10 Softwareentwicklung in komplexen Unternehmensverbünden durch Berücksichtigung der organisatorischen Veränderungsprozesse

Beate Stoffels, Kirsti Grobel, Klaus Henning
Lehrstuhl Informatik im Maschinenbau (HDZ/IMA), RWTH Aachen
Sebastian Kutscha,
sd&m GmbH + Co KG, München

10.1 Abstract

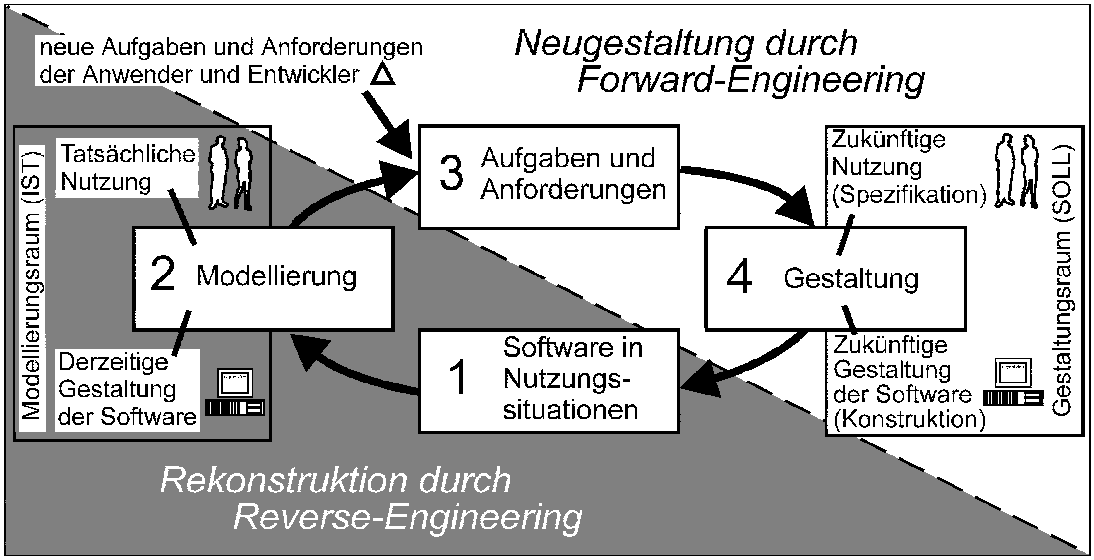
Selbst bei der Entwicklung neuer Softwaresysteme müssen die vorhandenen Systeme berücksichtigt werden, da sie Arbeitsabläufe und Organisationsstrukturen abbilden. Der Softwareentwicklungsprozeß kann unerwartete Entwicklungen auf organisatorischer Ebene auslösen, die sogar den Projekterfolg insgesamt gefährden können. Findet die Softwareentwicklung zudem in komplexen Unternehmensverbünden statt, sind oftmals die Aufgaben, Rollen und Interessen der beteiligten Unternehmen unklar. Zusätzlich zur rein technischen Software-Neu-Entwicklung muß daher der technische und organisatorische Gesamtzusammenhang des bestehenden Systems in den Designprozeß mit einbezogen werden. Ausgehend von einem großen Softwareprojekt (> 100 Personenjahre auf Entwicklerseite, Laufzeit > 4 Jahre), in dem Händler eines weltweiten Vertriebsnetzes Zugriff auf eine zentrale Datenbank beim Hersteller erhalten sollen, ist ein Vorgehensmodell zur Planung, Abwicklung und Bewertung des Projekts entwickelt worden. Dieses basiert auf der Identifizierung der verschiedenen Nutzergruppen und auf dem Task-Artifact-Cycle als grundsätzliches Vorgehensmodell für Reengineering-Projekte jeweils auf einer organisatorischen und technischen Projektebene.

10.2 Problemfelder in komplexen Softwareentwicklungs-Projekten

Die Entwicklung komplexer Softwaresysteme kann heute nicht mehr als rein technischer Designprozeß „auf der grünen Wiese“ betrachtet werden. Vorhandene Systeme müssen berücksichtigt werden, die nicht nur aus Technik, sondern auch aus Menschen und Organisationen bestehen. Die Software bildet Arbeitsabläufe und Organisationsstrukturen ab. Der Softwareentwicklungsprozeß kann unerwartete Entwicklungen auf organisatorischer Ebene auslösen, die ein Ausmaß haben können, daß sogar der Projekterfolg insgesamt gefährdet sein kann. Die große technische Komplexität der Softwareprojekte auch aufgrund der über Jahrzehnte hinweg gewachsenen bestehenden Systeme wird in organisatorischer Hinsicht dadurch verstärkt, daß der Entwickler mit vielfältigen Interessengruppen beim Kunden konfrontiert wird. So kann aus einer ursprünglich technischen Fragestellung eine Fragestellung werden, bei der die Neugestaltung von Geschäftsprozessen verschiedener Nutzer der Software in den Vordergrund rückt. Zusätzlich zur rein technischen Software-Neu-Entwicklung muß daher der technische und organisatorische Gesamtzusammenhang des bestehenden Systems in den Designprozeß mit einbezogen werden. Dadurch enthält die Software-Neu-Entwicklung Reengineering-Anteile auf technischer wie auch organisatorischer Ebene. Findet die Softwareentwicklung zudem in komplexen Unternehmensverbünden statt, sind oftmals die Aufgaben, Rollen und Interessen der beteiligten Unternehmen unklar.

Ausgehend von einem großen Softwareprojekt (> 100 Personenjahre auf Entwicklerseite, Laufzeit > 4 Jahre), in dem Händler eines weltweiten Vertriebsnetzes Zugriff auf eine zentrale Datenbank beim Hersteller erhalten sollen, ist daher ein Vorgehensmodell zur Planung,

Abb. 1: Methode für das Software-Reengineering (Kesselmeier, 1997)



Abwicklung und Bewertung des Projekts entwickelt worden. Dieses basiert zum einen auf dem Task-Artifact-Cycle als grundsätzliches Vorgehensmodell für Reengineering-Projekte. Zum anderen auf der Unterscheidung verschiedener Nutzergruppen und der Anwendung des Task-Artifact-Cycles jeweils auf einer organisatorischen und technischen Projektebene.

10.3 Projekthintergrund der Fallbeispiele

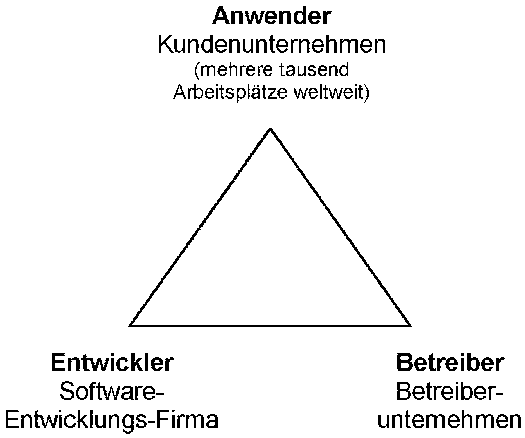
Grundlage für diese Untersuchung ist ein Vertriebsprojekt, ein Business Process Reengineering-Projekt, das die Verkürzung der Lieferzeiten sowie eine größere Transparenz der Vertriebsprozesse zum Ziel hat. Erreicht werden soll dies durch die Zusammenfassung von Bestell-, Produktions- und Distributionsdaten in einer zentralen Datenbank und durch Schaffung der Möglichkeit, Produktionskapazitäten direkt in dieser Datenbank zu buchen. Dieses Verfahren ist vergleichbar mit der Platzbuchung für Passagiere in Flugzeugen.

Betroffen von diesem Projekt ist die gesamte Vertriebsorganisation, bestehend aus einer Zentrale, verschiedenen Großhändlern sowie Händlern auf allen Kontinenten und deren Verkäufern. Insgesamt sollen mehrere tausend Arbeitsplätze an die Datenbank angeschlossen werden, auf der rund um die Uhr Kapazitätsbuchungen vorgenommen werden können. Diese Umstrukturierung umfaßt neben der Neuentwicklung von Software auch eine parallele Umstrukturierung der Vertriebsprozesse und verändert damit sowohl die Arbeitsprozesse der betroffenen Anwender als auch die Beziehungen der Vertriebseinheiten untereinander.

10.4 Konzepte für Reengineering-Projekte

Kesselmeier (1997) hat aus dem Task-Artifact-Cycle von Carroll (1991) eine Methode für das Software-Reengineering (Abb. 1) entwickelt, die für das Software-Reengineering zwei Phasen unterscheidet: die Rekonstruktion durch Reverse-Engineering und die Neugestaltung durch Forward-Engineering. Wesentliche Eigenschaften dieser Methode sind unter anderem die

Abb. 2: Nutzertypen von Software-Systemen



- ❒ explizite Berücksichtigung der derzeitigen technischen Gestaltung und der tatsächlichen Nutzung (Phasen 1 und 2) und das
- ❒ iterative Vorgehen im Sinne eines kontinuierlichen Gestaltungs- und Verbesserungsprozesses.

Aufbauend auf der Arbeit von Kesselmeier (1997) entwickelt Grobe (1998) ein Handlungskonzept, das Ansätze des Reengineering auf der organisatorischen Ebene mit Ansätzen des Reengineering auf der technischen Ebene, also des Software-Reengineering verbindet. Parallel zu dem Technikentwicklungsprozeß nach Kesselmeier (1997) wird hier die organisationsbezogene Analyse und Gestaltung durchgeführt. Dabei werden wiederum vier Arbeitsschritte durchlaufen:

1. Analyse der organisationsbezogenen Ist-Situation
2. Modellierung mit dem OSTO-Systemmodell
3. Organisationsbezogene Aufgaben und Anforderungen
4. Gestaltung in Pilotphasen

Aufbauend auf den Arbeiten zum Software-Reengineering von Kesselmeier (1997) bietet das Handlungskonzept von Grobe (1998) eine Methodik für allgemeine Reengineering-Projekte, das technische und organisatorische Aspekte gleichermaßen berücksichtigt und miteinander verbindet.

10.5 Entwicklung des Vorgehensmodells

Hier wird nun ein Vorgehensmodell zur Unterstützung komplexer Softwareentwicklungs-Projekte in zwei Stufen vorgeschlagen: erstens die Identifizierung der Nutzergruppen, deren Aufgaben, Ziele und organisatorische Einbindung; zweitens die Anwendung des Handlungskonzeptes von Grobe für jede der spezifischen Nutzergruppen auch für Software-Neu-Entwicklung, sobald sie in gewachsene Unternehmensstrukturen eingreift.

10.5.1 Nutzergruppen

Im Projekt ist neben der Software-Entwicklungsfirma und dem Kundenunternehmen auch der Betreiber der Rechenzentren und der Anwendungssoftware des Kundenunternehmens beteiligt. Dabei war die Aufgabenteilung ursprünglich in folgender Weise vorgesehen: Die organisatorische Umgestaltung der Vertriebsprozesse sollte von einem Projektteam des Kundenunternehmens (Anwender) vorgenommen werden; die Softwareentwickler (Entwickler) sahen ihre Aufgabe in der technischen Realisierung der zugehörigen Software; Aufgabe der Betreiber des Rechenzentrums (Betreiber) war der Betrieb des Altsystems und der Betrieb der neuen Software. Die verschiedenen Unternehmen und deren verallgemeinerte Rollen im Projekt zeigt Abb. 2. Ein Reengineering des Altsystems war nicht vorgesehen.

Die einzelnen Rollen lassen sich wie folgt charakterisieren:

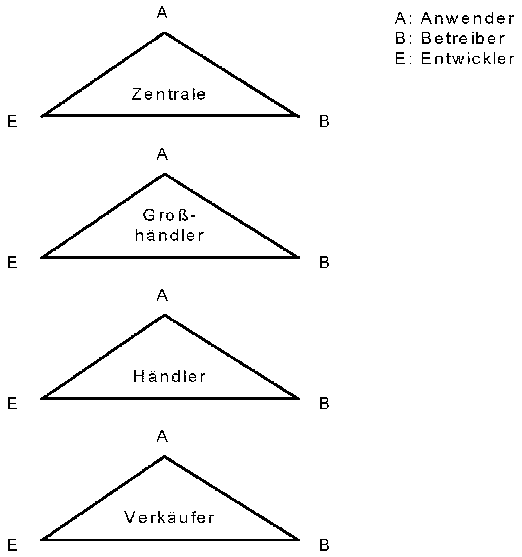
Anwender (auch Endanwender):
Ziel der Software ist, den Anwender in seiner Arbeitsaufgabe zu unterstützen. Lediglich beim Anwender kann die Software daher für das Unternehmen Nutzen bringen, indem die Arbeitsprozesse besser, schneller und effektiver ausgeführt werden.

Betreiber:
Die Betreiber sind die Mitarbeiter der Rechenzentren und EDV-Abteilungen, die dafür sorgen, daß die Anwendungssoftware und die zum Betrieb der Anwendungssoftware benötigte Hardware- und Basissoftware sowie eventuelle Netzverbindungen sicher zur Verfügung stehen. Sie liefern damit die technische Basis sowohl für die Anwendung als auch für die Entwicklung. Aus Sicht des Kundenunternehmens ist der Betrieb des Softwaresystems ein Kostenfaktor.

Entwickler:
Während eines Softwareprojekts gehören die Entwickler häufig zu einer Softwarefirma. Entwickler können jedoch auch Mitarbeiter des Kunden- oder Betreiberunternehmens sein, die nach Abschluß einer Entwicklungsphase im Betrieb selbst die Wartung und Weiterentwicklung der Software durchführen. Beide sind in Anlehnung an Kesselmeier (1997) Zulieferer für den Anwender (als Nutzer der Funktionalität) und den Betreiber (als Nutzer der Strukturen).

Die Identifizierung der einzelnen Nutzergruppen ist bedeutend, um deren Interessen und Kräfte im Projektverlauf abschätzen und damit in die Softwareentwicklung mit einbeziehen zu können. Die Bedeutung der einzelnen Nutzergruppen kann je nach Art des Projekts unterschiedlich sein. Charakteristisch für das Beispielprojekt ist, daß für die Softwareentwickler beide Gruppen, Anwender und Betreiber eine wichtige Rolle spielen, da sowohl Altsysteme abgelöst werden müssen als auch neue Geschäftsprozesse implementiert werden sollen. Die Komplexität wird zusätzlich dadurch erhöht, daß die charakteristischen Nutzertypen, wie Abb. 3 zeigt, potentiell auf allen Stufen der Vertriebshierarchie vorkommen können. Das bedeutet zum Beispiel, daß dieselben Module der neuen Software von Mitarbeitern in der Zentrale, beim Großhändler und beim Händler genutzt werden. Die jeweiligen Anwender nutzen die einzelnen Softwaremodule unterschiedlich häufig und stellen damit auch unterschiedliche Forderungen an die Software (z. B. Schnelligkeit, Hilfe, etc.).

Abb. 3: Nutzergruppen im Projekt orientiert an der Vertriebshierarchie



10.5.2 Einführung von Projektebenen

Ausgehend von der Nutzerstruktur mit zwei verschiedenen Nutzertypen aus komplett getrennten Unternehmen muß das Handlungskonzept von Grobe (vgl. Kapitel 10.4) entsprechend angepaßt werden. Es genügt nun nicht mehr, lediglich für einen Nutzertyp die organisatorischen und technischen Randbedingungen zu betrachten. Vielmehr müssen auf vier aufeinander aufbauenden Ebenen Reengineering-Zyklen stattfinden. Diese Ebenen umfassen die technische und organisatorische Ebene bei Anwendern und Betreibern. Abb. 4 zeigt diese Ebenen für eine Stufe der Vertriebshierarchie.

Dabei erzwingt der Task-Artifact-Cycle auf jeder Projektebene einen Reverse-Engineering-Anteil, also die Aufarbeitung sowohl der derzeitigen Gestaltung von Technik und deren tatsächlicher Nutzung als auch der Gestaltung von Geschäftsprozessen. Auf jeder der Ebenen erfolgt die Entwicklung zyklisch und ist nicht nach einem ersten Durchlauf abgeschlossen. (Kesselmeier 1997, Grobe 1998). Im folgenden soll nun anhand zweier Fallbeispiele aus dem Projekt gezeigt werden, wie diese verschiedenen Ebenen aufeinander einwirken.

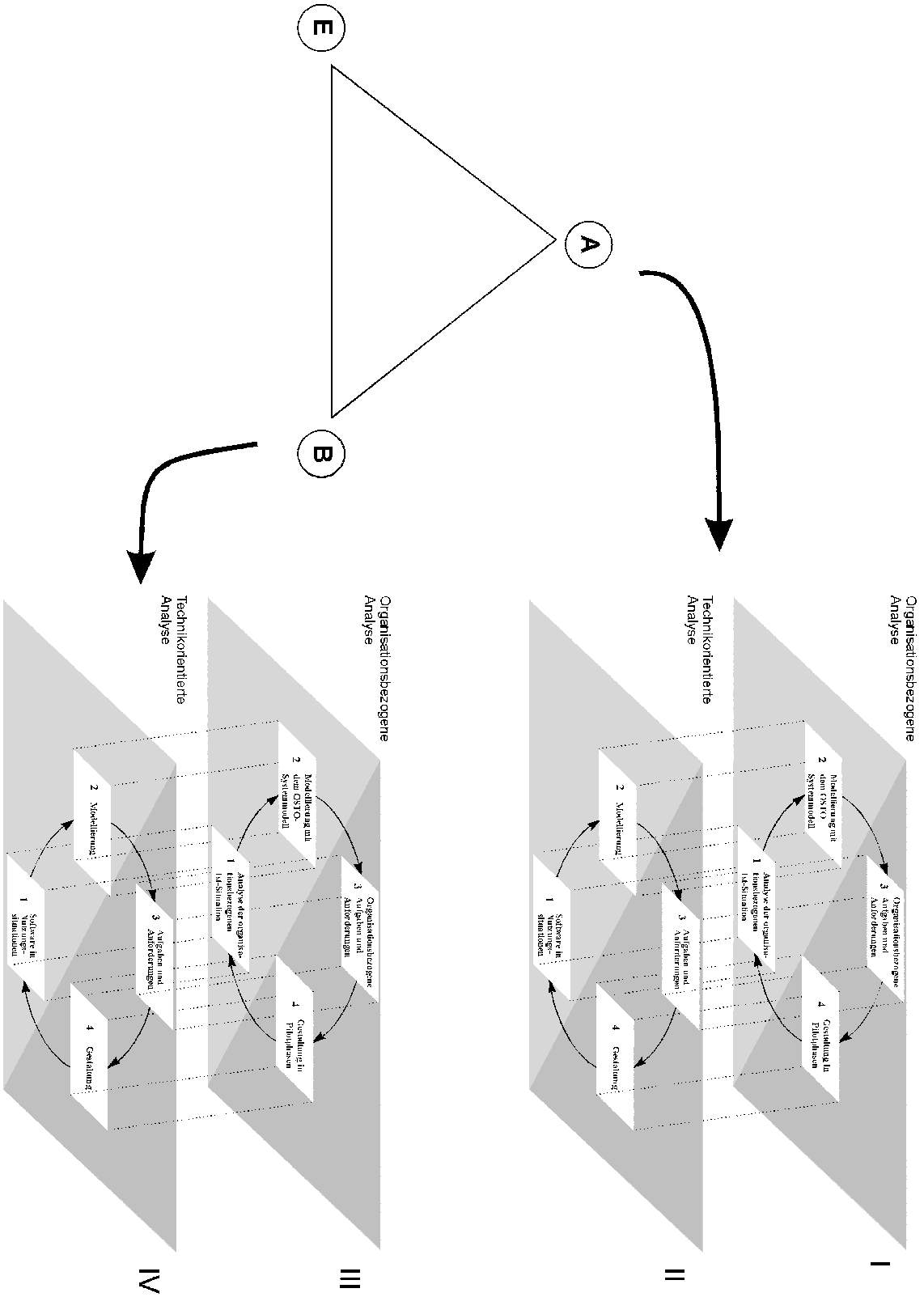
10.6 Fallbeispiele

10.6.1 Zusammenspiel zwischen Anwender, Betreiber und Entwickler im Gesamtprojekt

Im Laufe des Projektes wurde für die Entwickler deutlich, daß die zu Beginn des Projekts zwischen den Projektbeteiligten vereinbarte Aufgabenteilung und die Fokussierung auf die technische Seite[1] der Anwendungsentwicklung des Projektes nicht zielführend ist. Aufgrund der unzureichenden Größe des Kundenteams zur Umgestaltung der Vertriebsprozesse und aufgrund von Widerständen gegen diese Umgestaltung bei den zukünftigen Anwendern fehlten

[1] Mit technisch ist in Abgrenzung zu organisatorisch der gesamte Entstehungsprozeß der Anwendungssoftware inklusive der Einarbeitung in die Fachlichkeit gemeint, nicht jedoch die organisatorische Umgestaltung der Geschäftsprozesse.

Abb. 4: Vorgehensmodell für verschiedene Nutzertypen einer Stufe der Vertriebshierarchie; E: Entwickler, B: Betreiber, A: Anwender



den Entwicklern wichtige fachliche[2] Zulieferungen. Darüber hinaus stellte sich sehr spät heraus, daß der Reverse-Engineering-Anteil im Gesamtprojekt unterschätzt wurde, was zu großen Schwierigkeiten bei der Einführung der ersten Entwicklungsstufe führte. Die Entwickler waren schließlich gezwungen, zur Laufzeit in enger Kooperation mit den Betreibern Fehler im System zu beheben, um den Betrieb der Anwendungssoftware sicherzustellen.

Die Softwareentwickler sahen sich aufgrund dieser Abhängigkeiten in wachsendem Maße faktisch in der Rolle des Generalunternehmers des gesamten Projekts mit den vier Arbeitsschwerpunkten, die den Ebenen des Vorgehensmodells (Abb. 4) zugeordnet werden können:

- ❒ Software-Neu-Entwicklung (Ebene II)
- ❒ Ablösung des Altsystems (Ebene IV)
- ❒ Reengineering des Softwareprozesses (Ebene III)
- ❒ Business Process Reengineering (Ebene I)

Das erste Beispiel zeigt, daß die Ebenen im Vorgehensmodell nicht losgelöst voneinander betrachtet werden dürfen, sondern daß ihre Vernetzung und ihr Zusammenwirken schon bei der Planung von Projekten berücksichtigt werden muß.

10.6.2 Entwicklung einer Systemarchitektur in einem Teilprojekt

Das zweite Beispiel bezieht sich auf die Systemarchitektur. Das Gesamtprojekt wurde zur Bearbeitung in verschiedene Teilprojekte unterteilt, die jeweils eigenständig von Projektteams bearbeitet werden. Diese Unterteilung erfolgt aufgrund zweier Kriterien: Zum einen wird die Anwendung in logische Einheiten geteilt, die getrennt voneinander bearbeitet werden können und in der Regel einen eigenen Anwenderkreis bedienen. Zum anderen erfolgt eine Unterscheidung zwischen fachlichen Teilprojekten und technischen Teilprojekten. Aufgabe der technischen Teilteams soll die – von der konkreten Anwendung unabhängige – Bereitstellung der erforderlichen informationstechnischen Strukturen (Softwareproduktionsumgebung) für die Entwicklung der Anwendungen und deren Implementierung durch die fachlichen Teilteams sein. Auf der Kundenseite haben die Anwendungsentwickler aus dem fachlichen Teilteam engen Kontakt mit den Anwendern des betroffenen Fachbereichs, die Entwickler aus dem technischen Teilteam hingegen eher mit den Betreibern.

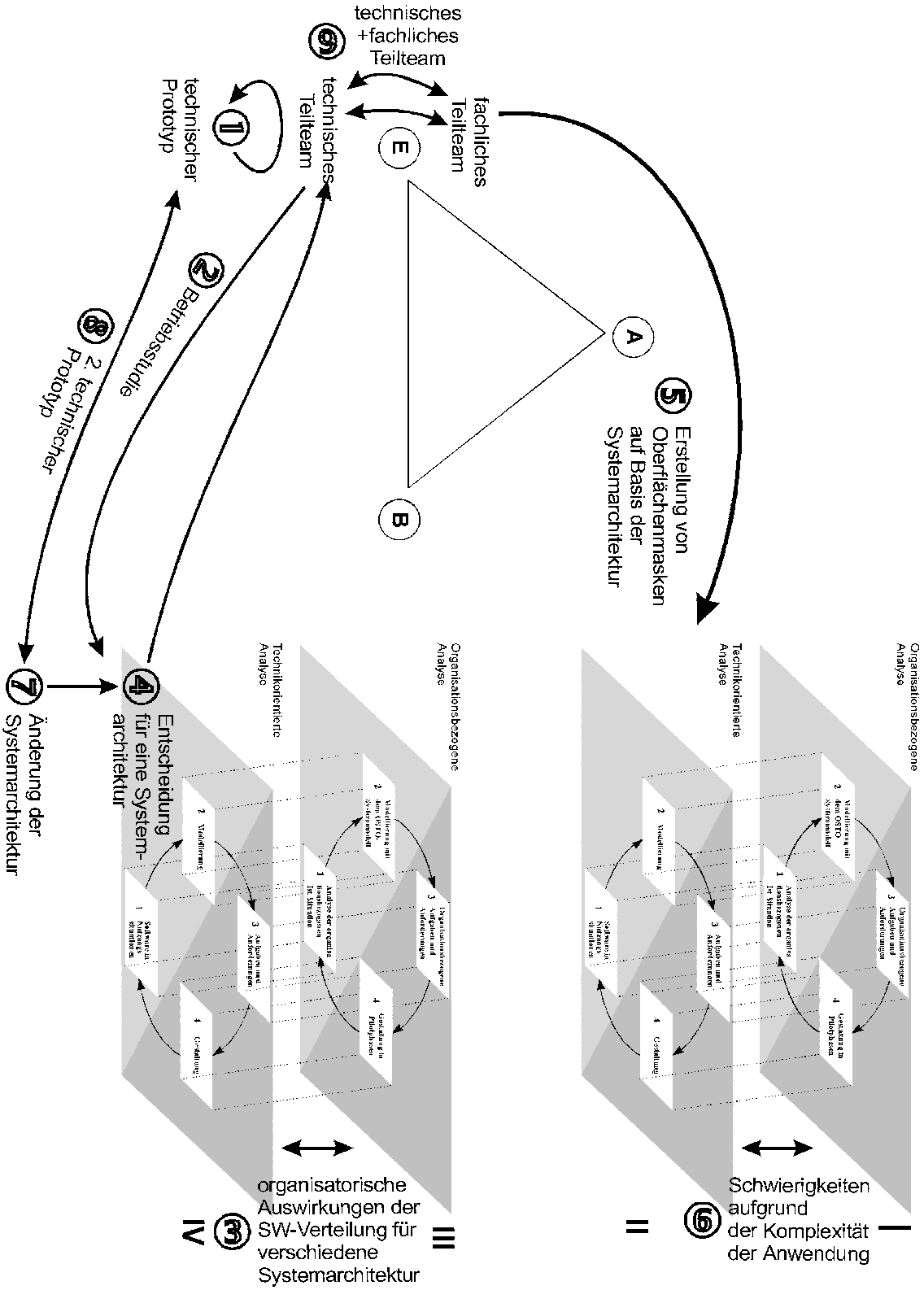
Im folgenden soll nun die Entwicklung einer Systemarchitektur und Entwicklungsumgebung für eine Client-Entwicklung zur Anbindung der Außenstellen analysiert werden. Die Aufgabe des technischen Teilteams war dabei, für das fachliche Teilteam eine Entwicklungsumgebung bereitzustellen. Technische Voraussetzung war dabei, daß zentrale Funktionalitäten und eine Datenbank auf einem Host genutzt werden mußten. Abb. 5 zeigt diese Entwicklung, deren einzelne Schritte im folgenden erklärt werden sollen.

1) Das betrachtete technische Teilteam hatte im Dezember 1996 anhand eines Prototypen die technische Machbarkeit einer Client-Server-Lösung mit Fat-Client in Smalltalk nachgewiesen. Diese sollte ursprünglich für alle Anwendungs-Teilprojekte eingesetzt werden.

---

[2] „Fachlichkeit“ und „fachlich“ bezeichnen die Aspekte der Anwendung, die zur Implementierung der Software erforderlich sind.

Abb. 5: Zusammenspiel der Ebenen bei der Entwicklung einer Systemarchitektur



2) Aufgrund von Zweifeln des Kunden (Anwender-Unternehmen) an der Betreibbarkeit einer solchen Client-Server-Lösung mit Fat-Clients wurde vom technischen Teilteam daraufhin eine Untersuchung von Betriebsaspekten verschiedener Systemarchitekturen durchgeführt. Die untersuchten Systemarchitekturen waren:
   - ❒ Eine klassische Hostarchitektur mit Terminals oder mit einer Terminalemulation auf den Client-Rechnern.
   - ❒ Smalltalk: Eine Client-Server-Architektur mit Smalltalk-Fat-Clients, die beispielhaft für alle Fat-Client-Varianten untersucht wurde.
   - ❒ Browser: Reine HTML-Lösung am Client ohne die Verwendung von Java Script oder Java.
   - ❒ Java: Realisierung der Client-Oberfläche als Java-Applet, das zur Laufzeit über das Netz auf den Client-Rechner geladen wird.

   Diese unterschiedlichen Architekturen wurden anhand der Kriterien Softwareverteilung, Infrastrukturanforderungen (Client u. Netz), relative laufende Kosten im Betrieb und Risiko (technisch und wirtschaftlich) untersucht und bewertet.

3) Während der Studie wurde deutlich, daß die Wahl der Systemarchitektur nicht nur erhebliche Auswirkungen auf die Benutzeroberfläche und die technischen Randbedingungen hat, sondern daß die Betriebsorganisation davon stark beeinflußt wird. Während Lösungen mit zentral wartbaren Anwendungsprogrammen (Terminal, Browser, Java) auch eine zentrale Betreiberstruktur begünstigen, setzt der Einsatz von Fat-Clients auch bei automatischer Softwareverteilung die Anwesenheit von Experten vor Ort beim Anwender voraus.

4) Im Rahmen einer Sitzung im März 1997 fiel in einem Entscheidungsgremium mit Vertretern von Anwendern, Betreibern und Entwicklern zunächst die Entscheidung zugunsten der Variante „Browser“ aus. Java sollte aufgrund seiner mangelnden Produktreife erst für spätere Anwendungskomponenten (ca. ab Mitte 1998) zum Einsatz kommen.

5) Auf der Grundlage dieser Entscheidung wurden im nächsten Schritt vom fachlichen Entwicklungsteam Beispielmasken mit HTML erzeugt.

6) Im weiteren Verlauf wurde deutlich, daß die Anwendung doch zu komplex war, um in HTML realisiert zu werden.

7) Die Systemarchitekturentscheidung wurde ohne erneute Einberufung des Entscheidungsgremiums zwar nicht völlig außer Kraft gesetzt, aber doch deutlich verändert.

8) Für diese neue Systemarchitektur wurde in enger Zusammenarbeit zwischen dem Entwicklungsteam der Betreiber und dem technischen Teilteam ein weiterer technischer Prototyp erstellt.

9) Um weitere Inkompatibilitäten zwischen den technischen Komponenten und der Anwendung zu vermeiden, wurde aus technischem und fachlichem Teilteam ein einziges Team.

Diese Entwicklung zeigt, daß in einem Prozeß, der auf einer der Projektebenen III oder IV angesiedelt ist, immer wieder Einflüsse von den Ebenen I und II eine Rolle spielen. Es wird

Tab. 1: Kriterienkatalog aufgeteilt nach Projektebenen

	Anwender	**Betreiber**
organisatorische Ebene	• Wer sind die Anwender der Software? • Wie ist deren organisatorische Einbindung? • Was sind ihre Arbeitsplätze, gegenwärtigen Arbeitsprozesse und Aufgaben? • Welche zukünftigen Aufgaben haben sie und welche davon sollen zukünftig SW-technisch unterstützt werden? • Wie verändert die Einführung der Software die Arbeitsplätze, organisatorische Einbindung etc.? • Welche Widerstände sind zu erwarten (z.B. Ängste, Besitzstandswahrung)?	• Welche Personen warten derzeit SW, HW und Netz? • Wo sind die Arbeitsplätze der Betreiber? • Wie ist deren organisatorische Einbindung? • Welche Widerstände gegen die neue Software gibt es? • Welches Verhältnis besteht zwischen Betreiber- und Anwenderorganisation? • Welche organisatorischen Auswirkungen hat die Wahl einer neuen Technik für die Betreiber? • Welche Qualifikationen bringen die Betreiber mit?
technische Ebene	• Welche Technik/SW benutzen die Anwender heute? • Welche Schwächen hat diese in bezug auf die Arbeitsprozesse und welche davon sollen durch die neue Software behoben werden? • Wie organisieren die Anwender ihre Arbeit, um trotz der Schwächen der Technik / Software ihre Aufgaben zu erfüllen? • In welcher Art und Weise soll die Technik zukünftig die Aufgaben unterstützen?	• Welche Technik (HW / SW / Netz) ist heute in Betrieb? • Wo sind die Probleme des Systems in Betrieb/Wartung/Weiterentwicklung? • Welche neue Technik ist überhaupt betreibbar? • Welche Tools zur Entwicklung (z.B. KM, SPU) stehen zur Verfügung? • Wie passen neue SW-Komponenten zum bestehenden System?

auch deutlich, daß es nicht möglich ist, eine Systementscheidung unabhängig von den konkreten Bedürfnissen der jeweiligen Nutzer, deren Arbeitsprozesse und deren organisatorischem Umfeld zu fällen, sondern daß die konkreten Erfordernisse in Kooperation mit den jeweiligen Nutzern erarbeitet werden müssen.

10.6.3 Kriterienkatalog

Die Fallbeispiele verdeutlichen, daß Softwareentwickler immer alle vier Ebenen im Blick haben müssen, um Wechselwirkungen zwischen den Ebenen abschätzen zu können. So können beispielhaft die Fragen der Tabelle 1 helfen, für jede Ebene die notwendige Transparenz zu schaffen. Für konkrete Anwendungen kann und soll dieser Fragenkatalog je nach den projektspezifischen Randbedingungen erweitert und angepaßt werden. Nach dem Task-Artifact-Cycle müssen jedoch immer sowohl Fragen zur bestehenden Situation als auch Fragen zur zukünftigen Gestaltung enthalten sein.

Dieser Kriterienkatalog verdeutlicht, daß zur Erfassung der Zusammenhänge zwischen dem technischen Entwicklungsprozeß und dem organisatorischen Veränderungsprozeß eine intensive Auseinandersetzung mit dem Nutzer und dessen Arbeitsprozessen erforderlich ist. Erst in dieser konkreten Auseinandersetzung, z. B. unter Zuhilfenahme von Prototypen, können die wesentlichen Kriterien für die Gestaltung der zukünftigen Software ermittelt werden. Diese wiederum sind erforderlich, um zu gewährleisten, daß die entwickelte Software auch tatsächlich den Bedürfnissen der Nutzer entspricht und deren Arbeitsprozesse optimal unterstützt. Der Kriterienkatalog gibt den Entwicklern damit eine Hilfestellung, um nach dem Vorgehensmodell in der konkreten Entwicklungssituation alle Ebenen im Blick zu behalten.

10.7 Zusammenfassung

Dieser Bericht stellt ein Vorgehensmodell für komplexe Software-Entwicklungs-Projekte vor, das die Softwareentwickler darin unterstützt, sowohl die technischen wie auch organisatorischen Ebenen im Blick zu behalten. Im ersten Schritt werden dazu die Nutzergruppen und konkreten Nutzer der Software identifiziert. Der Task-Artifact-Cycle als Entwicklungszyklus auf allen Ebenen erzwingt die explizite Berücksichtigung des soziotechnischen Unternehmenskontextes.

Zwei Fallbeispiele zeigen, wie anhand der Unterscheidung verschiedener Nutzergruppen und den technischen und organisatorischen Ebenen komplexe Software-Reengineering-Projekte analysiert werden können. Die Analyse zeigt, daß sowohl für das Gesamtprojekt als auch für einzelne Teilprojekte immer mehrere Ebenen zusammenwirken. Wesentliche Aspekte des Beispielprojekts können so erklärt werden. Ein Kriterienkatalog konkretisiert schließlich das Vorgehensmodell und erlaubt damit die Übertragung auf andere Anwendungsbeispiele.

Literaturverzeichnis

Carroll, J.M.; Kellogg, W.A.; M.B. Rosson
The Task-Artifact-Cycle. In: Carroll, J.M. (Ed.): Designing Interaction. Psychology at the Human-Computer Interface. Cambridge, 1991

Kesselmeier, H.
Entwicklung einer Methode für Software-Reengineering-Projekte, Dissertation RWTH Aachen, 1997

Grobe, J.
Reengineering von computerunterstützten Geschäftsprozessen am Beispiel von Großkrankenhäusern, Dissertation, RWTH Aachen, 1998

Kesselmeier, H. et al.
Enterprise Networks: The Reengineering of complex software systems, Proceedings of the 9th Symposium on Information Control in Manufacturing, IFAC, June 1998, Nancy