Industrialisierung der Softwarewartung & -weiterentwicklung

Harry M. Sneed

ANECON GmbH
Alserhof 4
1090 Wien
Harry.Sneed@anecon.com

Abstract: In diesem Beitrag wird geschildert, wie Software-Wartung & -Weiterentwicklung als Dienstleistung von externen Software-Firmen angeboten werden kann. Es wird argumentiert, dass die Vergabe der Systemerhaltung die einzige Möglichkeit ist die Kosten derselben in Griff zu bekommen. Der Outsourcing Partner braucht allerdings einen straffen, automatisierten Wartungsprozess, der hier beschrieben wird. Einen Nutzen durch diese Dienstleistung wird es erst geben, wenn sowohl der Anwender als auch der Outsourcing Partner genau definierte Rechte und Pflichten haben, die in einem Service-Level-Agreement festgehalten sind. Jener Vertrag ist für den Erfolg des Outsourcing ausschlaggebend. Welche Rechte und Pflichten das sind und wie sie einzuhalten sind, sowie der Prozess zur Durchführung der vereinbarten Dienstleistungen ist das Thema des vorliegenden Beitrags.

1 Hintergrund – Die Wartungsfalle

Die Last für die Erhaltung bestehender IT-Systeme ist ein Mühlstein um den Hals vieler Anwendungsbetriebe, vor allem für diejenigen, die nicht auf ERP-Produkte ausweichen können. Dazu gehören im ersten Range die Finanzdienstleister und die Behörden. Sie betreiben Legacy-Systeme, die größtenteils aus den 80er Jahren stammen. Diese Systeme werden nicht nur korrigiert und angepasst (corrective and adaptive maintenance), sondern auch optimiert und weiterentwickelt (perfective und enhancive maintenance). Um diese Aufgaben zu bewältigen sind Heersscharen alter Programmierer beschäftigt. Auch in Zeiten, in denen sie nur gering ausgelastet sind, müssen die Programmierer trotzdem bereit stehen, falls sie gebraucht werden. Durch ihr Wissen über die alten Systeme sind sie für den Anwender unentbehrlich. Stellvertretend für diese Situation ist das Softwareprofil eines typischen deutschen Finanzdienstleisters. Die IT Abteilung der Firma muss eine Altlast bestehend aus

- 3.652 IMS Datenbanken,
- 2.320 DB2 Tabellen,
- 31.453 IMS Masken,
- 10.933 Assembler Programmen,
- 10.450 PLI Programmen,
- 27.374 COBOL Programmen,
- mit mehr als 70 Millionen Code-Zeilen, bzw. 46 Millionen Anweisungen

betreuen und fortschreiben. Für die Pflege jener Altsysteme stehen gerade 360 Programmierer zur Verfügung. Daraus leitet sich ein Verhältnis zwischen Personal und Anweisungen von

- 1:127.777

ab.

Das heißt, ein Wartungsprogrammierer muss circa 128 Tausend Anweisungen betreuen - und das bei einer mittleren jährlichen Änderungsrate um 5 %. Aus der Literatur geht hervor, dass ein Wartungsprogrammierer nicht mehr als 80 Tausend Anweisungen betreuen sollte [Pi97]. In diesem Falle muss der einzelne Programmierer fast 6.388 Anweisungen im Jahr, bzw. 32 Anweisungen pro Tag ändern, löschen und hinzufügen. Unter diesen Umständen ist es äußerst schwierig, den Wartungsbetrieb überhaupt aufrecht zu erhalten- vor allem, wenn man berücksichtigt, dass die durchschnittliche Produktivität eines Entwicklers bei 25 Anweisungen pro Tag liegt [Bo99].

Codezeilen sind physikalische Einheiten, deren Länge von der Art des Editors bestimmt wird. Auf dem Mainframe in TSO sind sie 72 Zeichen; auf dem PC können sie beliebig lang sein. Codezeilen zu zählen macht wenig Sinn. Anweisungen sind hingegen logische Einheiten wie Sätze in der natürlichen Sprache. Sie werden mit einem Delimiter wie bspw. Punkt oder Semikomma beendet. Es können mehrere Anweisungen in einer Zeile vorkommen. Anweisungen können sich auch über mehrere Zeilen erstrecken, was eher vorkommt. Deshalb ist die Zahl der Anweisungen, bzw. „delivered source instructions“, in der Regel kleiner als die der LOCs [Hu00] (siehe Abbildung 1).

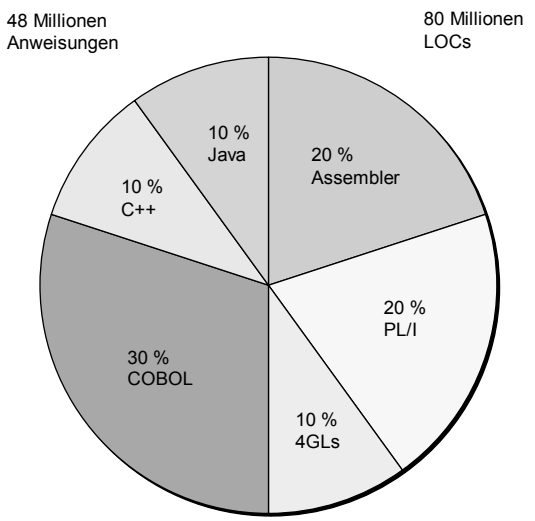

Abbildung 1: Softwareprofil eines Finanzdienstleisters

Die Kosten für die Aufrechterhaltung derartiger Wartungsbetriebe sind immens. Laut einer Umfrage der Gartner-Gruppe betragen sie im Durchschnitt 72 % des gesamten IT-Budgets [Ga05]. Der Wartungsbetrieb bindet wertvolle Ressourcen, vor allem wichtige Know-How-Träger, die für andere Aufgaben dringend gebraucht werden. Die alte Software zwingt die Anwender dazu, die alte Hardware zu behalten und teure Lizenzen zu bezahlen, da die Kosten einer Migration für sie oft unerschwinglich sind. Die wenigen Anwenderbetriebe, die es doch schaffen ihre Legacy-Systeme zu migrieren, müssen dafür einen hohen Preis bezahlen und tun sich schwer, diese Kosten gegenüber dem Geldgeber zu rechtfertigen, denn die Endanwender haben meistens nichts davon. Wenn alles gut geht, dürften sie die Umstellung kaum merken. Demzufolge sind sie wenig geneigt, diese Kosten zu tragen [BS95].

Folglich bleibt der Anwender in einem Dickicht von Abhängigkeiten stecken. Er ist abhängig von den Lieferanten der alten Software- und Hardware-Produkten, abhängig von den externen Spezialisten, die sich mit den alten Produkten auskennen und abhängig von den eigenen Programmierern, die einzig und allein in der Lage sind, die Altsoftware fortzuschreiben. Dem in der Legacy-Falle gefangene IT Anwender bleibt kaum noch Spielraum für Verbesserungsmaßnahmen, geschweige denn für Innovationen und neue Investitionen.

2 Befreiungsmöglichkeiten

Es bieten sich nur wenige Möglichkeiten an, sich aus dieser misslichen Lage zu befreien:

- Zum ersten, könnte der Anwender auf Standard-Software, sprich auf ERP Systeme umstellen.
- Zum zweiten, könnte der Anwender durch einen herkulischen Akt die Altsysteme sanieren und erneuern.
- Zum dritten, könnte der Anwender mit einem noch größeren Kraftakt die Altsysteme durch Neuentwicklungen ablösen.
- Zum vierten, könnte der Anwender die Altsysteme an einen externen Outsourcing-Partner übergeben.

Es sei zu betonen, dass eine Migration zwar den Anwender von der alten Hardware und Basissoftware befreien kann, jedoch nicht von der eigenen Anwendungssoftware, die in der neuen Umgebung unverändert bleibt. Durch eine Migration wird weder an der Größe, der Komplexität noch an der Qualität der Software gerüttelt [SAW08].

Der Hauptgrund für eine Migration ist einzig und allein der Wunsch nach Unabhängigkeit von einer bestimmten Hard- bzw. Software. Die zuständigen Manager wollen den Lieferanten wechseln, die Wartungskosten spielen dabei nur eine geringe Rolle. Die Abhängigkeit von gewissen Schlüsselpersonen scheint niemanden zu stören solange man glaubt, die Leute behalten zu können. Hinzu kommt, dass die meisten IT Leiter ohnehin nur kurzfristig planen und hoffen, die Altsysteme so lange über Wasser halten zu können. Ergo wird weder Sanierung noch Migration als Ausweg aus der Wartungskrise gesehen.

2.1 Standard-Software

Die erste Möglichkeit - die Umstellung auf Standard Software - setzt voraus, dass geeignete Standardlösungen angeboten werden. Für die Industrie ist dies inzwischen weitestgehend der Fall. Bis auf wenige Ausnahmen kann der industrielle Anwender seine IT-Systeme kaufen oder mieten. Er kann sogar zwischen Produktbeschaffung und Servicebeschaffung, sprich Software as a Service, wählen [SS03]. Er muss allerdings seine alten Daten migrieren und seine Geschäftsprozesse anpassen. Dies sei jedoch ein geringer Preis für die Befreiung aus der Wartungsfalle. Er begibt sich zwar in eine neue Abhängigkeit, aber diese ist kalkulierbar und tragbar. Sonst würde nicht der Großteil der Industrie diesen Weg einschlagen. Für Unternehmen mit besonderen Anwendungen gibt es diese Alternative leider nicht.

2.2 Sanierung und Erneuerung

Die zweite Möglichkeit erfordert eine größere einmalige Investition. Die Altsoftware wird im figurativen Sinne durch eine Jungfernmühle getrieben und kommt als erneuerte, wartbare und weiterentwicklungsfähige Software auf der anderen Seite wieder heraus. Dieser alte Traum des Software-Reengineering geht auf die 80er Jahre zurück und bleibt bisweilen weitgehend unerfüllt. Es gibt kaum Beispiele wirklichen Gelingens. Einzelne Bausteine bzw. Module lassen sich restrukturieren und bereinigen, einzelne Klassen lassen sich ebenfalls refaktorieren, aber die Gesamtarchitektur bleibt im Wesentlichen so, wie sie war. Auch die einzelnen Komponenten lassen sich nur bedingt verbessern. Man kann zwar die GOTO Anweisungen entfernen und den Kontrollfluss restrukturieren. Man kann auch die IO-Operationen auslagen und die Programme zerlegen in kleinere Module, aber die Qualität der Komponente steigt nur marginal. Die Software-Bausteine bleiben in deren ursprünglicher Technologie behaftet. Außerdem bleibt die Abhängigkeit zu den alten Programmierern weiter bestehen. Die Wartungslast wird dadurch nur um wenige Tausend leichter [Sn08].

Denn aller Erfahrung zu Folge steigt auf Grund einer Sanierung, sprich Restrukturierung oder Refaktorierung, die innere Qualität einer Software nie mehr als 30%. Das liegt daran, dass eine technische Lösung sich nur begrenzt verändern lässt, ohne die Lösung als solche zu zerstören. Das gleiche gilt auch für ein altes Haus, das saniert wird. Die Grenze zwischen Sanierung und Abbruch ist fließend. Da die Qualität eines Softwareproduktes nur eine von vier Wartungskostentreibern ist, neben

- der Wartungsumgebung,
- dem Wartungsprozess und
- dem Wartungspersonal,

kann sie allenfalls ein Viertel der Wartungskosten verursachen. Auch wenn eine Sanierung die Wartbarkeit um 30 % steigern könnte, würde im besten Falle circa 10 % der Wartungskosten gespart werden, da die vom Produkt verursachten Kosten höchstens einen Drittel der Gesamtkosten ausmachen [Sn91]. Die restlichen zweidrittel der Kosten werden von der Umgebung, dem Prozess und dem Personal verursacht.

Es ist daher fraglich, ob Anwender bereit sind die Kosten eines größeren Software-Reengineering Projektes zu tragen, wenn der Return on Invest so niedrig ist. Sie werden eher nach einfacheren, mehr Gewinn versprechenden Möglichkeiten suchen.

2.3 Neuentwicklung

Die Neuentwicklung bedeutet das alte System nachzudokumentieren und anhand der wieder gewonnen Dokumentation neu zu kodieren. Diese Lösung hat drei Nachteile. Zum einen ist es sehr schwierig herauszubekommen, was das alte System wirklich tut. Der Reverse Engineering Technologie ist es nie gelungen das eigentliche Fachkonzept aus dem Code wieder zu gewinnen. Zum zweiten ist eine neue Entwicklung sehr teuer, insbesondere was den Test anbetrifft. Eine völlig neue Testbasis muss aufgebaut werden. Das treibt die Kosten in die Höhe. Zum dritten ist eine neue Entwicklung immer mit hohen Risiken verbunden. Es hat sich nämlich gezeigt, dass die Ablösung eines bestehenden Systems um das zweifache risikoreicher ist als die Entwicklung eines völlig neuen Systems. Das kommt daher, dass die Anwender alles in dem alten System behalten wollen bei gleichzeitiger Ergänzung des Systems durch alle neue Anforderungen, die sich über die Jahre angesammelt haben. Man spricht hier vom „second system effect" - bzw. der Wunsch alles besser zu machen mit der Folge, dass das System überlastet wird [Th95].

Nur die wenigsten Anwender lassen sich auf dieses Abenteuer ein. Zu hoch scheinen ihnen die Risiken. Mit der Ablösung des alten Systems folgt die Ablösung der alten Programmierer, die jetzt umgeschult werden müssen. Durch die Umstellung auf eine neue Technologie werden die bisherigen Programmierer abgehängt. Sie können oft nur noch als Analytiker oder Tester weiterarbeiten. Nur wenige große Anwender sind in der Lage ihr altes Personal umzuschulen, bzw. auszuwechseln. Es bleibt den Anderen nur übrig, die alte Technologie beizubehalten.

2.4. Outsourcing der Software-Wartung

Als vierte Alternative bietet sich das Outsourcing der Softwarewartung an. Die Verantwortung für die Pflege und Weiterentwicklung wird an einen externen Dienstleister vergeben, der mit dem Anwender ein Service-Level-Agreement abschließt, wonach er Probleme in einer vereinbarten Zeit beseitigt und neue Versionen mit Änderungen und Erweiterungen in vereinbarten Intervallen ausliefert.

Die Idee für die 3rd Party Maintenance ist nicht neu. Sie hat Hillary Calow schon 1988 in England aufgegriffen. Calow hat eine Firma namens FI gegründet, die ausschließlich aus Müttern bestand, die vorher als Programmiererinnen gearbeitet hatten und nun zu Hause bleiben wollten, ohne ihren Beruf aufzugeben. FI schloss Verträge mit der lokalen Industrie ab, wonach die Frauen einmal wöchentlich zu Gesprächen mit den Kunden kamen. Die anderen Tage blieben sie zu Hause wo sie jederzeit erreichbar waren. Sie hatten einen lokalen Netzanschluss zum Hostrechner des Kunden. Sie bekamen ihre Wartungsaufträge über das Netz und hatten Zugriff auf die Source-Bibliotheken des Kunden. Außerdem hatten sie auch eine eigene Testumgebung.

Allen Berichten zu Folge hat das Geschäft mit der 3rd Party Maintenance gut funktioniert und die Kunden waren zufrieden. Die von zu Hause aus operierenden Frauen waren hoch motiviert und haben professionell gearbeitet [Ca88]. Der Untergang des FI Geschäfts setzte ein, als die Inder Ende der 90er Jahre in großem Stil die Software aus England nach Indien abgezogen haben. FI konnte nicht mit den niedrigen Preisen der großen, indischen Anbieter TCS, Wipro und InfoSys konkurrieren. Anwender, die bereit waren, ihre Software aus der Ferne warten zu lassen, haben dies nun in Indien erledigen lassen. Inzwischen ist Business Process Outsourcing ein gängiger Begriff geworden. Das was dahinter steckt ist im Grunde genommen 3rd Party Maintenance. Nur der Name hat sich geändert [Ni08].

Die Vorteile eines Outsourcings des Wartungsbetriebs liegen auf der Hand:

- Erstens, ist es Sache des Outsourcing-Partners, die Software zu sanieren. Wenn er dadurch kosten spart, wird er dies tun, aber ohne die Kontinuität der Dienstleistung zu gefährden oder den Rahmen des vereinbarten Wartungsbudgets zu springen.
- Zweitens, wird die Wartungsdienstleistung in einem Vertrag zwischen dem Kunden und dem Dienstleister bis ins kleinste Detail abgestimmt. Der Prozess wird vertraglich im SLA festgehalten und die geringste Abweichung davon wird mit einer Konventionsstrafe geahndet.
- Drittens, wird der Dienstleister dafür sorgen, dass er mit den besten und modernsten Werkzeugen ausgestattet ist, denn nur so wird er effizient arbeiten können. Es liegt in seinem Interesse, die Wartungsproduktivität zu steigern, denn mit jeder Produktivitätssteigerung verdient er mehr.
- Viertens, werden die Mitarbeiter des jeweiligen Dienstleisters professionelle Wartungsprogrammierer mit einer speziellen Ausbildung sein, die motiviert sind, ihre Arbeit best möglichst zu verrichten. Dies steht im starken Gegensatz zur amateurhaften, unmotivierten Verhalten der meisten Anwenderprogrammierer, die ihren Job als Programmpfleger nur widerwillig ertragen. Es wäre sogar sinnvoll und möglich, das externe Wartungspersonal nach Leistung zu bezahlen, z. B. nach korrigierten Fehlern und durchgeführten Änderungsanträgen.
- Fünftens, hat der Anwender eine Garantie, dass seine Systeme termingerecht und mit einer zugesicherten Qualität fortgeschrieben werden. Er ist nicht länger von einzelnen Personen im seinem Betrieb abhängig, deren Leistung er nicht bestimmen kann.

Das Outsourcing der Systemwartung und -weiterentwicklung ist somit ein wichtiger Schritt in Richtung Industrialisierung des Software-Managements. Dadurch werden die Wartungsabläufe optimiert, die Wartungsleistung messbar und die Wartungskosten transparent. Die Reverse-Engineering Aufgaben sind in der Verantwortung des Dienstleisters. Er hat den Hauptnutzen von einer Nachdokumentation der Software. Deshalb soll er sich um das Reverse Engineering kümmern. Es obliegt ihm ebenfalls, dafür zu sorgen, dass die funktionale Äquivalenz erhalten bleibt. Es werden hochwertige Wartungswerkzeuge eingesetzt, da dies im Interesse des Service Providers liegt, die Wartungsaufträge möglichst effizient und ohne Qualitätsverlust durchzuführen. Dass der Regressionstest möglichst systematisiert und automatisiert wird, versteht sich von selbst, denn dadurch kann der Anbieter die meisten Kosten einsparen [SHT05].

3 Der Wartungsprozess

Solange die Anwendungssoftware hausintern gewartet wird, hat der Anwender kaum eine Möglichkeit den Wartungsprozess zu straffen. Zu eng sind die Verflechtungen zwischen dem IT Personal und dem Fachpersonal. Es läuft alles informal ab. Oft erhalten die zuständigen Wartungsprogrammierer ihre Aufträge per Telefon oder im direkten Gespräch. Erst wenn die Software außer Haus ist, hat der Anwenderbetrieb die Möglichkeit, den Wartungsprozess richtig zu formalisieren und die Wartungsdienstleistungen nach Nutzen zu berechnen.

Die wichtigste Voraussetzung für einen effizienten Wartungsbetrieb ist, zwischen technischen Rahmen und fachlichen Inhalt unterscheiden zu können. Deshalb empfiehlt sich ein Schichtenmodell. Der Dienstleister ist vertraglich verpflichtet den Rahmensoftware – Datenbanksystem, Enterprise Bus und sonstige Middleware – bereitzustellen und zu erhalten. Das Budget dafür ist im Wartungsvertrag vorgesehen. Dazu kommt eine Serviceschicht allgemeingültiger fachlichen Funktionen, die von allen einzelnen Fachanwendungen gemeinsam benutzt werden. Diese könnten als Subroutinen, Klassenbibliotheken oder gar als Web Services implementiert werden. Ihre Erstellung und Erhaltung muss von den Gemeinkosten gedeckt werden.

In der Mitte befindet sich die Anwendungsschicht mit den Applikationen, sprich Geschäftsprozessen, der einzelnen Fachgruppen oder Fachdienste. Für jeden Geschäftsprozess wird ein Softwareprodukt definiert, bestehend aus einem oder mehreren Teilsystemen. Ein Teilsystem setzt sich aus Komponenten zusammen und Komponente bestehen wiederum aus Klassen oder Modulen samt deren Schnittstellen. Diese hierarchische Gliederung der Software – die so genannte Produktstruktur – ist streng einzuhalten (siehe Abbildung 2).

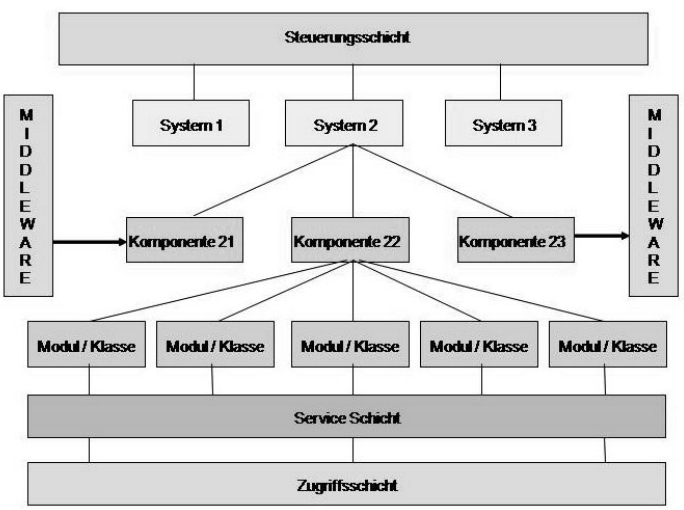

Abbildung 2: Struktur eines Softwareproduktes

Die jeweilige Fachgruppe hat das Recht jederzeit Wartungsaufträge zu stellen um ihr Produkt zu verändern. Die Wartungsaufträge sind in drei Klassen zu teilen:

- Fehlerberichte,
- Change Requests und
- Neue Anforderungen.

Der Dienstleister ist verpflichtet für jeden einzelnen Wartungsauftrag eine Kostenschätzung abzugeben. Sofern der Anwender dies akzeptiert wird der Auftrag durchgeführt und danach der Betrag bezahlt, den der Dienstleister vorgeschlagen hat. Die zuständige Fachabteilung muss wissen, was ihnen die Korrektur, Änderung oder Erweiterung wert ist. Wenn sie meint, die Kosten sind zu hoch kann sie den Auftrag stornieren. Kein Wartungsauftrag muss durchgeführt werden und wenn ja, muss die Fachabteilung bereit sein den vollen Preis dafür zu zahlen. Auch auf Fehlerkorrekturen kann verzichtet werden, wenn die Kosten der Fehlerbehebung zu hoch erscheinen. Es kommt vor allem darauf an Kostentransparenz zu schaffen und dies ist nur möglich wenn jeder Auftrag getrennt abgerechnet wird (siehe Abbildung 3).

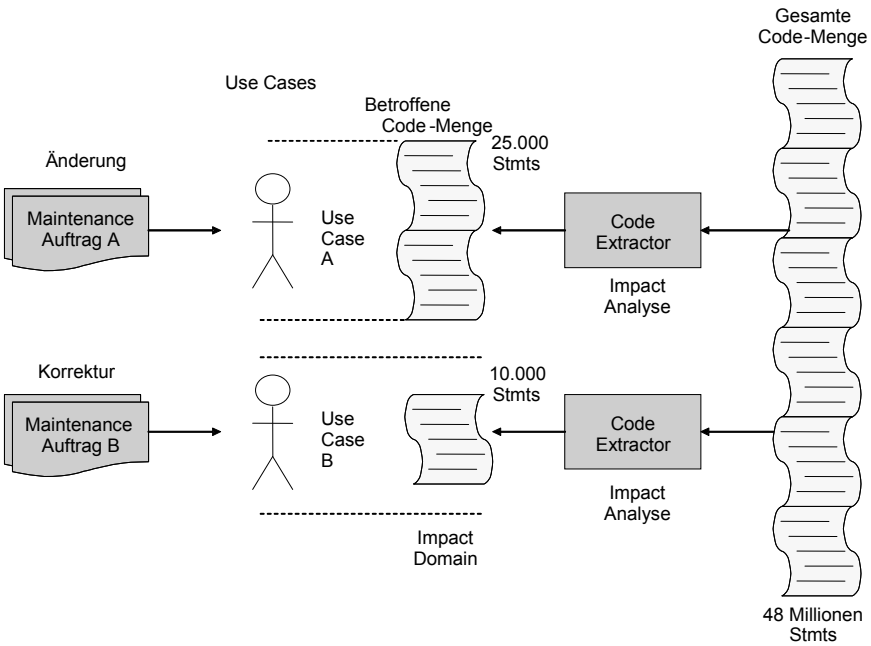

Abbildung 3: Verknüpfung der Wartungsaufträge mit dem Code

Zur Erleichterung der Kostenschätzung und der Wartungsdurchführung müssten alle Wartungsaufträge sich auf bestehende oder neue Anwendungsfälle – Use Cases – beziehen. Der Wartungsbetrieb ist verpflichtet die Beziehungen zwischen den Anwendungsfällen und den Komponenten, Klassen und Moduln in einer Repository zu pflegen. Mit Hilfe der Repository werden betroffene Codebausteine aus der Gesamtcodemenge abgezogen und zur näheren Analyse bereitgestellt. Diese Bausteine unterliegen anschließend einer automatisierten Impact-Analyse aus der ihre Größe und Komplexität hervorgehen. Die Größe lässt sich in Anweisungen, Object-Points oder Function-Points errechnen. Durch eine Verbindung zur Produktivitätsdatenbank können die Kosten der Wartungsaufträge schnell und relativ zuverlässig geschätzt werden [Sn04].

Sofern die zuständige Fachgruppe mit dem Preis und der Dauer einverstanden ist, ist der Dienstleister verpflichtet den Wartungsauftrag innerhalb der vereinbarten Zeit und zu den vereinbarten Kosten abzuwickeln. Wenn er dies nicht schafft, droht eine Konventionsstrafe. Der Anwenderbetrieb ist seinerseits verpflichtet die durchgeführten Änderungen abzunehmen. Dies kann auf zweierlei Weise geschehen, zum einen durch einen dynamischen Abnahmetest und, zum zweiten, durch eine statische Analyse des Codes und der Dokumente. Sie dürfen von der vereinbarten Qualitätsnorm, z. B. die Einhaltung gewisser Codierregel und die Erreichung gewisser Testüberdeckungsziele, nicht abweichen. Wenn ja, hat der Kunde das Recht die Änderungen abzulehnen und braucht dafür nicht zu bezahlen.

Da ein Wartungsbetrieb von einem Anwender allein nicht immer ausgelastet wird, empfiehlt es sich für den Wartungsbetrieb immer mehrere Systeme gleichzeitig zu pflegen. Dies bedarf einer besonders sorgfältiger Planung und einer strenger Organisation. Es versteht sich vom allein, dass der Outsourcing Partner alles daran setzen wird, seine Kosten zu minimalisieren und trotzdem seine vertraglichen Verpflichtungen einzuhalten. Das Ziel der Wartung ist es, einen gerade ausreichenden Wartungsservice zu möglichst niedrigen Kosten anzubieten.

4 Die Wartungsorganisation

Was die Organisation der Softwarewartung anbetrifft, empfiehlt es sich, den Wartungsbetrieb als Spiegelbild des Anwenderbetriebes zu organisieren. Für jede Fachabteilung sollte es eine entsprechende Wartungsprojektgruppe geben. Diese fachlich ausgerichteten Projektgruppen sind für die Wartung und Weiterentwicklung der fachlogischen Komponente zuständig. Die fachlogischen Komponenten sind in dem einheitlichen technischen Rahmen, bzw. in der gemeinsamen Architektur, eingebettet. Die Architektur wird von einer separaten Gruppe gepflegt. Sie besteht aus mehreren Schichten, darunter eine Prozess-Steuerungsschicht, eine gemeinsame Serviceschicht und eine Datenverwaltungsschicht. Es ist wichtig, dass sich die Architektur unabhängig von den Anwendungskomponenten weiterentwickeln lässt. Deshalb muss der fachliche Inhalt von der technischen Verpackung getrennt bleiben (siehe Abbildung 4).

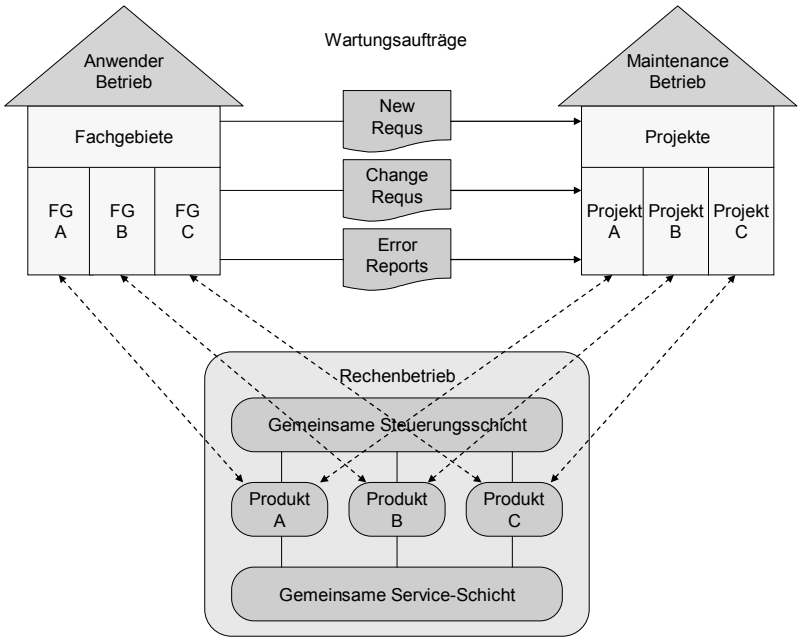

Abbildung 4: Wartungsorganisation entspricht der Software-Architektur

5 Das Service Level Agreement (SLA)

Die rechtliche Grundlage für den oben geschilderten Wartungsprozess ist das Service Level Agreement – eine rechtsverbindliche Vereinbarung zwischen dem Anwenderbetrieb und dem externen Wartungsbetrieb [Bö08]. Die SLA für die Wartung legt die Rechte und Pflichten beider Parteien fest und schreibt den Prozess vor. Eine wichtige Ergänzung dazu ist der Eskalationsplan. Dieser regelt was passiert wenn Pflichten nicht eingehalten werden. Dazu gehört der Ausmaß der Geldstrafen und die Bedingungen für die Terminierung des Vertrages. Es wird z. B. vorgesehen, im Falle einer Termin- oder Kostenüberschreitung, dass der Auftraggeber einlenkt und mehr Zeit bzw. mehr Geld zubilligt. Wenn er aber dazu nicht bereit ist, ist der Auftragnehmer verpflichtet, die Strafe auf sich zu nehmen. Vorausgesetzt wird, dass alle Aufträge nach einer Norm spezifiziert werden. Der Auftragnehmer hat also durchaus das Recht, Aufträge aufgrund mangelnder Qualität abzulehnen, genau so wie der Auftraggeber das Recht hat durchgeführte Änderungen und Korrekturen abzulehnen, falls diese die vereinbarten Qualitätskriterien nicht erfüllen.

Den Qualitätsvorschriften kommt auf dieser Weise eine enorme Bedeutung zu. Im Gegensatz zu hausinternen Wartungsprozessen bei denen solche Qualitätsvorschriften nicht ernst genommen werden, spielen sie hier eine zentrale Rolle. Wer gegen sie stößt muss den Verstoß verantworten. Demzufolge wird auch die Qualität der Normen zunehmen. Beide Parteien – der Kunde wie auch der Dienstleister – müssen mit der Norm zur Gestaltung der Wartungsaufträge einverstanden sein. Das Gleiche gilt für die Norm zur Abnahme der durchgeführten Änderungen am Produkt. Der Auftraggeber muss sich streng an dieser Norm halten, wenn er die Änderungen abnimmt und darf nur Abweichungen von der Norm beanstanden. Dies klingt zwar sehr bürokratisch, aber nur so sind größere Konflikte zu vermeiden. So weiß jeder woran er ist [ZHB05].

Wenn also von Industrialisierung des Wartungsbetriebes die Rede ist, dann müssen als erstes die organisatorischen Rahmenbedingungen geschaffen werden. Dies kann wiederum am besten ein Outsourcing-Vertrag mit einem Service Level Agreement gewährleisten. Allerdings muss der Vertrag richtig formuliert sein, wie die folgenden Fallstudien zeigen.

6 Fallstudien in Maintenance Outsourcing

Hier werden zwei Fälle geschildert bei denen Anwendungssysteme an externe Outsourcing Partner übergeben worden. Im ersten Fall war das Ergebnis wegen der mangelhaften Rechtslage und des ungenügend geregelten Prozesses unbefriedigend. Im zweiten Fall kann man von einer echten Industrialisierung der Softwarewartung sprechen.

6.1 Die Arbeitslosengeldauszahlung ALU-II – ein missglücktes Outsourcing

Die ALU-II Software für die Auszahlung der Arbeitslosengelder ist ein Beispiel für missglückte Outsourcing-Projekte. Die Software wurde ursprünglich von einem kleinen Softwarehaus für die Bundesagentur für Arbeit entwickelt. Ehe es zum Abschluss der Entwicklung kam, ist die besagte Firma in Konkurs gegangen. Da T-Systems Generalunternehmer war, müsste sie die Software übernehmen und das System schlecht oder recht zu Ende bringen. Damit übernahm sie auch die Verantwortung für die Wartung und Weiterentwicklung des Systems. Seit der Freigabe der ersten Version Anfang 2005 gab es ständig Probleme. Schon gleich zu Beginn erwies sich das System als untauglich. So war es nicht möglich, die veränderten Krankenkassenbeiträge im System nachzuvollziehen. Folglich wurden monatlich 25 Millionen Euro zu viel an die Krankenkassen überwiesen. Der Fehlbetrag summierte sich am Ende des ersten Jahres auf 364 Millionen Euro [CW08].

Dies war jedoch nur der Anfang. Es gelang dem Lieferanten weder, die Fehler zu korrigieren noch die Änderungsanträge in einer vertretbaren Zeit durchzuführen. Demzufolge gibt es inzwischen mehr als 146 Umgehungslösungen, welche die Sachbearbeiter ausführen müssen. Dies verursacht den Verlust von durchschnittlich zwei Sachbearbeiterstunden pro Tag. Allein in einem Bundesland produzierten die Fehler der Software jährliche Zusatzkosten in Höhe von 230 bis 380 Millionen Euro. Bundesweit betrachtet sind das mehr als 2 Milliarden Euro pro Jahr. Gutachter haben festgestellt, dass das IT-System mehr kostet, als wenn die Beamten die Gelder manuell auszahlen würden. Das ALU-II Software-System ist also das Beispiel für einen negativen ROI. Schuld an der Lage ist nicht der Dienstleister, Schuld sind die Verträge. Es wurde versäumt, die Pflichten und Rechte beider Partner genau festzulegen. Folglich handelte jeder nach der eigenen Interessenslage.

Sollte es je zu einem neuen Vertrag kommen, muss der Wartungsprozess und die einzelnen Wartungsdienste genauer spezifiziert werden. Es muss auch jede Wartungsleistung vorher kalkuliert und entsprechend vergütet werden. Falls die Fehlerkorrekturen und Änderungsanträge nicht korrekt oder nicht zeitgerecht durchgeführt werden, muss der Auftragnehmer für den Schaden aufkommen. Hinzu kommt, dass die ALU-II Software hier in Deutschland ohne ausreichende Automatisierung fortentwickelt wird. Wenn solche Arbeiten in einem Hochlohnland wie Deutschland verrichtet werden, müssen sie im hohen Grade durch automatisierte Werkzeuge unterstützt werden, so wie in der zweiten Fallstudie der Fall ist. Eine Industrialisierung des Software-Managements setzt nicht nur eine solide Vertragsbasis voraus. Sie verlangt auch ein hohes Grad an Automatisierung.

6.2 Das GEOS Wertschriftenverwaltungssystem - ein Beispiel für den industrialisierten Wartungsbetrieb

Das GEOS System für die Verwaltung von Wertschriften bei den österreichischen Banken liefert ein Beispiel dafür, wie komplexe Systeme gemanaged werden sollten. Dort gibt es einen ausgereiften Wartungsprozess, unterstützt durch zahlreiche automatisierte Werkzeuge. Der Code wurde mehrfach saniert und optimiert. Die 7,5 Millionen Code-Zeilen wurden auf 5,5 Millionen bzw. auf 2,2 Millionen Anweisungen reduziert. Die mittlere Komplexität der Module ist von 0,76 auf 0,59 gesunken und die mittlere Qualität von 0,52 auf 0,63 gestiegen. Schließlich wurde die Wartungsmannschaft von 250 auf 180 Mitarbeiter reduziert. Alle Fehlerkorrekturen und Änderungsanträge werden in einem gemeinsamen Repository abgelegt und analysiert. Eine automatische Auswirkungsanalyse ermittelt die Größe der Auswirkung und setzt dies in einen Aufwand um. Dadurch bekommt der Kunde immer sofort ein Angebot bezüglich der Kosten seiner Änderungswünsche. Ist er damit einverstanden, fließt der Änderungsantrag in das nächste Release ein. Neue Releases folgen in Intervallen von drei Monaten. Fehler werden innerhalb von 48 Stunden korrigiert und getestet. Die Korrekturen werden innerhalb der 48 Stunden auf dem System wirksam. Inzwischen ist die jährliche Fehlerrate unter 0,0002 bzw. auf 2 Fehler pro 10.000 Anweisungen gesunken [BS03].

Der Regressionstest ist zu 100% automatisiert. Nicht nur die Dialoge werden von Robotern automatisch bedient, auch die Batch-Prozesse werden automatisch angestoßen und überwacht. Am Ende werden alle Ergebnisse, sowohl die Bildschirmausgaben als auch die Datenbankinhalte automatisch abgeglichen und fehlerhafte Ergebnisse ausgewiesen. Dazu wird die Testüberdeckung gemessen und eine Teststatistik aufbereitet. Zuständig für die Wartung und Weiterentwicklung vom GEOS ist übrigens das gleiche Haus wie für die Wartung und Weiterentwicklung der ALU-II Software. Das bezeugt wie schwer es ist, einheitliche Wartungsprozesse in einem Unternehmen durchzusetzen (siehe Abbildung 5).

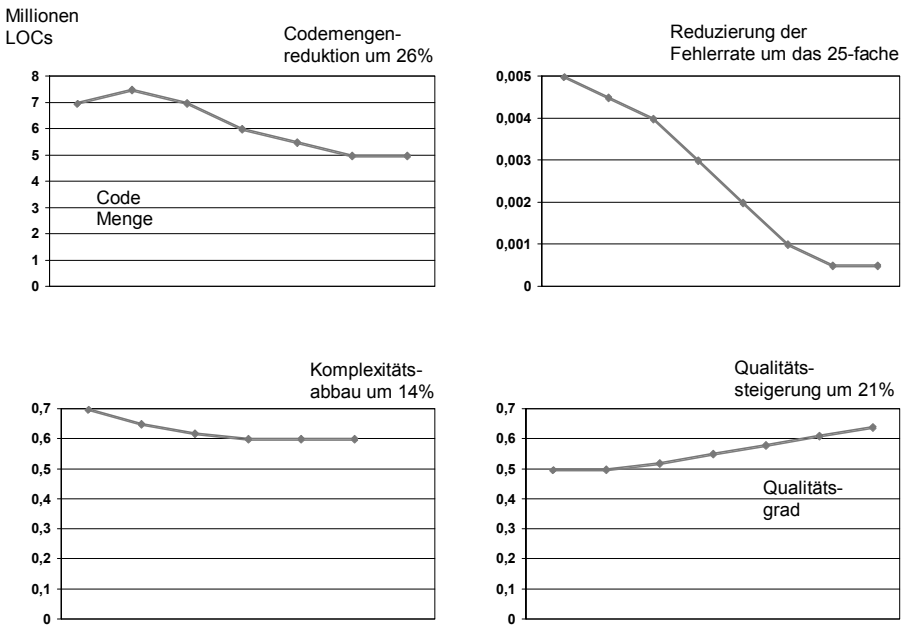

Abbildung 5: Monitoring des GEOS Wartungsbetriebes

7 Zusammenfassung

Das GEOS Projekt bezeugt, dass eine Industrialisierung des Software-Managements durchaus möglich ist. Es ist vor allem eine Frage der Organisation und der Verträge. Wenn sie stimmen, folgen die anderen Maßnahmen fast von alleine:

- Der Wartungsprozess wird optimiert;
- Die Wartungsumgebung wird modernisiert;
- Das Wartungspersonal wird qualifiziert;
- Das Wartungsprodukt wird ständig saniert.

Die regelmäßige Sanierung der Software wird zum festen Bestandteil des Produktlebenszyklus, ebenso, wenn nötig, die Migration auf eine neue Plattform. Wichtig ist, dass die Wartung von einem separaten Dienstleistenden nach fest vereinbarten Kostensätzen auf der Basis eines detaillierten Service Level Agreements gewährleistet wird. Das Ziel muss es sein, den laufenden Prozess ständig zu überwachen und zu verbessern. Die Verbesserungen müssen messbar sein. Deshalb besteht die Notwendigkeit einer Metrik-Datenbank, mit deren Hilfe der Dienstleistende und der Kunde die Qualität der Dienstleistung messen und vergleichen lassen. Industrialisierung erfordert Messbarkeit, erst recht für Software-Management.

Literaturverzeichnis

[Pi97] Pigoski, T.: Practical Software Maintenance – Best Practices for managing your software Investment, John Wiley & Sons, New York, 1997, S. 101

[Bo99] Boehm, B. et al.: Software Cost Estimation with COCOMO-II, Prentice-Hall, Upper Saddle River, N. J., 1999, S. 14

[Hu00] Hughes, B.: Practical Software Measurement, McGraw-Hill, Maidenhead, 2000, S. 47

[Ga05] Gartner Group: "Distribution of Costs in the IT-Industry", Gartner-Report Nr. 1005-4, Chicago, 2005

[BS95] Brodie, M.; Stonebraker, M.: Migrating Legacy Systems–Gateways, Interfaces & Incremental Approach, Morgan Kaufmann Pub., San Francisco, 1995, S. 6

[SAW08] Sneed, H.; Ackermann, E.; Winter, A.: Softwaremigration, dpunkt, Heidelberg, 2008, S. 15

[SS03] Sneed, H.; Sneed, S.: Web-basierte Systemintegration, Vieweg, Wiesbaden, 2003, S. 71

[Sn08] Sneed, H.: „20 Years of Software Reengineering – A Resume", Proceedings of 10th GI Workshop on Software Rreengineering, Ed. Gimnich, Kaiser, Quante, Winter, Bad Honeff, May 2008, S. 115

[Sn91] Sneed, H.: Economics of Software Reengineering, Journal of Software Maintenance, Vol. 3, Nr. 3, Sept 1991, S. 163

[Th95] Thomsett, R.: "Project Pathology – A Study of Project Failures, American Programmer, Vol. 8, Nr. 7, Juli 1995, S. 9

[Ca88] Calow, H.: "Practicing 3rd Party Maintenance", in Proc. of First European Workshop on Software Maintenance, British Centre of Software Maintenance, Durham, Sept. 1988, S. 110

[Ni08] Nicklisch, G.; Borchers, J.; Krick, R.; Rucks, R.: IT-Near- und –Offshoring in der Praxis, dpunkt, Heidelberg, 2008

[SHT05] Sneed, H.; Hasitschka, M.; Teichmann, M.-T.: Software-Produktmanagement, Wartung und Weiterentwicklung bestehender Anwendungssysteme, dpunkt, Heidelberg, 2005

[Sn04] Sneed, H.: „A Cost Model for Software Maintenance", Proc. of 20th International Conference on Software Maintenance, IEEE Computer Society Press, Chicago, Sept. 2004, S. 264

[Bö08] Böttcher, R.: IT-Servicemanagement mit ITIL V3, dpunkt, Heidelberg, 2008, S. 38

[ZHB05] Zarnekow, R.; Hochstein, A.; Brenner, W.: Service-orientiertes IT Management ITIL Fallstudien, Springer, Berlin, 2005, S. 30

[CW08] CW-Redaktion: "ALU-II Pannensoftware vor der Ablösung", Computer Weekly, Nr. 11, Feb. 2008, S. 8

[BS03] Broessler, P.; Sneed, H.: "Critical Success Factors in Software Maintenance", Proc. of ICSM-2003, IEEE Computer Society Press, Amsterdam, Sept. 2003, S. 190