in Abbildung 6.

Die Analyse zeigt: Hauptproblem bei der Performanceoptimierung von Eigenentwicklungen sind meist nicht die sehr häufig verwendeten Eigenentwicklungen, sondern die große Zahl an Eigenentwicklungen, die zwar regelmäßig, pro Woche aber nur mit einem bis 500 Dialogschritten verwendet werden.

4.5 Kosten von Eigenentwicklungen

Dem TCO-Modell der SAP [Uel04] folgend lassen sich die Kosten für Eigenentwicklungen wie folgt gliedern:

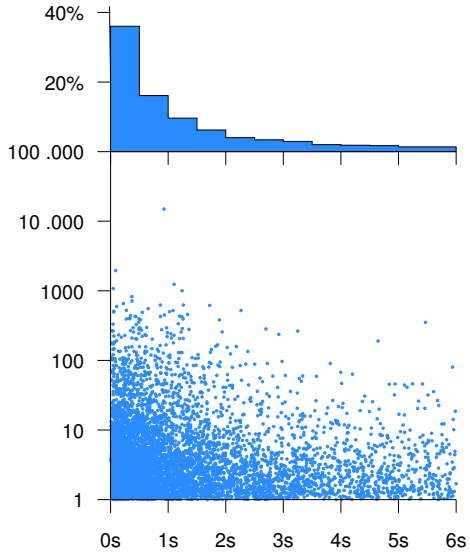

Abbildung 7: Verteilung der mittleren Antwortzeiten für häufig genutzte Eigenentwicklungen (mindestens ein Step pro Tag) in Abhängigkeit von der Antwortzeit. Im unteren Teil der Graphik ist für jede Eigenentwicklung ein Punkt eingezeichnet, aufgetragen ist die Zahl der Steps pro Tag gegen die Antwortzeit. Der obere Teil der Graphik zeigt die zugehörige Verteilung.

1. Herstellungkosten.
2. Reine Betriebskosten, gemeint ist hier der Anteil der Eigenentwicklungen an den Betriebskosten des Systems. Pflegeaufwände (bis zu zwei Manntage) und Fehlerbehebungen sind hier eingeschlossen.
3. Upgrade-Kosten für die Weiterentwicklung der Eigenentwicklungen (Aufwand über zwei Manntagen).
4. Projektkosten, die in Upgrade- oder Releasewechselprojekten für das ganze System zum Beispiel als Testaufwände entstehen. Auch bei Systemkonsolidierungen oder anderen großen Projekten treten diese Kosten auf.
5. Kosten bei End-User-Nutzung, insbesondere Kosten, die entstehen, weil durch schlechte Performance Arbeitszeit der User unnötig vergeudet wird.

ad 1. Die Herstellungskosten wurden nicht erfasst. Da es hier um Eigenentwicklungen geht, die bereits vorhanden sind, betrachten wir keine Herstellungskosten.

ad 2-4. Diese Kosten tragen nach [ZSB04, SK07] etwa 60% bis 80% gesamten Kosten. In typischen Situation liegen der Anteil der Eigenentwicklungen an den gesamten Betriebskosten bei 25%. In 80% aller Fälle liegt der Anteil zwischen 15% und 35%. Eine Kostenverteilung (gemittelt über die betrachteten Systeme) zeigt Abbildung 8.

Für die Projektkosten spielen vor allem die nicht genutzten Eigenentwicklungen eine Rolle. Aufgrund der in Abschnitt 4.1f dargestellten Metriken kann man in einem einfachen System

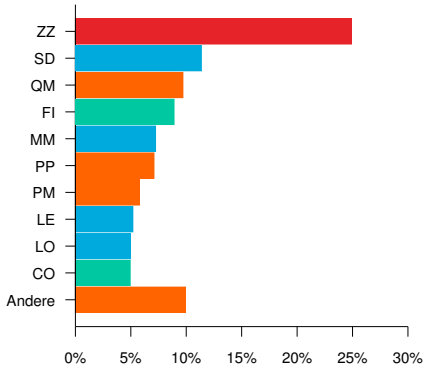

Abbildung 8: Verteilung der Betriebskosten auf Module in SAP ERP-Systemen, ZZ sind Eigenentwicklungen. Die Verteilung wurde durch Mittelung über die betrachteten 118 SAP-Systeme gewonnen.

mit etwa 130 pro Tag verwendeten Eigenentwicklungen mit 600 bis 800 verwendeten Eigenentwicklungen pro Jahr rechnen, für komplexere Systeme entsprechend mehr. In dem System ist mit 1.000 bis 1.300 vorhandenen Programmen zu rechnen. Daraus ergibt sich eine Zahl von etwa 500 vorhandenen aber nicht benötigten eigenentwickelten Programmen. Dazu kommen nicht verwendete Funktionen, Includes, etc. Der Testaufwand für diese Eigenentwicklungen kann eingespart werden. Die Einsparung durch Elimination der nicht mehr genutzten Eigenentwicklungen liegt also bei 30% bis 40%.

In dieser Abschätzung ist das Bereinigungspotential für wenig verwendete Eigenentwicklungen (wenige Steps pro Jahr, Verwendung von einem Benutzer, Reporting, das in ein BW transferiert werden kann, etc.) noch nicht enthalten.

Präzise Abschätzungen für den Testaufwand von Eigenentwicklungen liegen nicht vor. Die nach unserer Kenntnis höchste Schätzung [BW06] gibt einem Anteil von 60% bis 80% für den Testaufwand von Eigenentwicklungen am Gesamtbudget eines Upgradeprojekts an.

ad 5. In diesen Punkt geht vor allem die schlechtere Performance von Eigenentwicklungen ein, die zu längeren Wartezeiten beim Benutzer und damit zu Kosten führt. Präzise Aussagen sind zu diesem Punkt nicht möglich.

5 Zusammenfassung und Diskussion

Wir haben aus Daten von 118 über ein Jahr vermessenen SAP ERP-Systemen Metriken für Nutzung, Performance und Kosten von Eigenentwicklungen in SAP-Systemen empirisch untersucht. Die Motivation für eine solche Untersuchung ist klar: Modifikationen an SAP-Systemen sind in allen SAP-Systemen betrachteten Systemen vorhanden und in der Praxis kaum vermeidbar. Andererseits sind sie Kostentreiber sowohl bei den Betriebskosten als auch bei Projektkosten, insbesondere bei Upgradeprojekten. Das Hauptproblem ist, dass Unternehmen diese Kosten heute häufig nicht kennen. Die Metriken liefern damit erste

Hinweise darauf, wie sich Nutzung, Performance und Kosten von Eigenentwicklungen typischerweise verhalten. Die vorgestellten Metriken enthalten aber Streuungen. Sie ersetzen deshalb keine Untersuchung z. B. der Kosten im Einzelfall, sie liefern aber Hinweise darauf, wie die im Einzelfall bestimmten Kosten einzuordnen sind.

Die vorgestellten Metriken sind in vielen Bereichen hinreichend präzise, weil die Streuungen innerhalb der Stichprobe der 118 vermessenen Systeme nicht zu groß sind. Dazu gehören vor allem die Aussagen zu genutzten Eigenentwicklungen, zur Performance und zu Kosten. Nicht in einer Metrik darstellbar sind die Aussagen zu vorhandenen Funktionsgruppen, Modulpools und Includes. Hier sind die Streuungen innerhalb der Stichprobe so groß, dass sich keine generellen Aussagen treffen lassen.

Ein wesentliches Resultat ist, dass die Zahl der tägliche genutzten kundeneigenen Transaktionen und Reports als unabhängige Variable sehr gut geeignet ist. Wichtig ist auch, dass die typischen Abhängigkeiten zwischen Größen keine Proportionalität aufweisen, zum Teil nicht einmal linear sind, und dass es wichtig ist, die relative Zeitabhängigkeit der Größen, also ihre Dynamik zu kennen. Die fehlende Proportionalität führt dazu, dass Kennzahlen zur Beschreibung von Eigenentwicklungen zwar definiert werden können und für das Monitoring eines Systems sehr nützlich sein können, dass man aber bei Vergleichen zwischen Systemen sehr vorsichtig sein muss. Die Dynamik führt dazu, dass man bei der Beurteilung eines Systems ggf. auch sein Alter heranziehen muss. Neue Systeme liegen häufig, nicht immer näher am Standard als ältere.

Ein interessanter und für uns überraschender Punkt ist, dass die Branche des Unternehmens, von dem das System genutzt wird, kein signifikanter Faktor ist.

Es gibt eine Reihe weiterführender Fragestellungen, die sich an diese Untersuchung anschließen. Ein wichtiger Punkt ist die Dynamik. Kosten von SAP-Systemen ändern sich auf der Zeitskala eines Jahres. Ein signifikanter Einfluß des „Alterns“ von Systemen ist nur auf Zeitskalen von deutlich mehr als einem Jahr zu erkennen. Auf diesen Zeitskalen gibt es aber andere Tendenzen und Trends, die Einfluss auf die betrachteten Größen haben. Beispielsweise ist heute ein klarer Trend hin zu virtualisierten Umgebungen zu erkennen, der erheblichen Einfluss auf Kosten hat, gerade auch auf Projektkosten, weil z. B. Testsysteme in virtuellen Umgebungen viel schneller erstellt werden können. Auch für Performance, Hardwarekosten etc. spielt Virtualisierung eine wichtige Rolle. Ein anderer Trend ist die Entwicklung hin zu globalen Systemen. Globale Systeme stellen die IT z. B. vor die Herausforderung, dass es keine klaren Dialogzeiten mehr gibt, so dass Dialog- und Batchprozesse in Konkurrenz treten. Da Eigenentwicklungen, die im Batch verwendet werden, häufig einen besonders hohen Ressourcenbedarf haben, spielt ihre Optimierung in globalen Systemen eine zunehmend wichtigere Rolle.

Ein zweiter Punkt betrifft nicht-ERP-Systeme, also z. B. BW-Systeme, CRM-Systeme, Systeme mit Branchenlösungen. SAP-Landschaften sind heute zunehmend hoch integriert und für einen Anwender spielt es keine Rolle, wo eine Transaktion oder ein Report abläuft. Bei der Untersuchung anderer Arten von Systemen müssen aber ggf. andere Bestandteile betrachtet werden, gerade z. B. in BW-Systemen.

Ein dritter Punkt, der mit dem zweiten in Beziehung steht, sind Schnittstellen. Hier werden zunehmend auch Schnittstellen zu nicht-SAP-Systemen wichtig, weil Unternehmen mit

steigenden Anforderungen an die IT und mit zunehmender Integration innerhalb der IT zu tun haben. In vielen Situationen werden solche Schnittstellen durch Eigenentwicklungen bedient. Diese Eigenentwicklungen können ggf. durch Drittsysteme (z. B. SAP XI) ersetzt werden, dabei entstehen aber zusätzliche Kosten, sowohl bei der Implementierung als auch im Betrieb, die den Kosten für die existierenden Eigenentwicklungen entgegenzuhalten sind.

Literatur

[Boh06] D. Bohr. *KPI Scan. A Cost/Benefit Analysis of SAP Systems*. WestTrax/Experton Group, Hahnstätten, 2006.

[BW06] R. C. Beatty und C. D. Williams. ERP II: Best Practices for Successfully Implementing an ERP Upgrade. *Communications of the ACM*, 49(3):105–109, March 2006.

[Dal08] P. Dalgaard. *Introductory Statistics with R*. Springer Verlag Berlin, Heidelberg, New York, 2nd. Auflage, 2008.

[DW08] M. Daneva und R. Wieringa. Cost estimation for cross-organizational ERP projects: research perspectives. *Software Qual J*, 16:459–481, 2008.

[FKS07] L. Fahrmeir, T. Kneip und S.Lang. *Regression*. Springer Verlag Berlin, Heidelberg, New York, 2007.

[Fra01] C. Francalanci. Predicting the implementation effort of ERP projects: empirical evidence on SAP/R3. *Journal of Information Technology*, 16:33–48, 2001.

[Küt03] M. Kütz. *Kennzahlen in der IT*. dpunkt.verlag, Heidelberg, 2003.

[Lig01] B. Light. The maintenance implications of the customization of ERP software. *J. Softw. Maint. Evol.: Res. Pract.*, 13:415–429, 2001.

[Mie06] A. Mielke. Elements for Response Time Statistics in ERP Transaction Systems. *Performance Evaluation*, 64:635–653, 2006.

[Ng01] C.S.P. Ng. A decision framework for enterprise resource planning maintenance and upgrade: A client perspective. *J. Softw. Maint. Evol.: Res. Pract.*, 13:431–468, 2001.

[SAP05] SAP. A model to analyze total cost of ownership, 2005.

[SK07] E. P. Sekatzek und H. Krcmar. Ein Kennzahlensystem zur Optimierung von SAP Systemen. In A. Oberweis, Hrsg., *8. Internationale Tagung Wirtschaftsinformatik*, Jgg. 2, Seiten 271–288. Wirtschaftsuniversitätsverlag, Karlsruhe, 2007.

[SK09] E. P. Sekatzek und H. Krcmar. Messung der Standardnähe von betrieblicher Standardsoftware. *Wirtschaftsinformatik*, 3:273–283, 2009.

[Ste01] E. Stensrud. Alternative approaches to effort prediction of ERP projects. *Information and Software Technology*, 43:413–423, 2001.

[Uel04] T. Uellerich. *TCO-Modell für SAP-Systeme am Beispiel mySAP CRM mit SAP Enterprise Portal*. Gallileo Press, Bonn, 2004.

[ZSB04] R. Zarnekow, J. Scheeg und W. Brenner. Untersuchung der Lebenszykluskosten von IT-Anwendungen. *Wirtschaftsinformatik*, 46:181–187, 2004.