Outsourcing
in der Softwareentwicklung

Seminararbeit im Rahmen des Integrationsseminars
der Lehrstühle Organisation und Wirtschaftsinformatik II
„Implementierungsmanagement“
im Wintersemester 2004/05

Thema Nr. 8

Vorgelegt am Betriebswirtschaftlichen Institut der Universität Stuttgart
Abteilung IX
Lehrstuhl für ABWL und Wirtschaftsinformatik,
insbesondere Unternehmenssoftware

Von Reinhard Kuntz,
Matrikelnummer 1966907

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

In wirtschaftlich angespannten Zeiten orientiert sich das Firmenmanagement verstärkt an den Grundtugenden des Wirtschaftens. Jede Aktivität wird geprüft, ob sie mittelbar oder unmittelbar einen positiven Beitrag zum Unternehmenswert leistet. Ist dies nicht der Fall, ist sie verzichtbar oder sogar schädlich.[1] In der Vergangenheit wurde vom Management auf solche Situationen zumeist mit Reorganisation, Hierarchieverschlankung und verstärktem Umorientieren auf Geschäftsprozesse reagiert. Neuerlich stehen in diesem Zusammenhang auch die IT und spezielle Bereiche der IT, beispielsweise die Softwareentwicklung, auf dem Prüfstand.[2] Meist mit dem Argument der Kostenreduktion wird in diesem Bereich in jüngster Zeit Outsourcing oder sogar Offshoring als strategische Maßnahme herangezogen. Dieses Vorgehen ist jedoch sehr umstritten und wird zum Teil sogar als Vertuschen früherer Managementfehler betrachtet. Auch wird oft kritisiert, dass die Probleme des IT-Management durch Outsourcing nicht gelöst, sondern lediglich abgeschoben werden.[3] Kritisiert wird oftmals außerdem, dass die Entscheidung für oder gegen Outsourcing meist nur anhand von Kosten orientierten, kurzfristigen Argumenten und „einfachen“ Entscheidungsinstrumentarien gefällt wird, wobei zu wenig auf die Möglichkeit inherente Vorteile bei der Eigenerbringung bilden zu können geachtet werde.[4]

Dies lässt den Verdacht aufkommen, dass der Entscheidungsprozess, welcher oftmals sehr nah an das Vorgehen bei einer in der produzierenden Wirtschaft entstandenen „Make or Buy“-Entscheidung angelehnt ist, nicht die richtige Methode darstellt.

In dieser Seminararbeit soll nun beleuchtet werden, welche Besonderheiten beim Outsourcing in der Softwareentwicklung auftreten können.

Zunächst wird auf die Begrifflichkeiten „Outsourcing“ und „Softwareentwicklung“ eingegangen. Daraufhin werden ein „Make or Buy“- und ein Outsourcing-Praxisbeispiel vorgestellt, mit deren Hilfe einige für die Softwareentwicklung spezifische Sachverhalte herausgearbeitet werden. Hierbei soll versucht werden herauszukristallisieren, warum Outsourcing insbesondere in der Softwareentwicklung sehr umstritten ist. Abschließend wird ein Lösungsansatz formuliert, der thesenartig eine Möglichkeit aufzeigen soll dem Kostendruck gerecht zu werden und trotzdem die strategische Bedeutung der Softwareentwicklung nicht zu vernachlässigen.

[1] vgl. Dobschütz (1995), S. 110
[2] vgl. Dobschütz (1995), S. 110f
[3] vgl. Steppan (2004), URL siehe Quellenverzeichnis
[4] vgl. Bruch (1998), S. 17

1.1 Outsourcing

Der Begriff „Outsourcing“ ist ein Kunstwort, welches auf **Out**side Re**sourc**e Us**ing**[5] zurückzuführen ist. Die Semantik des Begriffes lässt sich jedoch nicht einheitlich definieren. Das Spektrum reicht vom Nutzen kleiner Dienstleistungen bis hin zum Bilden „unabhängiger“ (Dienstleistungs-)Tochterunternehmungen. Zumindest ist jeglichen Definitionen gemein, dass Outsourcing auf eine mittel- meist jedoch langfristige Vergabe der Aktivitäten angelegt ist.[6] Auch besteht Einigkeit darüber, dass es sich um Aktivitäten handelt, welche bisher vom eigenen Unternehmen erbracht wurden.
In dieser Seminararbeit soll der Begriff „Outsourcing“ wie folgt verstanden werden:

Outsourcing stellt eine typische Ausprägung arbeitsteiligen Wirtschaftens dar. Hierbei soll der Zugriff auf außerhalb der Unternehmung liegenden Ressourcen und Potentiale ermöglicht werden. Prinzipiell handelt es sich dabei um Sach- sowie um Dienstleistungen, die bisher selbst erbracht wurden und in Zukunft über einen mittleren bis langfristigen Zeitraum bezogen werden sollen.[7]

An sich ist Outsourcing somit nichts Neues. Das traditionelle „Make or Buy“-Vorgehen wurde zur Minimierung der Fertigungstiefe herangezogen. Outsourcing soll nun zusätzlich die Dienstleistungstiefe herabsetzen. Dabei wird in besonderem Maße betont, dass dies nicht aus rein wirtschaftlichen, sondern auch strategischen Überlegungen heraus geschehe.[8]

1.2 Softwareentwicklung

Unter dem Begriff „Softwareentwicklung“ wird oft nur das „reine“ Entwicklungsprojekt in Verbindung gebracht. Idealtypisch beinhaltet der Begriff der Softwareentwicklung jedoch den gesamten Lebenszyklus von Software.[9] Der Softwarelebenszyklus besteht, unabhängig vom verwendeten Software-Prozess-Modell, stets aus allen in Abbildung 1 dargestellten Elementen.[10]

Je nach Prozessmodell können die Elemente unterschiedlich gegliedert und zusammengefasst sein. Manche Software-Prozess-Modelle sehen außerdem Zyklen zur Wiederholung von Elementen unter bestimmten Bedingungen vor, beispielsweise bei dem

[5] vgl. Münzl, Neesen (1997)
[6] vgl. Mertens, Knolmayer (1995)
[7] vgl. Bruch (1998), S. 17
[8] vgl. Bruch (1998), S. 24
[9] vgl. Balzert (2000), S. 39
[10] vgl. Ludewig (2003), S. „5-1“

Evolutionären Prozessmodell.[11] Bei manchen Prozessmodellen kann der Eindruck entstehen, dass einzelne der in Abbildung 1 aufgeführten Elemente nicht vorhanden sind. Bei genauerer Betrachtung zeigt sich jedoch, dass diese implizit in anderen Elementen eingebunden sind. Dieser Sachverhalt ist beispielsweise beim Extreme Programming (XP) zu beobachten.[12] Bei XP sind keine explizite Integrations- und Testphase vorgesehen, da in kurzen Abständen beim Codieren integriert und getestet wird.[13]

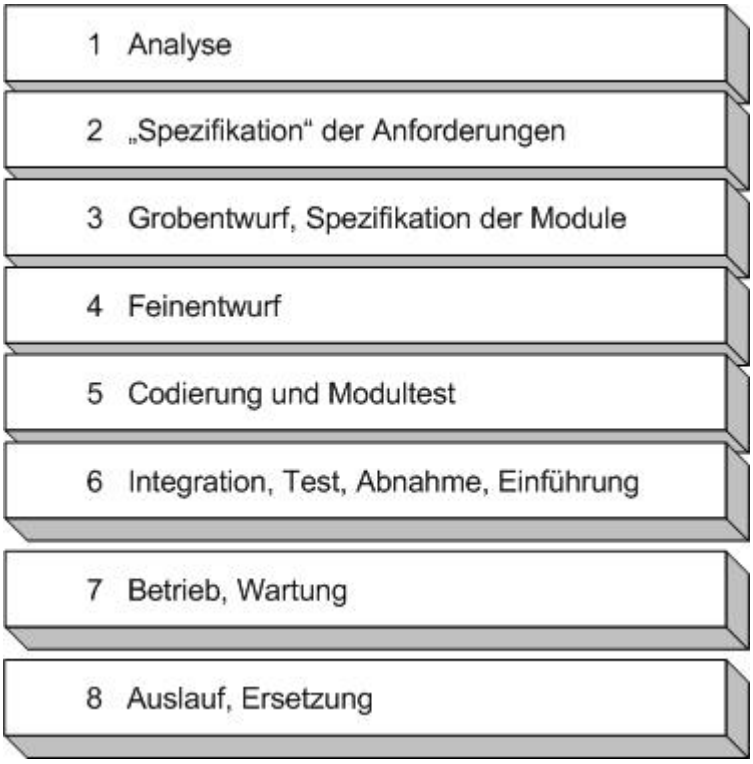

Abbildung 1 - Elemente des Softwarelebenszyklus[14]

Die Elemente 1 bis 6 der Abbildung 1 werden in der Regel als Teil eines Projektes durchgeführt. Bei modernen Softwareentwicklungsverfahren ergibt sich für die Elemente 1 bis 6 eine Aufwandsverteilung von 65% vor der Codierung, 15% für die Codierung und 20% nach der Codierung. Hierbei ist der Aufwand für die Elemente 7 und 8 nicht mitbetrachtet.[15]

Das Elemente 7 „Betrieb, Wartung" muss zu den ständigen Aufgaben einer Softwareentwicklungsabteilung gezählt werden. Typischerweise liegt der anteilige Aufwand für das Element 7 also erheblich höher als für die restlichen Elemente. Über den gesamten

[11] vgl. Balzert (2000), S. 120-122
[12] vgl. Wells (2001), URL siehe Quellenverzeichnis
[13] vgl. Ludewig (2003b)
[14] in Anlehnung an Ludewig (2003)
[15] vgl. Ludewig (2003), S."4-3" basierend auf Baskette (1987) o.O.

Softwarelebenszyklus hinweg entfallen etwa 2/3 der Kosten auf die Wartung. Die Wartung setzt sich dabei aus der Fehlerbehebung, der Anpassung und der Erweiterung zusammen.[16]

Das in Abbildung 1 gezeigte Element 8 „Auslauf, Ersetzung“ kann in den meisten Fällen dem Projekt einer Ersetzungssoftware und deren Elementen 1 bis 6 zugerechnet werden.

2 Praxisbeispiele

Im Rahmen dieser Seminararbeit wurden Firmeninterviews durchgeführt. Die Namen der interviewten Firmen und die der Gesprächspartner dürfen aus Image- und Geheimhaltungsgründen leider nicht genannt werden. Im Folgenden sind nun ein „Make or Buy“-Beispiel eines Fahrzeugherstellers und ein Outsourcing-Beispiel eines Finanzdienstleisters aufgeführt.

2.1 Beispiel 1: „Make or Buy“

Nach Auskunft des Fahrzeugherstellers wurden früher Getriebe selbst gefertigt. Heute werden diese zwar noch selbst entwickelt, die Fertigung jedoch an andere Firmen vergeben. Bei der „Make or Buy“-Entscheidung werden verschiedene Anbieter sowie die Eigenproduktion betrachtet. Der Vergleich mit der Eigenproduktion wird jedoch nur zu Kontrollzwecken durchgeführt. Diese Vergleichsvorgänge werden für jedes Fahrzeugmodell von neuem durchlaufen. Mittlerweile ist es Strategie, die eigenen Anforderungen zu entwickeln, mit Hilfe von Lastenheft sowie Pflichtenheft diese festzuschreiben, um auf deren Basis die Getriebe produzieren zu lassen. Da die Fertigungsfirmen ohnehin auch weitere Getriebereihen herstellen und somit über notwendige Maschinen und Fertigungs-Know-How verfügen, können diese erhebliche Skalenvorteile geltend machen. Da sich dieser Sachverhalt in Zukunft nach Einschätzung des Fahrzeugherstellers nicht ändern werde, besteht kein Interesse an fertigungsspezifischem Know-How. Eigenes Know-How fließt nur in vertretbarem Maß ab. Für das im Auftrag produzierende Unternehmen ist zwar ersichtlich, auf welche Eigenschaften Wert gelegt wird und welche Rahmendaten erfüllt werden müssten, jedoch sind die Gründe für diese nicht ersichtlich. Ein Vorsprung in der Entwicklung basiert jedoch gerade auf den Überlegungen hinter den Anforderungen. Es wird also tatsächlich die Fertigung und das Fertigungs-Know-How bezogen, ohne dass nennenswert Know-How herausgegeben wird.

[16] vgl. Balzert (2000), S. 1090-1094

Das Entscheidungsprozedere bei einer solchen „Make or Buy“-Entscheidung sieht, wie schon erläutert, die Erstellung von Lasten- und Pflichtenheft vor. Dies geschieht durch die Entwicklungsabteilung anhand überwiegend spezifizierbarer und überprüfbarer Anforderungen. Softe Anforderungen sind nur sehr wenige enthalten. In Zusammenarbeit mit dem Einkauf wird die „Make or Buy“-Entscheidung getroffen. Hierbei liefert der Einkauf Informationen zu den Zulieferfirmen. Über Jahre hinweg wurde ein Rating dieser Firmen erstellt und laufend gepflegt. Für die Klassifizierung sind insbesondere ausschlaggebend, ob entsprechende Fertigungskapazitäten vorhanden sind und ob das Unternehmen liquide ist und in Zukunft voraussichtlich sein wird. Auch fließt mit ein, ob und welche Erfahrungen mit den entsprechenden Unternehmen schon gemacht wurden. Kosten spielen eigentlich nur im Vergleich Eigenerstellung oder Fremdbezug eine gewichtige Rolle. Bei der Entscheidung, welcher Anbieter gewählt wird, sind die Kosten eher nebensächlich, da die Angebote sich in diesem Punkt durch ausreichend Konkurrenz nicht wesentlich unterscheiden. Qualität und Planungssicherheit sowie kalkulierbares und geringes Risiko sind wesentlichere Kriterien. Ist die Entscheidung in den Fachabteilungen gefallen, wird diese dem Vorstand vorgelegt. Die Rolle des Vorstandes beschränkt sich in den meisten Fällen auf eine Kontrollfunktion und bestätigt daher meist die Entscheidung der Fachabteilungen. Die Einkaufsabteilung ist darauf folgend in Zusammenarbeit mit der Qualitätskontrolle für die Zulieferbeziehungen zuständig.

2.2 Beispiel 2: Outsourcing

Der interviewte Finanzdienstleister verwendete bis vor kurzem zur Rentenmodellberechnung ein gewachsenes Softwaresystem. Dieses bestand aus verschiedensten selbst entwickelten Tools im Zusammenspiel mit Excel, das mit entsprechenden Makros erweitert wurde. Dieses Softwaresystem sollte durch ein System „in einem Guss“ ersetzt werden. Zur Realisierung wurde Outsourcing herangezogen, wobei kein Vergleich zwischen verschiedenen Anbietern gemacht wurde. Den Zuschlag erhielt das Dienstleistungsunternehmen, welches schon für die Bereitstellung von Servern und der Netzwerkinfrastruktur verpflichtet war. Diese Entscheidung wurde zum einen mit dem Argument der geringeren Kosten, zum anderen mit dem Fehlen des eigenen Know-How’s im Bereich des Softwareengineering begründet. Auch spielten die guten Erfahrungen mit dem Dienstleister eine gewichtige Rolle.

Die Entscheidung zum Outsourcing ist auf der Managementebene getroffen worden. Anstelle eines Lasten- und eines Pflichtenheftes ist lediglich ein Rahmenwerk erstellt worden, auf dessen Basis evolutionär entwickelt wurde. Die Berechnungsmodelle, die die Software realisieren soll, sind in Eigenentwicklung entstanden. Eine anpassbare

Standardsoftware ist deshalb am Markt nicht vorhanden. Das Entwickeln der Modelle ist die eigentliche Kernkompetenz des Finanzdienstleisters. Die Software bildet diese ab und macht die Modelle anwendbar. Die entwickelte Software ist Besitz des Finanzdienstleisters. Insourcing-Aktivitäten sind vertraglich nicht geregelt. Auch sind keine Mechanismen eingerichtet worden um das Entwicklungs-Know-How oder weiteres softwarespezifisches Wissen zu erlangen. Regelungen für den Konkursfall des Outsourcing-Anbieters sind ebenfalls keine getroffen worden. Ein entsprechender Vertrag für Serviceleistungen, Wartung und Weiterentwicklung ist jedoch im Entstehen.

Die im ersten Jahr entstandenen Kosten lassen sich in etwa zu 60% der Entwicklung und zu 40% der Wartung zuordnen. Die Grenzen zwischen Wartung und Entwicklung lassen sich jedoch wegen des evolutionären Entwicklungsvorgehens nicht klar definieren.

3 Entscheidungsprozess, Entscheidungsträger und Betroffene

In Beispiel 1 wurde beschrieben, das beim „Make or Buy“ für jeden einzelnen Fall eine Art „bottom up“-Entscheidungsprozess stattfindet. Hierbei ist zu beobachten, dass die Fachabteilung, die die Entscheidung des Managements vorbereitet, nicht selbst von der Entscheidung betroffen ist. Dies liegt daran, dass die Entwicklung von der Produktion getrennt ist. In der Softwareentwicklung ist dies meist nicht der Fall. Wie an der folgenden hypothetischen Überlegung deutlich wird, liegen die Gründe hierfür in der Historie der Softwareentwicklung:

Würde eine Unternehmung ein Bürohaus selber bauen und betreiben oder mieten? In den meisten Fällen eher mieten. Aber, wie fällt die Entscheidung aus, wenn im Unternehmen eine Bauabteilung vorhanden ist? In diesem Fall würde mit hoher Wahrscheinlichkeit die Entscheidung gefällt werden, die Leistungen selbst zu erbringen. Oft sogar auch dann, wenn die Eigenerbringung teurer ist als der Bezug von anderen Unternehmen.[17]

In dieser Weise wurden in der Vergangenheit meist Entscheidungen in der Softwareentwicklung gefällt. Dadurch ist oftmals nicht einmal eine gesonderte Gruppe im Unternehmen vorhanden, die für Einkaufs- oder Selbsterstellungsentscheidungen im Bereich der Softwareentwicklung zuständig ist. Dieser Zustand ist außerdem dadurch bedingt, dass Software völlig andersartige Eigenschaften wie Hardware beziehungsweise wie Produktionsgüter aufweist. Software ist immateriell, weshalb sie nicht gefertigt, sondern

[17] vgl. Dobschütz (1995) S. 108

ausschließlich entwickelt wird.[18] Dies äußert sich beispielsweise auch darin, dass Kopie und Original völlig identisch sind.

Da die betroffenen und die eine Entscheidung vorbereitenden Personen aus demselben Personenkreis stammen, treten Interessenskonflikte auf. Ein Mitarbeiter wird tunlichst vermeiden, eine Empfehlung für Fremdbezug auszusprechen, wenn hierdurch sein Arbeitsplatz gefährdet werden könnte.

Für Fachleute ist eine Sourcing-Entscheidung in der Softwareentwicklung wegen soften Kriterien (siehe Abschnitt „Entscheidungskriterien“) ohnehin schwierig. Für das Management ist es praktisch unmöglich zu beurteilen, ob in der Handlungsempfehlung der Fachabteilung noch zusätzlich persönliche Interessen eine gravierende Rolle gespielt haben. Wie in Beispiel 1 deutlich wird, ist dies im produzierenden Gewerbe erheblich besser möglich, da sich das Entscheidungsobjekt zum größten Teil spezifizieren lässt und dadurch auch vom Management einschätzbar ist.

Diese Problematik führt - wie in Beispiel 2 sichtbar wird - oft zu reinen Managemententscheidungen, die dann nicht aufgrund fachlicher und strategischer Überlegungen geschehen, sondern nur anhand weniger Fakten, wie beispielsweise den Kosten, getroffen werden. Unter Umständen sogar „zur Beseitigung des Problems“, indem der Problembereich einfach durch Outsourcing „entfernt“ wird.

Diese Problematik hat noch weitere Folgen. In der Fertigungsindustrie ist es mittlerweile normal „Make or Buy“-Abwägungen durchzuführen, wie in Beispiel 1 deutlich wird. Die im Zweifel Betroffenen bekommen durch die Trennung von Entwicklung und Fertigung im Normalfall von diesen Prozessen nichts mit. In der Softwareentwicklung ist das jedoch nicht der Fall, wie obige Überlegungen zeigen. Deshalb besteht beim Anstoßen von Outsourcing-Überlegungen verstärkt die Gefahr, dass Mitarbeiter abwandern. Betroffen sind dann insbesondere sehr gut ausgebildete Spezialisten, da diese leicht in anderen Unternehmen eine „sichere“ Stelle finden können. Hierdurch geht dem Unternehmen dann unwiederbringbar Erfahrung und Fachwissen verloren.

4 Entscheidungskriterien

Fertigungsprodukte lassen sich wie in Beispiel 1 deutlich wird, meist sehr genau beschreiben oder sogar spezifizieren. Bei Software jedoch kann ein Großteil nur mit so genannten „soften“ Kriterien beschrieben werden. Hierbei handelt es sich um Kriterien wie beispielsweise eine ergonomische Bedienung, eine flexible Nutzung oder auch eine gute Wartbarkeit, welche nicht messbar sind und oft gegenseitig korrelierend wirkende

[18] vgl. Balzert (2000), S. 26-28

Anforderungen aufweisen.[19] Aus diesem Grund ist eine Kosten-Nutzen-Einschätzung in der Softwareentwicklung erheblich schwieriger als in der Produktion und basiert immer auf einer Reihe von Annahmen.[20] Dies spiegelt sich auch darin wieder, dass im Beispiel 1 ein Lasten- und ein Pflichtenheft verwendet wird, im Beispiel 2 nur ein Rahmenwerk.

Produktionsgüter, wie im Beispiel 1 die Getriebe, werden innerhalb einer Produktionsreihe nicht wesentlich verändert. Software wird in ihrem Lebenszyklus jedoch in erheblichem Maße verändert und angepasst. Das liegt daran, dass Details einer Aufgabe durch Software realisiert werden, welche die Verbindung zwischen Hardware und der natürlichen Umgebung schafft. Software spiegelt somit die Realität.[21] Änderungen in der Aufgabenstellung haben primär Einfluss auf die Software. In Beispiel 2 wäre eine mögliche Änderung der Aufgabenstellung eine veränderte Rentenrechtssprechung. Dementsprechend muss Software häufig angepasst werden und unterliegt einem schnellen Wandel. Hieraus ergeben sich spezifische Anforderungen wie Anpassbarkeit und Wartbarkeit. Hinzu kommt, dass Fehler in Software nicht durch Be- oder Abnutzung entstehen und sich nicht ankündigen, sondern sprunghaft auftreten. Auch ist die Funktionalität von Software nicht testbar. Leider werden aus diesem Grund bei knapp über 50% der Unternehmen erst gar keine Qualitätsanforderungen definiert[22], wie dies auch im Beispiel 2 der Fall war. Hierdurch wird zusätzlich der Vergleich von Software erschwert.

Bei einem Getriebehersteller, der einmal eine aus technischer Sicht gute Getriebeserie geliefert hat, kann davon ausgegangen werden, dass er dies auch wieder machen wird. Bei einem Softwarehersteller ist dieser Schluss nicht möglich, da die Aufgabenstellungen und somit auch das benötigte Know-How, die benötigten Werkzeuge und Sprachen sich erheblich unterscheiden und schneller ändern können.

Dieser Sachverhalt führt zu einem sehr undurchsichtigen Markt im Vergleich zur Zulieferindustrie. In Beispiel 1 wurde beschrieben, dass im Unternehmen eine Kategorisierung der Zulieferer gemacht und ständig aktualisiert wird. Diese Vergleiche werden, wie in Beispiel 2 sichtbar, in der Softwareentwicklung oftmals leichtfertig weggelassen. Im Gegensatz zur Produktion kann in der Softwareentwicklung Qualität, wie schon beschrieben, nur schlecht skaliert werden. Aus diesem Grund fehlt oft ebenfalls eine ausgereifte Metrik zur Bewertung von Unternehmen und ihrer Produkte. Deshalb gestaltet es sich äußerst schwierig, Angebote von Outsourcing-Anbietern zu vergleichen und Erfahrungen zu bewerten. Neuere Verfahren - wie beispielsweise die Beurteilung des Prozessreifegrades anhand einheitlicher Kriterien - lassen eine annähernde Aussage über Unternehmen zu, sind aber bei weitem nicht flächendeckend vorhanden. Auch geben diese

[19] vgl. Balzert (2000), S. 27-28 und S. 34-35
[20] vgl. Ludewig (2003)
[21] vgl. Balzert (2000), S. 34-35 und vgl. Ludewig (2003), S. „2-4“
[22] vgl. Balzert (1998), S. 284

Verfahren, wie beispielsweise das Capability Maturity Modell (CMM)[23] oder das Software Process Improvement and Capability Determination (SPICE)[24] natürlich keine Garantie über die Softwarequalität.

5 Kostenorientierung

Wie im letzten Abschnitt schon beschrieben sind Produktionsgüter zu einem hohem Grad spezifizierbar. Aus diesem Grund sind bei Kostenbetrachtungen nur noch Produktionsfehler von Bedeutung. Produktionsfehler wiederum sind durch Qualitätsprüfung, beziehungsweise Tests im Vergleich mit der Spezifikation, auffindbar. Die Kosten orientieren sich also insbesondere am Stückpreis, in den möglicher Ausschuss vom Produzenten eingerechnet ist. In Beispiel 1 somit der Preis pro Getriebe.

Da in der Softwareentwicklung die Hauptkosten durch die Gehälter der Entwickler entstehen[25], ist eine Argumentation über Einsparungen in diesem Bereich nahe liegend. Outsourcing-Anbieter können mit einem größeren Entwickler-Pool und einer größeren Zahl an Kunden durch entsprechendes Scheduling Leerlaufzeiten der Entwickler vermeiden. Auch kann beispielsweise bei Offshoring zusätzlich an den Gehältern eingespart werden. Hieraus ergeben sich entsprechende Skalenvorteile. In diesen Argumentationen wird jedoch nicht beachtet, dass schon bei der Eigenentwicklung Qualitätskosten, also der Aufwand für Qualität beziehungsweise Kosten aufgrund mangelnder Qualität, die „nackten“ Entwicklungskosten bei weitem übersteigen können[26]. In der Softwareentwicklung ist also eine Orientierung an den Qualitätskosten oder anders ausgedrückt, dem Risiko, unabdingbar. So kann eine Rückrufaktion von Produkten mit Embedded-Software allein durch hohe Stückzahlen den Konkurs bedeuten. Auch in Beispiel 2 kann ein Fehler in der Software dazu führen, dass fehlerhafte Berechnungen gemacht und diese zu einem erheblichen Imageschaden oder sogar Schadenersatzforderungen führen können.

Hinzu kommt, dass die Fehlerkosten mit der Latenzzeit zwischen Fehlerentstehung und Fehlerentdeckung expotentiell ansteigen.[27] Durch den sehr hohen Anteil von soften Anforderungen in der Softwareentwicklung kann es schon in sehr frühen Phasen der Entwicklung zu Fehlern kommen, zum Beispiel durch ein unterschiedliches Verständnis von Anforderungsbeschreibungen. Wie das in Abbildung 2 dargestellte Badewannenmodell zeigt, werden Fehler typischerweise auf derselben Abstraktionsebene entdeckt. Für Fehler in der Definition der Anforderungen bedeutet das, dass diese erst bei der Installation, also einer

[23] vgl. Balzert (1998), S. 362-376
[24] vgl. Balzert (1998), S. 377-383
[25] vgl. Ludewig (2003), S. „4-2“
[26] vgl. Balzert (2000), S. 1093
[27] vgl. Balzert (1998), S. 287

sehr späten Phasen entdeckt werden und sehr hohe Kosten verursachen. Bei Outsourcing kommt durch das erforderliche Kommunizieren der Anforderungen zum Outsourcing-Anbieter eine weitere gravierende Fehlerquelle auf dieser Abstraktionsebene hinzu.

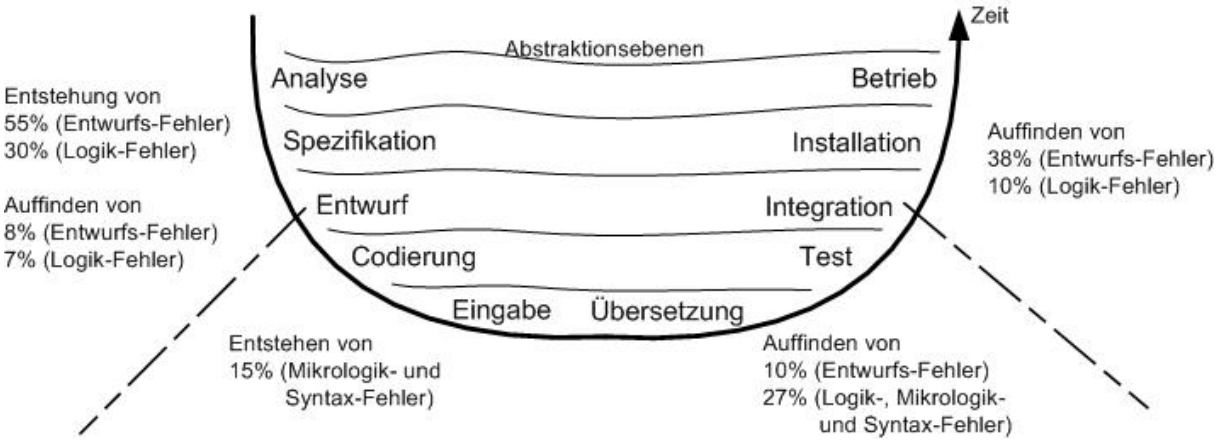

Abbildung 2 - Badewannenmodell[28]

6 Strategie, Kerngeschäft und Know-How

Wie schon im Abschnitt „Problemstellung“ deutlich wurde, wird mittlerweile im Bereich des Outsourcing vermehrt mit Hilfe von strategischen Überlegungen argumentiert. So werden oft nicht mehr die Kosten, sondern das Ziel, sich auf Kernkompetenzen konzentrieren zu wollen, in den Vordergrund gestellt.[29] In Beispiel 1 wurde sichtbar, dass die Strategie beziehungsweise die Definition, was Kernbereich ist und was nicht, stark durch die taktischen Empfehlungen sowie durch Entwicklungseinschätzungen der entsprechenden Fachabteilung beeinflusst wurde. In der Softwareentwicklung spielen in diesem Prozess wieder der im Abschnitt „Entscheidungsprozess, Entscheidungsträger und Betroffene“ beschriebene Interessenkonflikt mit hinein. Im Beispiel 1 zeigt sich außerdem, dass die Festlegung auf den Bezug und somit die Festlegung des Kernkompetenzbereiches mittelfristig ist, da die Festlegungszeit der Fahrzeugmodelllaufzeit entspricht. Im Vergleich zur Zuliefererbranche ist in der Softwareentwicklung jedoch eine erheblich schnellere Technologie- und Marktentwicklung zu beobachten.[30] Hierdurch ist eine Einschätzung, ob die Softwareentwicklung in den Bereich der Kernkompetenzen gehört oder in Zukunft gehören wird, erheblich schwerer und erfordert eine Überprüfung in kürzeren Zyklen. Dies widerspricht jedoch der Definition von Outsourcing als eine Form der langfristigen

28 In Anlehnung an Ludewig (2003), S. „4-4“ und an Balzert (1998), S. 288-289
[29] vgl. Münzel, Neesen (1997), S. 268
[30] vgl. Balzert (2000), S. 29-30 und vgl. Kemper (2001)

Externalisierung. Festzuhalten ist außerdem, dass Outsourcing eine permanente Herausforderung und ein zusätzliches ständig offenes Managementproblem darstellt.[31] Es muss zusätzlich zur weiterhin notwendigen Marktbeobachtung und dem dazugehörigen Technologiemanagement, die zur Überprüfung und Anpassung der Kernbereichsdefinition notwendig sind, der Outsourcing-Anbietermarkt ständig analysiert werden. Dies muss geschehen, damit reagiert werden kann, wenn der bisherige Outsourcinganbieter beispielsweise nicht in ausreichender Weise neue Technologien verwendet.

Auch sollte aus strategischer Sicht die Entscheidungsgrundlage beziehungsweise die Herangehensweise an die Beurteilung von Outsourcing-Möglichkeiten nicht anhand momentaner Kosteneinsparungsmöglichkeiten, sondern anhand inherentem Potential geschehen. Outsourcing ist unter Umständen billiger, jedoch kann durch die Eigenerstellung zusätzlicher Wissensaufbau im Kernbereich vorangetrieben werden. Dieses zusätzliche erworbene Know-How bildet oftmals, insbesondere im Bereich der Softwareentwicklung, die Grundlagen für spätere Konkurrenzfähigkeit, da zukünftig besser und schneller als mit dem „Standard“ vom Outsourcinganbieter reagiert werden kann. Dies ist in besonderem Maße wichtig, da durch den stark ansteigenden Wertanteil der Software an den Gesamtkosten die Software zur Differenzierung von der Konkurrenz von immer größerer Bedeutung ist.[32] Insbesondere im Bereich der Embedded-Software, der ca. 80% aller Software ausmacht, hat dies seine Gültigkeit. Aber auch bei Software, die unterstützend oder ermöglichend auf die Produkterstellung einwirkt, ist dieser Sachverhalt von Bedeutung. Im Beispiel 2 wurde die Problematik, dass zur Eigenentwicklung nicht ausreichend Know-How vorhanden ist, nicht gelöst, obwohl dies zu einem großen Teil in den Bereich der Kernkompetenzen gehört. Eine weitere Differenzierung von Konkurrenten wird so nicht gefördert.

7 Risiko, Abhängigkeit und Flexibilität

Oft wird als Vorteil von Outsourcing die Abwälzung von Risiko an den Outsourcing-Anbieter genannt. In Beispiel 1 wird deutlich, dass das Produktionsrisiko nicht mehr beim Fahrzeughersteller sondern beim Zulieferer liegt. Das ist möglich, da die Qualität der Getriebe und die Übereinstimmung mit den spezifizierten Anforderungen geprüft werden kann. Dadurch liegen die Kosten, die durch Produktionsfehler entstehen können und somit auch das Risiko, beim Zulieferer. Auch das Investitionsrisiko und das gebundene Kapital können auf Seiten des Fahrzeugherstellers erheblich minimiert werden, da nicht in Maschinen investiert werden muss und nach Bedarf „just in time“ geliefert werden kann. In der Softwareentwicklung ist eine entsprechende Kapitalfreisetzung oftmals nur in sehr

[31] vgl. Bruch (1998), S. 24
[32] vgl. Balzert (2000), S. 28-29

geringem Maße möglich. Wie in Beispiel 2 ersichtlich, entsteht keine nennenswerte Kapitalfreisetzung. Auch eine Senkung von Investitionskosten entsteht durch das Outsourcing nur in geringem Umfang. Der Unterschied liegt lediglich in der Anzahl der fest eingestellten Entwickler. Eine Risikoabwälzung findet in Beispiel 2 überhaupt nicht statt. Das liegt darin begründet, dass Software nicht testbar ist. Fehler in Software können sprunghaft, sogar nach Jahren des fehlerfreien Betriebes auftreten. Würde im Beispiel 2 ein Fehler in unregelmäßigen Abständen in der Rentenberechnung auftreten, so würde dies einen erheblichen Imageschaden und Schadenersatzforderungen nach sich ziehen, die den Konkurs für das Unternehmen bedeuten könnten. Dieses Risiko trägt nicht der Outsourcer, sondern das Finanzdienstleistungsunternehmen selbst, da im Zweifel Schadenersatzforderungen an den Outsourcing-Anbieter anhand unzureichend festgeschriebener Anforderungen scheitern. Dies muss nicht notgedrungen durch Fehler des Auftraggebers bedingt sein, sondern liegt meist einfach daran, das Software nicht spezifizierbar ist.

In Beispiel 2 wird außerdem deutlich, dass in der Softwareentwicklung oft eine erhebliche Menge fachspezifischen Know-How an den Outsourcing-Anbieter weitergegeben werden muss. Dies ist erforderlich, damit es dem Outsourcing-Anbieter überhaupt möglich ist, eine entsprechende Software zu erstellen. Hierin liegt ein weiteres Risiko. In Beispiel 2 handelt es sich um Kernkompetenzen, welches das Unternehmen von anderen unterscheidet. In der Praxis ist nicht kontrollierbar, ob der Outsourcing-Anbieter dieses Know-How auch für die Erstellung weiterer Softwaresysteme verwendet und das Know-How somit Konkurrenten zugänglich wird.

Das Argument, mehr Flexibilität durch ein schlankeres Unternehmen zu erlangen, welches sowie bei „Make or Buy“ als auch bei Outsourcing oft genannt wird, trifft im Beispiel 1 sicherlich zu. In Beispiel 2 ist dies fraglich, da durch die Software eine langfristige Bindung an den Outsourcing-Anbieter geschieht, da nur dieser wirklich über Know-How über die Software verfügt. Dem kann sicherlich entgegengewirkt werden, indem entsprechende Mechanismen vereinbart und umgesetzt werden, die das Software-Know-How dem Finanzdienstleister zuführt. Hierfür muss aber erheblich größerer Aufwand betrieben werden, als dies beispielsweise bei der obligatorischen Erstellung der Dokumentation geschieht, welche in der Praxis meist auch schon unzureichend ausfällt. Außerdem stellt eine entsprechende Forderung einen gewaltigen Interessenskonflikt zwischen Outsourcing-Anbieter und dem -Kunden dar, da der Anbieter Interesse an einer langfristigen Bindung hat. So wird oft von Outsourcing-Anbietern gezielt die Strategie verfolgt, Kunden über geringe Entwicklungskosten zu locken. Der Outsourcing-Anbieter verdient dann erst durch die langfristige Bindung in der Phase „Betrieb, Wartung“, die jedoch den größten Aufwand im Softwarelebenszyklus trägt (siehe Abschnitt „Softwareentwicklung“).

8 Fazit

Die Frage, ob es sinnvoll ist Outsourcing in der Softwareentwicklung zu betreiben, kann nicht allgemein gültig beantwortet werden. Wie sinnig Outsourcing ist, hängt davon ab, ob die Kernkompetenzen tangiert werden oder Kernkompetenzen zur Erstellung der Software notwendig sind. Bei Banken und Sparkassen wird beispielsweise Outsourcing sehr differenziert betrachtet, da insbesondere im Bereich der Anwendungsentwicklung erhebliche Überschneidungen mit den Kernkompetenzen auftreten und dadurch Branchenwissen im engen Zusammenhang mit der Bank-Informatik stehen.[33] Weiter ist festzuhalten, dass es erhebliche Unterschiede zum „Make or Buy“ im Fertigungssektor gibt, die beachtet werden müssen. So ist die Orientierung der Kosten nicht an Stückzahlen oder den Entwicklungskosten zu bemessen, sondern an den möglichen Fehlerfolgekosten. In welchen Softwareentwicklungsphasen Outsourcing zu empfehlen ist, kann ebenfalls nicht allgemein gesagt werden. Unter anderem hängt das vom verwendeten Software-Prozess-Modell ab. Klar ist jedoch, dass das Risiko in späten Phasen geringer wird, weil Fehler eine geringere Latenzzeit zwischen Entstehung und Entdeckung aufweisen. Der große Anteil an soften Anforderungen und die daraus folgende geringe Spezifizierbarkeit ist einer der gewichtigsten Unterschiede zur Fertigung. Auch die meist nicht vorhandene Trennung zwischen Outsourcing-Entscheidern und -Betroffenen ist als besondere Herausforderung in der Softwareentwicklung anzusehen. Diese betrifft nicht nur jede einzelne taktische Handlungsempfehlung, sondern auch das Bilden der Strategie und das Festlegen und Kontrollieren der Kernbereichszugehörigkeit. Zu diesen Managementaufgaben kommen bei Outsourcing in der Softwareentwicklung in besonderem Maße das Outsourcing-Management und die Beobachtung des Outsourcing-Marktes hinzu. So muss nicht zuletzt wegen der großen Zahl von Fachgebieten für jede Software im Einzelnen entschieden werden, ob Outsourcing sinnvoll ist und welcher Anbieter der Richtige ist. Daher wird eine Strategie, die Outsourcing für die Softwareentwicklung allgemein vorsieht, der Komplexität dieses Bereiches sicher nicht gerecht.

9 Lösungsansatz (These)

Meiner Meinung nach können die eigentlichen Probleme im Bereich der Softwareentwicklung nicht durch Outsourcing gelöst werden. Outsourcing selbst wird in diesem Bereich nicht mehr Kostentransparenz bewirken. Auch werden meiner Meinung nach, wenn der Know-How-Verlust und die nicht ausgenutzten inherenten Vorteile mit in die Kostenrechnugen einbezogen werden, in den meisten Fällen keine Einsparungen gemacht.

[33] vgl. Münzel, Neesen (1997), S. 269

Außerdem sind für differenzierte und fundierte Sourcing-Entscheidungen meiner Meinung nach nicht die organisatorischen Voraussetzungen gegeben.

In den meisten Fällen sind „Make and Buy“-Strategien[34] und ein Low-Level-Outsourcing Erfolg versprechender. Hierdurch könnte das Risiko geringer gehalten werden, weil kleine Module zu einem erheblich größeren Anteil spezifizierbar sind. Auch könnte damit erreicht werden, dass in geringerer Menge Know-How an die Outsourcinganbieter abfließt und mehr Know-How im eigenen Unternehmen gebildet wird. Im Low-Level-Outsourcing würden also nach Bedarf einzelne Spezialisten, beispielsweise Datenbankspezialisten, „ausgeliehen“ oder die Erbringung kleiner abgesteckter Aufgaben nach außen gegeben.

Um eine solche Strategie verfolgen zu können, müsste allerdings eine Zweiteilung der Softwareentwicklung in einen Softwareentwicklungsteil und einen Softwareentwicklungsmanagementteil vollzogen werden. Hierdurch würde der Interessenskonflikt (siehe Abschnitt Entscheidungsprozess, Entscheidungsträger und Betroffene) entschärft, was zu einer erheblichen Professionalisierung in der Softwareentwicklung im Technologiemanagement, im Softwaremanagement und der Markteinschätzung führen würde. Insbesondere wird die Professionalisierung durch objektive Beurteilung von Outsourcing-Möglichkeiten und differenzierte auf fachlichen und strategischen Überlegungen basierende Sourcing-Entscheidungen möglich. Auch für ein geregeltes Wissensmanagement im Bereich Sourcing wäre die Zweiteilung von Vorteil. Auf diese Weise würden eine erhöhte Kostentransparenz und mehr Flexibilität entstehen. Auch könnte dann eine stärkere Prozessorientierung in der Softwareentwicklung erreicht werden, beispielsweise wenn das Personalmanagement für die einzelnen Entwicklungsprojekte ebenfalls vom Softwareentwicklungsmanagement übernommen wird. Hierdurch würde auch der Vergleich zwischen Low-Level-Outsourcing und dem Einstellen von Mitarbeitern unter dem Gesichtspunkten „Know-How“ und „Bedarfsdauer“ einfacher.

[34] vgl. Münzel, Neesen (1997), S. 275

Quellenverzeichnis

Balzert H. (1998) *Lehrbuch der Software-Technik : Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung* - Berlin (Lehrbücher der Informatik) : Spektrum Akademischer Verlag Heidelberg

Balzert H. (2000) *Lehrbuch der Software-Technik : Software-Entwicklung* - [2. Aufl. (1. Nachdruck 2001)] - Berlin (Lehrbücher der Informatik) : Spektrum Akademischer Verlag Heidelberg

Dobschütz, L. *CoSourcing und IV-Controlling : Werte schaffen statt Sparen um jeden Preis!* - in Berg J., Horst G (1995) *Outsourcing in der Informationstechnologie* - Hrsg. Frankfurt / Main , New York : Campus-Verlag

Bruch H. (1998) *Outsourcing : Konzepte und Strategien, Chancen und Risiken* - Wiesbaden : Gabler

Kemper H. (2001) Folien zur Vorlesung *Informations- und Kommunikationssysteme I* – Stuttgart : Universität

Ludewig J. (2003) Vorläufiges Skript zur Vorlesung *Grundlagen des Software Engineering* – Stuttgart : Universität

Judewig J. (2003b) Folienergänzung zu Ludewig J. (2003)

Mertens P., Knolmayer G. (1995) *Organisation der Informationsverarbeitung : Grundlagen, Aufbau, Arbeitsteilung* - [2., vollst. überarb. Aufl.] - Wiesbaden : Gabler

Münzl G., Neesen H. (1997) *Innovative Informationstechnologie und Strategie des Teil-Outsourcing*
in (1997) *Outsourcing in Banken und Sparkassen : Strategien, Rechtsgrundlagen, Praxisbeispiele* - Stuttgart : Dt. Sparkassenverlag

Steppan B. (2004) *Offshoring verdeckt Management-Probleme* - Computerwoche online - (Stand: 10.09.04)
http://www.computerwoche.de/index.cfm?pageid=256&artid=55553&type=detail&kw=outsourcing%20softwareentwicklung%20implementierung

Wells D. (2001) *Extreme Programming (XP)* - (Stand: 10.09.2004)
http://www.extremeprogramming.org/