Teil B: Untersuchungsgegenstand und Untersuchungsmethode

1. Der Untersuchungsgegenstand: Internationale Softwareentwicklungskooperationen

1.1. Definition von Software

Eine einfache Definition von Software besagt, dass es sich dabei um ein Gefüge von Anweisungen, Prozeduren, Programmen, Regeln und Dokumentationen handelt, die auf unterschiedlichen physischen Trägern wie Bändern, Disketten, oder Stromkreisläufen gespeichert werden können und die Nutzung von elektronischen Geräten der Datenverarbeitung ermöglichen (OECD, 1985). In einem abstrakteren Sinne kann Software auch als programmiertes Problemlösungsverfahren definiert werden (Weber, 1992).

Software tritt sowohl als Programm zur Steuerung des Ablaufverhaltens von elektronischen Datenverarbeitungsanlagen, als auch eingebettet in elektronischen Bauelementen auf. Die elektronischen Bauteile wie Halbleiter und Mikrochips, werden der Hardware zugerechnet. Das Zusammenwirken der Bauteile wird durch Softwareprogramme gesteuert. Mit der technologischen Weiterentwicklung sind die Steuerprogramme mehr und mehr in die elektronischen Bauelemente selbst hinein verlagert worden. Neuere Bauelemente beinhalten durchaus hoch komplexe Programme. Dadurch wird die Abgrenzung von Hardware und Software zunehmend schwieriger.

1.2. Besonderheiten der Softwareproduktion

Die Produktion von Software weist auf den ersten Blick wenig Gemeinsamkeiten mit der industriellen Produktion materieller Güter auf. Die Herstellung von Software beinhaltet die Kodifizierung von Wissen und Informationen, so dass sowohl der Input als auch der Output dieses Produktionsprozesses immaterieller Natur sind. Das für die Entwicklung eine Software erstellte Planungskonzept kann schon fast als das Produkt selbst angesehen werden. Der Begriff der wissensintensiven Unternehmung (Starbuck, 1992) erscheint hier deshalb als angemessen.

Die Softwareentwicklung hat als wissensintensive Tätigkeit Gemeinsamkeiten mit anderen Bereichen industrieller Ingenieurstätigkeit. Vergleichbar ist die Softwareentwicklung

mit Design- und Entwurfstätigkeiten, wie sie in den traditionelleren Technologiebereichen und auch in den neueren biologischen und pharmazeutischen Disziplinen üblich sind.

Der Fertigungsprozess, der ein zentraler Kostenfaktor eines großen Teils herkömmlicher Industrien ist, entfällt bei der Softwareproduktion. Die Vervielfältigung eines Softwareprogramms erfolgt durch das Kopieren auf die unterschiedlichsten Datenträger, wobei nahezu keine Produktionskosten anfallen. Die Herstellungskosten von Software bestehen zum überwiegenden Teil aus Entwicklungskosten.[32]

Die Produktion von Software ist in der Regel eine innovative Tätigkeit, die es Anwendern ermöglicht, bestimmte Aufgaben und Funktionen auf eine effizientere Art und Weise durchzuführen. Dabei variiert der jeweilige Neuheitsgrad eines Softwareproduktes. Häufig werden an bereits existierenden Pogrammen nur kleine Veränderungen und Weiterentwicklungen durchgeführt oder kundenspezifische Software wird durch den Hersteller gewartet und an neue Anforderungen angepasst. Solche Routineentwicklungen, werden in der Regel durch neue Nutzeranforderungen oder durch Änderungen bei der Hardwaretechnologie initiiert. In diesen Fällen wird von evolutionären Anpassungsprozessen existierender Softwaresysteme an Umweltveränderungen gesprochen (Torrisi, 1998). Einen höheren Innovationsgehalt hat dagegen Software, die für die spezifischen Anforderungen bestimmter Kunden komplett neu entwickelt wird.

1.3. Der Softwareentwicklungsprozess

Seit den sechziger Jahren gibt es Versuche, den Softwareentwicklungsprozess mit den Hauptphasen Spezifikation, Design, Codierung und Tests, zu definieren und allgemeingültige Prozessmodelle zu formulieren. Dabei hat sich das sogenannte „Waterfall Model“ als das Standardmodell etabliert (Torrisi, 1998). Im „Wasserfallmodell“ besteht die erste Phase des Softwareentwicklungsprozesses in der Problem- bzw. Systemanalyse, also der Analyse der Anforderungen, die durch den zukünftigen Nutzer an ein System gestellt werden. Nachdem das Gesamtsystem analysiert wurde, erfolgt das Systemdesign, das dann in immer kleinere Einheiten für das Detaildesign aufgeteilt wird. Als dritte Phase erfolgt die Programmierung, in der die Designspezifikationen in einer Computersprache dargestellt werden. Für ein neues Programm entsteht so der „source code’“. Der „source code“, der in einer Computersprache wie Java oder C++ verfasst ist, wird dann in der Regel durch automatische Übersetzungsprogramme in eine Maschinensprache, einen binären Code übersetzt, in dem

[32] In Softwarefirmen wie Microsoft, Oracle oder Sybase ist auch der Anteil der Vertriebs- und Marketingkosten am Gesamtumsatz (ca.30-50%) relativ hoch. Mit der Nutzung des Internets als Vertriebskanal für Software können aber auch hier die Kosten merklich gesenkt werden (Tucker, 1997).

Einsen und Nullen die Anweisungen für die Hardware darstellen. Danach erfolgt die Systemintegration, bei der die Kompatibilität der einzelnen Komponenten und die Leistungsfähigkeit des Gesamtsystems geprüft wird. Der letzte Schritt ist die Installation des Systems beim Kunden und die Einweisung der Nutzer. Alle Zwischenschritte der Entwicklung werden Testverfahren unterzogen. Die Entwickler müssen immer wieder die Spezifikationsanalyse, das Programmdesign und die Programmierung hinsichtlich der Testergebnisse verbessern und miteinander in Einklang bringen, was ein wiederholtes Durchlaufen der Phasen erforderlich macht (Cusumano, 1992).

Obwohl Vorgehensmodelle in der Softwareentwicklung von Unternehmung zu Unternehmung variieren, sind sie doch zumeist auf Standardansätze, wie das genannte Wasserfallmodell, zurückzuführen (Boehm, 1976; Royce, 1970). In jüngerer Zeit ist viel von sogenannten ‚Rapid Development' Modellen, die vor allem bei der Entwicklung von Internetanwendungen eingesetzt werden, sowie von Objektorientierung die Rede. Dieser Trend, spiegelt die Notwendigkeit wider, Aufgaben der Softwareentwicklung in Teilprojekte zu gliedern, um Projektpläne zuverlässiger zu machen (GfK, 2000).

Die nationalen und internationalen Prozessstandards wie CMM[33], ISO oder DIN besitzen in Deutschland, nach den Erkenntnissen der GfK-Studie (2000), noch wenig Relevanz für die Hersteller. In Sekundär- und Primärbranche sind es weniger als die Hälfte der Firmen, die unternehmenseigene oder branchenübliche Vorgehensmodelle einsetzen. Bei kleineren Unternehmen, wie sie insbesondere während des New Economy Booms entstanden sind, gibt es kaum standardisierte Vorgehensmodelle. Es wird hier meist in kleinen und überschaubaren Teams gearbeitet, die unsystematischere Herangehensweisen zulassen. Die Koordinationsprobleme nehmen aber mit der Größe von Projekten und Systemen zu und bereiten selbst großen und etablierten Softwarefirmen Schwierigkeiten. Der Softwareindustrie hängt der Ruf an, sich seit ihrer Entstehung vor gut vierzig Jahren, in einer permanenten Zuverlässigkeitskrise zu befinden (Kraut/Streeter, 1995). Nicht eingehaltene Terminzusagen und Qualitätsmängel bei prominenten Fällen in Deutschland wie Toll Collect oder der Online Arbeitsbörse der Agentur für Arbeit, tragen auch in jüngster Zeit nicht dazu bei, dieses Bild zu korrigieren.

Obwohl lineare Entwicklungsansätze, wie das Wasserfallmodell, mit einer klaren Abfolge von Aktivitäten, die Planbarkeit von Softwareentwicklungsprojekten suggeriert, ist Unsicherheit hinsichtlich der benötigten Entwicklungszeit und der Qualität des Endproduktes

[33]Der CMM-Standard (Capabilities Maturity Model), der in den USA zur Beurteilung von Softwareprozessen zunächst für Auftragnehmer des DoD entwickelt wurde, ist mittlerweile gerade für indische Softwarefirmen ein wichtiges Aushängeschild ihrer hohen Prozessstandards geworden.

die Regel. Mehrere Gründe sind dafür ausschlaggebend. Die Größe von Softwaresystemen übersteigt oft das Verständnisvermögen von einzelnen Entwicklern und von Entwicklungsteams. Wenn die Systemgröße Millionen von Codezeilen beträgt, wie dies beispielsweise bei der Software von Telefonvermittlungsanlagen der Fall ist, wird die Koordination der Systementwicklung zu einer erheblichen Herausforderung.[34] Die mit der Systemgröße zunehmende Spezialisierung erschwert die Koordination, denn in der Regel entstehen dabei auch soziale, organisationale und geographische Barrieren, die die Kommunikation und den Informationsaustausch zwischen den beteiligten Personen behindern. Die Gefahr von Fehlentwicklungen durch mangelnden Wissensaustausch steigt daher mit zunehmender Projektgröße (Kraut/Streeter, 1995).

Neben dem Projektumfang ist es auch die Unsicherheit bezüglich der anfallenden Entwicklungsaufgaben und des Entwicklungsaufwandes, die die Planung und Koordination von Softwareprojekten schwierig macht. Dies betrifft in erster Linie Systeme, die neu entwickelt werden. Veränderungen von Umweltbedingungen und von Kundenwünschen treten oft nach Abschluss der Planung auf und müssen berücksichtigt werden. Darüber hinaus ist es eher die Regel, dass aufgrund mangelnder Informationen und mangelndem Wissen über eine spezifische Anwendungsdomäne auf Seiten der Entwickler, die Spezifikationen eines Softwaresystems zu Beginn der Entwicklung unvollständig sind. Auch die unterschiedlichen Perspektiven von Entwicklern und Nutzern bieten Raum für Fehlinterpretationen und Auslegungsdifferenzen. Schließlich bestehen Softwaresysteme oft aus einer Vielzahl von Komponenten und Modulen, deren Funktionalitäten genau aufeinander abgestimmt sein müssen. Die Koordination und Integration der unterschiedlichen Komponenten ist eine sehr komplexe Aufgabe und Fehlentwicklungen werden oftmals erst nach der Zusammenstellung des Gesamtsystems bemerkt (Kraut/Steeter, 1995). Generell nimmt mit der Größe und Komplexität eines Projektes die Unsicherheit über die Gesamtkosten, die Lieferfrist und die Leistungsmerkmale eines Endproduktes zu.

Trotz der dargestellten Schwierigkeiten wurde die Produktivität der Softwareherstellung, auch durch die Einführung neuer technologischer Ansätze, seit den achtziger Jahren verbessert. Methoden wie das Computer-aided Software Engineering und objektorientierte Entwicklungsansätze haben die Produktivität von Entwicklungsteams gesteigert.

[34] Die Software der Bodenkontrolle für die Apollo Raumfähre umfasste in den siebziger Jahren 23 Millionen Zeilen an Softwarecode. Die Software von den zur selben Zeit entwickelten Telefonvermittlungsstellen enthielt schon 10 Millionen ‚lines of code' (Fox, 1982).

1.4. Spezifisches Wissen in der Softwareentwicklung

Für die Entwicklung von Software ist spezifisches Wissen in verschiedenen Bereichen notwendig. Der wichtigste Bereich, da er als ein Alleinstellungsmerkmal von Softwareherstellern gilt, ist das Anwendungswissen (application knowledge). Mit Anwendungswissen wird das Wissen bezeichnet, das für die Entwicklung von Software in einen bestimmten Anwendergebiet wie Telekommunikation, Medizintechnik oder Maschinenbau notwendig ist. Dieses Wissen kann in Form von bestimmten Algorithmen in Programmen enthalten sein, es umfasst aber vor allem auch das Wissen der Ingenieure und Entwickler, die sich auf ein Anwendungsgebiet spezialisiert haben.

Des weiteren gibt es das Wissen über Qualitätsstandards und Qualitätsmanagement, das vor allem den Entwicklungsprozess betrifft. Die schon angesprochenen Industriestandards wie ISO 9000 und CMM gehören zu diesem Bereich des Wissens, der durch überbetriebliche Konventionen festgelegt wird. Es gibt aber auch unternehmensspezifische Qualitätsprozesse und Techniken. Das gleiche gilt für den Entwicklungsprozess selbst, für den internationale (Quasi-) Standards existieren, die in großen Teilen dokumentiert sind (Kobitzsch et al. 2001).

1.5. Firmengröße und Organisation

Die Größe der Firmen ist ein wichtiger Einflussfaktor auf die Gestaltung der Entwicklungsorganisation. Torrisi (1998) unterscheidet drei Typen von Firmen, die sich in der Organisation ihrer Softwareherstellung unterscheiden. Einmal große, etablierte Hersteller von spezialisierter Software oder auch integrierte Software-Hardware Produzenten. Zweitens kleinere etablierte Firmen, die in Marktnischen tätig sind und drittens kleinere Start-up Firmen. Die großen Firmen nutzen formalisierte Entwicklungsmethoden und in der Regel auch neueste Entwicklungssprachen und Werkzeuge. Häufig sind die Entwicklungsaktivitäten um Produkte und Märkte organisiert. Die Zielsetzung, die lange Zeit üblichen ad hoc Strukturen der Softwareentwicklung durch formalisierte Prozesse zu ersetzen, und die Produktivität durch projektübergreifend anwendbare Entwicklungswerkzeuge zu steigern, ist in diesen Firmen am ehesten verwirklicht.

Die kleineren etablierten Unternehmungen haben nach den Untersuchungsergebnissen von Torrisi (1998) andere Organisationslösungen. Ihre Softwareentwicklung ist in Projekten organisiert, wobei sie kleine Teams und informelle Kommunikationswege nutzen, aber kaum über strukturierte Entwicklungsmethoden verfügen. Das Hauptgeschäft besteht häufig in Veränderungen und Verbesserungen von Produkten, die sie in einer Marktnische erfolgreich verkaufen können oder aber in der Entwicklung von Subsystemen und Komponenten auf der Basis von Zulieferbeziehungen. Damit ähneln sie Start-up-Firmen, die in der Regel auf nur

ein oder einige wenige Produkte spezialisiert sind und kaum Gebrauch von strukturierten Entwicklungsmethoden machen. Oftmals besteht die Entwicklungsorganisation aus kleinen Teams, deren Mitglieder funktionsübergreifend arbeiten. Die Kommunikation ist direkt und informell und die Zeitpläne sind wenig strikt (Torrisi, 1998).

1.6. Die Software Factory als Modell für Softwareoutsourcing

Nach dem Urteil von Cusumano (1992) ist die Softwareproduktion ein Beispiel dafür, dass auch entwerfende Tätigkeiten wenigstens teilweise dem Effizienzregime der Produktion unterworfen werden können. Das Konzept der Software Factory ist ein Teil der Entwicklung zu immer weitergehender Arbeitsteilung und zu Outsourcing in der Softwareindustrie. Zunächst war es eine Reaktion auf die vielen Unsicherheitsfaktoren der Softwareentwicklung und die unbefriedigende Produktivität kleinteiliger Projektstrukturen. Angesichts der Komplexität und der schlechten Standardisierbarkeit der Entwicklungstätigkeiten war es unter Softwareproduzenten üblich, jedes Projekt ‚from the scratch' zu planen und durchzuführen. Für jedes Problem wurde eine neue Lösung entwickelt. Probleme traten häufig dann auf, wenn nicht mehr nur kundenspezifische Projekte, sondern Standardlösungen für einen breiteren Markt entwickelt werden sollten. Seit Ende der achtziger Jahre wurde in den USA die reine Projektorientierung der Softwarebranche von Praktikern und Wissenschaftlern zunehmend kritisiert, vor allem unter dem Hinweis auf mangelnde Produktivitätssteigerung und auf die Inflexibilität und Überalterung von Softwaresystemen, bei denen kleine Änderungen zu schwerwiegenden Fehlfunktionen führen können, weil die Entstehung des Gesamtsystem nicht mehr nachvollziehbar ist (Weber, 1992). Auch die Möglichkeiten der Wiederverwendbarkeit von Systemkomponenten und der Standardisierung und Automatisierung von Entwicklungsschritten, sei zu wenig genutzt worden. Aus diesen Überlegungen ist das Konzept der Softwarefabrik entstanden, das unter anderem vorsieht, dass gemeinsame Designelemente, Komponenten oder Entwicklungsmethoden in unterschiedlichen Projekten genutzt werden (Cusumano, 1992).

Ein weiterer Grund, für die Entstehung und Verbreitung des Fabrikkonzeptes war der Mangel an qualifiziertem Entwicklungspersonal, der teilweise bereits in den sechziger Jahren auftrat. 1985 wurde in Japan ein Defizit von 17 000 Softwareingenieuren und 26 000 Programmierern festgestellt (Dambrot, 1989). Japan war auch das Land, in dem die Organisationen der großen integrierten Hardware-Softwarehersteller wie Hitachi, Fujitsu oder Toshiba als erste dem Konzept der Software Factory folgten.

Torrisi (1998) hat einige übereinstimmende Merkmale von japanischen Software Factories identifiziert. Dazu gehören ein integrierter Bestand an Entwicklungswerkzeugen,

standardisierte Prozesse und Managementpraktiken sowie eine Matrixorganisation. In einigen Fällen existiert eine Arbeitsteilung zwischen Firmen, die grundlegende Design- und Projektmanagementaufgaben durchführen, und Tochterfirmen oder Subunternehmern, die die Programmierung übernehmen. Der Vorteil liege in der Spezialisierung und einfacheren Qualitätskontrolle. Während bei dieser Arbeitsteilung die großen Firmen Effizienz- und Flexibilitätsvorteile realisieren können, arbeiten die kleinen Firmen weit weniger effizient und haben aufgrund der weniger qualifizierten Programmiertätigkeiten auch weniger Innovationspotentiale. Eine vergleichbar gestufte Arbeitsteilung wie in Japan wurde in europäischen und amerikanischen Firmen nicht beobachtet. Selbst in Japan scheint der Trend in jüngerer Zeit dahin zu gehen, dass es auch Programmierfirmen ermöglicht wird, Kompetenzen im Softwaredesign aufbauen zu können (Torrisi, 1998).

Das Modell der Arbeitsteilung zwischen qualifizierter Tätigkeit von Systemanalysten und wenig qualifizierter Programmiertätigkeit in Zulieferbetrieben, galt in den achtziger Jahren noch als eine Besonderheit des japanischen Softwaresektors. In den neunziger Jahren begannen aber immer mehr amerikanische Softwarefirmen damit, Programmiertätigkeiten an Firmen in Asien, vor allem China und Indien, zu vergeben.

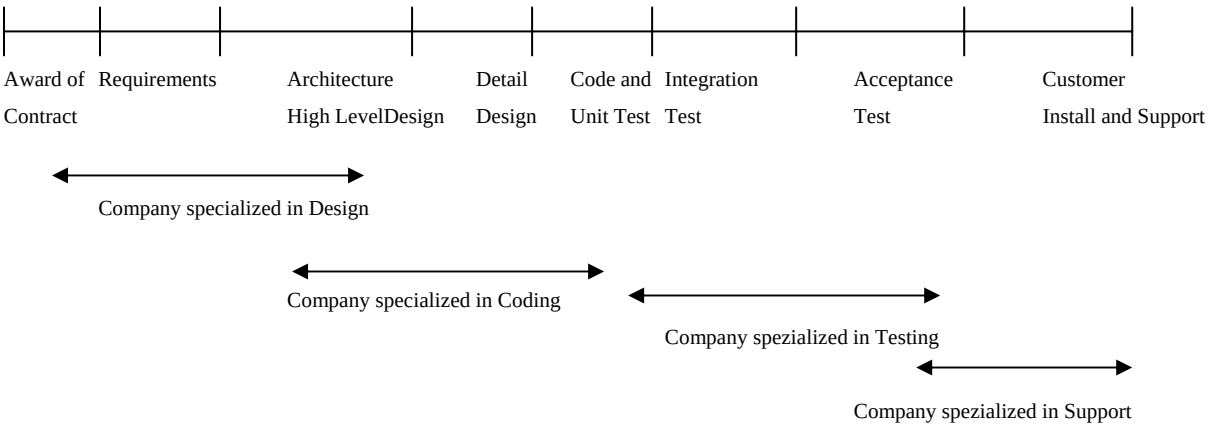
1.7. Gestaltung der Internationalisierung des Entwicklungsprozesses

Die genannten Merkmale von Softwarefirmen wie die Firmengröße, die Standardisierung von Entwicklungsprozessen, die jeweils angewandten Methoden und insbesondere der Grad der Arbeitsteilung und das notwendige Anwenderwissen haben Einfluss auf die Durchführbarkeit von internationalen Kooperationen und Outsourcing.

In der Ratgeberliteratur mangelt es nicht an Vorschlägen zur Gestaltung internationaler Softwarekooperationen. Dabei steht eindeutig die „Task Integration“, also die Aufgabenintegration im Vordergrund. Bei der Verlagerung von Entwicklungsschritten sollen die Firmen die Komplexität des Entwicklungsprozesses und die Interdependenzen der Projektakteure beachten. Die Entscheidung, welche Phasen und Softwaresysteme verlagert werden, kann nach unterschiedlichen Kriterien erfolgen. Die am häufigsten empfohlene und angewandte Methode, ist die Verteilung nach Maßgabe der Softwarearchitektur. Koordinationsprobleme werden verringert, indem die Interaktionsnotwendigkeit zwischen den Standorten niedrig gehalten wird (Carmel/Agarwal, 2001). Dabei kann jedem Standort die Entwicklung eines abgrenzbaren Subsystems zugewiesen werden. Nach diesem modularen Prinzip entspricht das Organisationssystem der Struktur des Softwaresystems (Parnas, 1972). Die internationale Arbeitsteilung wird somit zu einem Abbild der Systemarchitektur.

Ähnlich argumentieren Mockus und Weiss (2001), die empfehlen, dass eng gekoppelte Arbeitsschritte, die eine häufige und genaue Abstimmung erfordern, am gleichen Standort und innerhalb einer Gruppe durchgeführt werden sollen. Sie entwickeln auf Basis dieser Empfehlung auch ein Analysesystem (ebd., S.33f.), mit dessen Hilfe Arbeitsschritte identifiziert und in eine möglichst optimale Architektur von modular angeordneten Arbeitsgruppen gebracht werden sollen. Optimal sei die Arbeitsteilung in einer globalen Softwareentwicklungsorganisation dann, wenn die Koordinations- und Kommunikationserfordernisse zwischen den beteiligten Standorten minimiert werden konnten. Die Trennung zwischen Hauptsystem und Subsystemen setzt allerdings voraus, dass sich Schnittstellen im Vorfeld klar definieren lassen. Gerade die Definition der Schnittstellen der Subsysteme kann aber durchaus Schwierigkeiten bereiten (Karolak, 1998, S.45). Auch die Vorteile globaler und virtueller Entwicklungsteams, die unter Ausnutzung der Zeitzonenunterschiede rund um die Uhr an der gleichen Software arbeiten, können bei einer modularen Arbeitsteilung natürlich nicht realisiert werden (Carmel/Agarwal, 2001).

Eine andere Möglichkeit der Verlagerung besteht darin, die unterschiedlichen Prozessphasen wie Design, Codierung, Tests auf verschiedene Standorte zu verteilen, wobei Spezialisierungsvorteile ausgenutzt werden können:



[Abb. nach Karolak, 1998, S.40]

Der Nachteil dieses Vorgehens ist, dass der Auftraggeber abhängig von den ausgelagerten Phasen ist, aber keinen direkten Einfluss hat und auch Prozessbeschreibungen und Dokumentationen möglicherweise nicht vollständig zugänglich oder verständlich sind (ebd.).

Neben diesen eher strukturellen Überlegungen, sind es aber oft die jeweils verfügbaren Personalressourcen, die bei Terminknappheit den Ausschlag für die Verteilung von Entwicklungsarbeit geben. Ebenso kann die Ausstattung mit Softwaretools von Bedeutung für die Entscheidung sein. Deren Vorhandensein ist allerdings noch keine Garantie für eine gute Durchführung der Entwicklungsarbeit (ebd.). Schließlich sind die Schwierigkeiten der Integration unterschiedlicher Komponenten und Subsysteme, die in „normalen“, ko-lozierten Projekten schon groß sein können, in verteilten Projekten nochmals erheblich größer (Karolak, 1998, S.97).

1.8. Typologie internationaler Softwareprojekte unter Berücksichtigung der rechtlichen Verbundenheit

Im Hinblick auf die rechtliche Verbundenheit der kooperierenden Betriebe und Teams, lässt sich eine Typologie von vier Kooperationsformen bilden. Die internationale Dimension ist implizit berücksichtigt, indem davon ausgegangen wird, dass die Partnerfirmen jeweils unterschiedlicher Nationalität sind.

	Rechtlich unabhängig	Rechtlich verbunden
Getrennte Teams	Typ 1 (klassisches Outsourcing)	Typ 2 (Outsourcing an ausländische Tochterfirmen oder Joint Venture)
Ein Team	Typ 4 (Entwicklungsallianz mit einem virtuellen Team)	Typ 3 (Intra-organisationales Team an verschiedenen Standorten)

[Abb. nach Kobitsch et al., 2001.]

Typ 1 bezeichnet die Zusammenarbeit von getrennten Teams rechtlich unverbundener Firmen. Im Grunde handelt es sich dabei um eine Zulieferbeziehung. Rechtliche Fragen, Probleme des Wissenstransfers und des Projektmanagements sind zu erwarten, gerade wenn geographische Trennung und kulturelle Unterschiede bestehen. Typ 2 beschreibt die

Zusammenarbeit getrennter Teams in rechtlich verbundenen Firmen, also beispielsweise im Konzern. Die rechtlichen Fragen und die Fragen des Wissens- und Managementtransfers sind leichter zu lösen als in Typ 1, da eine klare hierarchische Beziehung zwischen Mutter- und Tochterunternehmen besteht. Typ 3 bezeichnet ein Team, das auf verschiedene Standorte rechtlich verbundener Firmen verteilt ist. Diese Konstellation wird häufig als virtuelles Team bezeichnet. Auch bei Typ 4 handelt es sich um ein virtuelles Team, das auf die Standorte rechtlich unabhängiger Firmen verteilt ist. Die Integrationsbedingungen sind hier, u.a. durch die rechtliche Selbstständigkeit der Partner, schwieriger als in Typ 3.

Nicht in die Typologie miteingeflossen sind die Fälle, in denen gemischte Teams an einem Standort zusammenarbeiten. Dazu gehört der im Softwarebereich öfters anzutreffende Fall der Personalüberlassung. Durch die Zunahme der Arbeitsmobilität selbstständiger Fachkräfte, sind kulturell gemischte Teams keine Seltenheit mehr. Es entstehen internationale Teams an einem Standort (co-located team), in denen die Teammitglieder aus unterschiedlichen Firmen und Nationen kommen. Aber auch umgekehrt werden Entwickler aus europäischen oder amerikanischen Firmen in die Partnerorganisationen oder Tochterfirmen geschickt, um vor Ort Managementaufgaben zu übernehmen.

Ferner lassen sich diese Grundformen der Verteilung internationaler Softwareentwicklung noch nach der Form des Markteintritts und der Beteiligungshöhe unterscheiden. So kann unterschieden werden, ob eine ausländische Tochterfirma neu aufgebaut wurde, also als ein sogenanntes ‚greenfield investment', oder ob eine andere Firma mit ihren Beschäftigten und der Infrastruktur durch eine Akquisition übernommen wurde. Eine Zwischenform wäre ein Joint Venture mit einem ausländischen Unternehmen, bei der je nach Beteiligungshöhe unterschiedliche Steuerungsmöglichkeiten für die beteiligten Firmen bestehen.

2. Methode

Die Begriffe Narration und Interpretation kennzeichnen das für diese Untersuchung gewählte Vorgehen bei der Erhebung und Analyse des empirischen Datenmaterials. Das wichtigste Erhebungsinstrument sind leitfadengestützte Interviews, aus denen Fallstudien zu den verschiedenen Integrationsprozessen gebildet werden. Der Fallstudienansatz hat sich in der sozialwissenschaftlichen Methodendiskussion, trotz einiger Kritik und der starken Stellung quantitativer Methoden, als wertvolles Forschungsinstrument behauptet. Angesichts neuerer Theorieentwicklungen, die insbesondere prozessuale Erklärungsansätze verfolgen, scheint seine Bedeutung sogar zu steigen (Pentland, 1999).

2.1.Fallstudienansatz

In der sozialwissenschaftlichen Methodendiskussion wird der Fallstudienansatz kritisiert, weil aus einer zwar detaillierten aber singulären Beobachtung, keine generalisierbaren Schlüsse gezogen werden könnten. Dazu kommt, dass fallstudienbasierter Forschung der Ruf einer gewissen methodischen Nachlässigkeit anhaftet (Miles, 1979; Ragin, 1991). Andererseits wird der Fallstudienansatz für Fragestellungen empfohlen, die sich durch eine komplexe Variablenkonstellation auszeichnen, und bei denen die Erhebung des sozialen Kontexts wesentlich für die Beantwortung der Forschungsfragen ist (Dyer/Wilkins, 1991). So sei Fallstudienforschung ein Instrument, das zur Theoriebildung führe, aber weniger geeignet ist, Theorien zu testen (Eisenhardt, 1989).

Als eine Lösung des Problems der mangelnden Verallgemeinerbarkeit von „in-depth" Fallstudien, wird der Fallstudienvergleich gesehen (Yin, 1981; 1989; Eisenhardt, 1991). Mit dem systematischen Vergleich von Fallstudien könne dieser Ansatz methodologisch ebenso abgesichert werden, wie Survey-Studien und andere quantitative Ansätze. Dabei werde allerdings nicht die statistische sondern die analytische Generalisierbarkeit angestrebt, d.h. der Vergleich der empirischen Daten mit den zentralen Aussagen und Thesen der zugrundeliegenden Theorien (Yin, 1989, S.38).

Die praktische Umsetzung des Fallstudienvergleichs ist allerdings voraussetzungsvoll. Insbesondere bei internationaler Forschung wird die tatsächliche Vergleichbarkeit von Fällen durch situative Faktoren auf verschiednen Ebenen wie Firma und Nation eingeschränkt. So ist es kaum möglich, Fälle auszuwählen, die unter identischen Bedingungen Internationalisierungs- und Integrationsprozesse durchlaufen. Viele Besonderheiten werden erst während der Untersuchung deutlich. Ein Wechsel der untersuchten Firma ist dann meist nicht mehr sinnvoll, auch aufgrund des nicht unerheblichen Aufwandes der Zugangseröffnung. Während die Unterschiede der untersuchten Firmen in der Interpretation der Fälle berücksichtigt werden können, ergeben sich Vergleichsprobleme auch im Hinblick auf die Tiefe des Zugangs. Nicht alle Firmen sind bereit, die gewünschten Gesprächspartner zu vermitteln oder wollen es bei einem Gespräch bewenden lassen, sei es aus Gründen der Informationspolitik, fehlendem Interesse oder Zeitmangel. Die Vergleichbarkeit ist deshalb auch aufgrund von Zugangsrestriktionen eingeschränkt, die nicht im Einfluss der Forscher liegen.

2.2. Narration und Prozess

Der Fallstudienansatz erscheint besonders geeignet, der im Konzept formulierten wissensbasierten und prozessualen Perspektive eine empirische Basis zu liefern. Gerade das

oft kritisierte ‚narrative Element' der Fallstudienforschung (Lippert, 1999, S.31), ist eine wichtige Grundlage, um den Ablauf von Ereignissen, unter Bezugnahme auf prozessuale Ansätze, zu erklären. Wie Pentland (1999) ausführt, können durch Narration und „Stories" nicht nur Daten zur Sequenz von Ereignissen erhoben werden, sondern auch Informationen über die Akteure, die narrative Perspektive und den evaluativen und kulturellen Kontext gewonnen werden. Narrationen sind aber nicht nur Träger von Informationen über Sequenzen und Kontexte, sie sind als Ausdruck einer bestimmten Kultur selbst handlungsanleitend und ein Mittel der Sozialisation (ebd., S.716). Narrationen bieten also einen Zugang zum kollektiven Wissen einer Organisation, das im wissensbasierten Ansatz als Wesen der Unternehmung identifiziert wurde. Pentland formuliert dies so: „when we analyze narrative, we are starting with raw material that is central to the cognitive and cultural world of our subjects" (ebd., S.717).

Laut Pentland können Geschichten auch selbst als Prozesstheorien verstanden werden. Um aber die „Prozesstheorie", die in einer Narration enthalten sein kann, zu verstehen, muss ihre Tiefenstruktur deutlich werden. Die Tiefenstruktur stellt den generativen Mechanismus eines Prozesses dar. Als Beispiele für generative Mechanismen seien die vier, von Van de Ven und Poole (1995) identifizierten Mechanismen, die hinter Veränderungsprozessen in Organisationen stehen, genannt: Lebenszyklusprozesse, teleologische Prozesse, ökologische (evolutionäre) und dialektische Prozesse. Der generative Mechanismus eines Prozesses ist unabhängig von den spezifischen Details einer Geschichte. Er bildet den Motor, der vielfältige Oberflächenphänomene eines Prozesses in unterschiedlichen Kontexten erzeugt.

Wie eine Narration oder ein Prozess dargestellt und beschrieben wird, ist allerdings vom Standpunkt des Erzählers abhängig und kann deshalb sehr unterschiedlich ausfallen. Die Zahl möglicher Beschreibungen und Ausgestaltungen ist, auch wenn ein spezifischer Fokus vorhanden ist, prinzipiell unbegrenzt. Damit kommt ein anderes Problem idiographischer Forschung in den Blick, nämlich die Handhabung von Interviews und der Umgang mit dem erhobenen Datenmaterial.

2.3. Interviews als zentrales Erhebungsinstrument

Für die vorliegende Untersuchung wurden zur Erhebung der Daten leitfadengestützte Interviews geführt. Durch leitfadengestützte Interviews können die Themen eines Gespräches am Forschungsinteresse ausgerichtet werden, und ermöglichen gleichzeitig, im Gegensatz zu standardisierten Fragebögen, eine flexible Gesprächsführung (Meusser/Nagel, 1991, S.449). Gleichzeitig signalisieren sie dem Interviewten die inhaltliche Kompetenz des Interviewers, und im Idealfall gibt es ein gemeinsames Interesse von Forscher und befragter Person an einer

Thematik, die zu einer positiven Grundstimmung und prinzipiellen Offenheit in der Gesprächssituation führt (ebd. S.450).

Die Annahme, dass sich durch eine möglichst optimale Interviewsituation - vorbereitete Forscher, in der Sache kompetente und offene Interviewpartner - entsprechend relevante Daten erheben lassen, wird allerdings immer häufiger hinterfragt. Der Einfluss postmoderner und sprachanalytischer Ansätze hat zu grundsätzlichen Zweifeln an dem „Wirklichkeitsgehalt“ von Interviewdaten geführt. Dabei geht es nicht um die natürlich immer schon in Betracht kommende Möglichkeit, dass in Interviews die Wahrheit verschwiegen oder zurecht gebogen wird. Vielmehr ist zu bedenken, dass die interviewten Personen mit ihren Aussagen vorgegeben Mustern folgen und die Wirklichkeit oder „Wahrheit“ somit nicht bewusst, sondern fast zwangsläufig verzerren. Alvesson (2003, S.18f.) kritisiert deshalb eine Sichtweise, die Interviews nur als Instrumente der Informationsbeschaffung betrachtet. Vielmehr müssten Interviews auch selbst als komplexe soziale Phänomene verstanden werden, in denen verschiedene Bedeutungsebenen vorhanden sind und Realitätskonstruktionen vorgenommen werden. Er nennt beispielhaft acht Metaphern oder Konzeptionen von Interviewsituationen, die solche Realitätskonstruktionen spezifizieren: (1) Der Kontext eines Interviews (Teilnehmer, hierarchische Positionen, Geschlecht, etc.) wirkt sich auf die Aussagen aus. (2) Befragte entwickeln eigene Vorstellungen über das Forschungsinteresse und formen dementsprechend ihre Aussagen. (3) Befragte nutzen das Interview zur Konstruktion eines mehr oder weniger realistischen Selbstbildes oder einer Identität. (4) Befragte nutzen ‚scripts’, also bestehende Denk- und Erklärungsmuster, die von Firmen, Organisationskulturen etc. vorgegeben werden, für ihre Aussagen. (5) Das Interview wird zur Darstellung der (moralischen) Überlegenheit des eigenen organisationalen Handelns genutzt (Impression Management). (6) Aussagen werden nach organisationspolitischen Erwägungen getroffen. (7) Befragte konstruieren zur Beantwortung von Fragen Realität auch mit Hilfe von fiktionalen Elementen, um Erwartungen hinsichtlich ihrer „Authentizität“ zu erfüllen. (8) Befragte können gar nicht „authentisch“ antworten, sondern geben den herrschenden Managementdiskurs wider. Das Vokabular und die Aussagen eines Diskurses konstituieren demnach das Denken von „Subjekten“. In letzterer Sichtweise, die dem Poststrukturalismus zuzurechnen ist, hat das Subjekt kaum Möglichkeiten, sich außerhalb eines gegeben Sprachgebrauchs zu stellen, und abweichende Bedeutungen zu produzieren.

Die genannten acht Punkte sollen zeigen, dass die Interviewsituation und Interviewaussagen in der Regel eine Mischung aus Wissensweitergabe und sozialen, politischen, psychologischen und sprachlichen Mustern ist. Angesichts der sozialen

Komplexität der Interviewsituation rät Alvesson (2003, S.25) zu einem pragmatischen und reflexiven Umgang mit dem erhobenen Material. Reflexivität bezeichnet er als den bewussten und durchgängigen Versuch, einen Gegenstand aus verschiedenen Blickwinkeln zu betrachten, ohne eine Sicht- und Denkweise a priori zu bevorzugen. Reflexivität bedeute somit, mit mehreren Interpretationsmöglichkeiten zu arbeiten, und im Umgang mit Erklärungsmustern selbstkritische Distanz zu bewahren.

Die acht genannten Metaphern über Interviewsituationen können einen Anhaltspunkt bieten, Aussagen aus verschiedenen Blickwinkeln zu interpretieren, ohne dabei jedoch das Ziel der Wissens- und Informationserhebung ganz aufzugeben. Nimmt man die Konzeption von Interviews als komplexer sozialer, sprachlicher und subjektiver Prozess ernst, sollten auch die Widersprüche und Mehrdeutigkeiten des Datenmaterials nicht einem singulären Interpretationsmuster und der vermeintlichen Schlüssigkeit einer Narration, untergeordnet werden.

Für die Fragestellung dieser Arbeit sind Interviews das wichtigste Instrument, um einen tiefergehenden Einblick vor allem in die soziale Dimension des Integrationsprozesses zu erlangen. Dafür soll möglichst eine vollständige Sequenz des Prozesses beschrieben werden können, und die Akteure und Mechanismen, die den Prozess antreiben, identifiziert werden. Darüber hinaus sollen Identitätskonstruktionen auf der Organisationsebene sichtbar gemacht werden. Identitätskonstruktionen sind Teil des kollektiven Wissens, bzw. mit diesem identisch und beinhalten die Normen, symbolischen Formen und Regeln, die das Handeln der Akteure anleiten. Gleichzeitig wird darauf zu achten sein, wie Identitäten konstruiert und möglicherweise auch instrumentell gehandhabt werden, um bestimmte Ziele durchzusetzen oder sich Erwartungen zu entziehen.

2.4. Methodischer Anspruch und praktische Umsetzung

Bei der praktischen Umsetzung der Erhebung bestünde im optimalen Fall Zugang zu Interviewpartnern bei allen an einem Integrationsprozess beteiligten Firmen, um möglichst alle Sichtweisen abzudecken. Dies war hier vor allem aufgrund der internationalen Dimension des Forschungsvorhabens nicht zu leisten. Für den dazu notwenigen Reiseaufwand standen zum einen nicht die notwendigen Mittel zur Verfügung, aber auch der Zugang zu den erreichbaren und gewünschten Interviewpartnern konnte nicht immer hergestellt werden. So musste versucht werden, ein möglichst vollständiges Bild aus dem vorhandenen und zugänglichen Informationen und Daten zu zeichnen und gleichzeitig die Leerstellen zu benennen und einseitige Darstellungen zu vermeiden.

Ein weiteres Problem der Verbindung von Untersuchungskonzept und empirischer Erhebung besteht in der Länge des Integrationsprozesses. Zum einen ist es schwierig das Ende eines Integrationsprozesses objektiv festzustellen. So haben auch die Evolutionsmodelle der Allianzentwicklung einen offenen Charakter (s. Teil A, Kapitel 5.3.). Interessant sind für die Untersuchung vor allem die ersten Jahre nach dem Beginn der Kooperation. Es kann aber nicht der Anspruch erhoben werden, abgeschlossene Episoden untersucht zu haben.

2.5. Firmen und Interviewpartner

Für die empirischen Untersuchungen wurden sowohl Firmen der Sekundär- als auch der Primärbranche der Softwareindustrie ausgewählt, wobei die Mehrzahl der Sekundärbranche zuzurechnen sind. Nach Produktmerkmalen sind sowohl Hersteller von ‚packaged Software', ‚embedded Software' und von ‚customized Software' vertreten.

Zu den interviewten Personen gehören sowohl Softwareentwickler als auch Projektmanager, Personalmanager, einige Vertriebsmanager und leitende Führungskräfte. Insgesamt wurden 56 Interviews geführt, mit einem Schwerpunkt bei der amerikanischen Firma MultiCom[35], zwei Joint Venture Firmen von MultiCom in Deutschland und Frankreich und einer Tochterunternehmung in Großbritannien. Die Informationen über die Kooperationen von MultiCom in Österreich und in Indien wurden in den nordamerikanischen Zentrallabors erhoben. Darüber hinaus wurden bei kleineren und mittleren Firmen aus Deutschland und der Schweiz je ein Interview durchgeführt und es wurden zwei Expertengespräche mit Verbandsvertretern geführt.

Die Interviews wurden auf der Basis von themenorientierten Leitfäden geführt. Die Gesprächsführung war offen für Schwerpunktsetzungen durch die interviewten Personen, die sich aus dem jeweiligen Tätigkeitskontext ergeben haben. Die Interviewdauer lag in der Regel bei ein bis zwei Stunden, teilweise auch darüber. Die Tabelle gibt eine Übersicht über die zeitliche Verteilung und die Anzahl der Interviews.

[35] Name geändert

	MultiCom Central Lab A (Nordamerika)	MultiCom Central Lab B (Nordamerika)	MultiCom Primus (Deutschland)	MultiCom FrenchCel (Frankreich)	Weitere Firmen (D+CH)
1995			4		
1996	6	4	16	3	
1997	4		3		
1998	3		5		
1999					
2000					2
2001					6