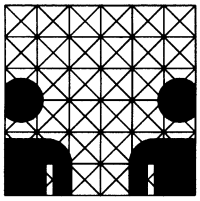
Universität Hamburg
Fachbereich Informatik
Diplomarbeit

Analyse-Methoden im Spannungsfeld von Softwareneuentwicklung und Einführung von Standardsoftware



Holger Tewis

Matrikelnummer: 47 253 11
Elligersweg 6
22307 Hamburg
Tel.: 040 – 500 48 236
eMail: tabacha@t-online.de

Erstbetreuer: Dr. Ralf Klischewski
Zweitbetreuer: Prof. Arno Rolf

Inhaltsverzeichnis

1 Einleitung

1.1 Problemstellung

Bei der Einführung von Standardsoftware im Vergleich zur Neuentwicklung einer Individuallösung wird normalerweise davon ausgegangen, dass die Neuentwicklung um ein Vielfaches aufwändiger ist (auch wenn es hier immer noch an empirisch gesicherten Daten fehlt). Je mehr ein Produkt "off-the-shelf" verwendet werden kann, umso weniger Anpassungs- und Entwicklungsaufwand braucht man. Je mehr ein Produkt neu entwickelt oder aus Einzelkomponenten zusammengefügt werden muss, umso größer ist der Aufwand.

Auf der einen Seite ist der Hauptgrund für die Einführung von Standardsoftware (insbesondere ERP-Software) der Einkauf von Standardwissen und Standardgeschäftsprozessen, die sich positiv auf die Organisation des Unternehmens auswirken. Aus diesem Know-how-Pool kann bei einer Neuentwicklung nicht geschöpft werden.
Andererseits lässt sich bei Standardsoftware nur schwer von den durch die Software festgelegten Gegebenheiten abweichen. Eine Reorganisation des Unternehmens ist zum Teil die einzige Lösung. Diese ist umso wahrscheinlicher, je mehr die Standardsoftware „off-the-shelf“ ist, also je weniger sie veränderbar und somit an die Organisation anpassbar ist. Der Reorganisationszwang wird dabei zum Teil als Vorteil deklariert, um veraltete Organisationsstrukturen leichter aufzubrechen.

Sowohl Neuentwicklung von Software als auch Einführung von Standardsoftware sind Softwareprozesse, zu deren Bewältigung in der Literatur umfangreiche Vorgehensmodelle angeboten werden. Teil der Softwareprozesse ist die Durchführung einer Analyse, die die Grundlagen der Neuentwicklung bzw. Einführung liefern soll. Dabei geben die Vorgehensmodelle der Analyse unterschiedliche zu analysierende Gegenstandsbereiche vor und erwarten von ihr unterschiedliche Ergebnisse. Diese Erwartungen und Vorgaben an Analyse werden zum Großteil aber nicht oder nicht explizit benannt. Der Stellenwert, der der Durchführung von Analyse eingeräumt wird, ist je nach Vorgehensmodell sehr unterschiedlich. In Analogie zum Entwicklungsaufwand steht der Wunsch, bei Einführung von „off-the-shelf“-Software möglichst wenig Analyse-Aufwand zu haben. Andererseits kann auch gefragt werden, ob die durch Einführung von Standardsoftware oft nötig werdenden Reorganisationsmaßnahmen nicht eine besonders gründliche Analyse benötigen oder zumindest eine Analyse, die speziell auf Reorganisation abgestimmt ist.

Nachdem es für Analyse in der Informatik viele Jahre keine eigenständigen Methoden gab, etablierten sich in den letzten Jahren einige Analyse-Methoden. In der Softwareentwicklung waren dies insbesondere objektorientierte Analysemethoden. Zur Modellierung konzentrieren sich immer mehr Methoden auf die Unified Modeling Language (UML). UML entwickelte sich zu der OO-Standardmodellierungssprache für Softwareentwicklung.
Auf der anderen Seite haben sich aber auch prozessorientierte Analyse-Methoden und Modelle (EPK, Petri-Netze) in Theorie und Praxis einen festen Platz erobert. Vor allem bei Einführung von Standardsoftware und Bereichen, denen sich besonders die Wirtschaftsinformatik widmet, sind diese nicht mehr wegzudenken. Geschäftsprozessanalyse dient hierbei vor allem der Vorbereitung der Gestaltung der Organisation, um das Potential von Standardsoftware voll auszuschöpfen.

Die Unterschiede zwischen prozessorientierten, objektorientierten und anderen Analyse-Methoden werfen in der Praxis die Frage auf, welche Methode man verwenden sollte. Besonders im Spannungsfeld zwischen Standardsoftwareeinführung und Softwareneuentwicklung besteht hier noch eine Methodenverwirrung. Da die geistige Heimat der Standardsoftwareeinführung in der Wirtschaftsinformatik liegt, ergibt sich auch die Chance, von dort Anregungen in die Informatik bzw. Softwaretechnik herüberzuholen und von den unterschiedlichen Sichtweisen zu profitieren. Eine Klassifizierung der existierenden Methoden und die Bereitstellung eines Katalogs von Methodenoptionen kann aus dieser Methodenverwirrung heraushelfen und dem Praktiker helfen, eine passende Methode auszuwählen. Das für den Praktiker zu lösende Problem lautet:

> „Welche der vielen existierenden Analyse-Methoden soll ich im Verlauf meiner Softwareentwicklung bzw. Standardsoftwareeinführung benutzen?“

Die Beantwortung dieser Frage soll auch klären, inwieweit Analyse-Methoden der Softwareneuentwicklung für die Einführung von Standardsoftware geeignet sind und umgekehrt.
Ist eine Analyse-Methode zur Neuentwicklung von Software fehl am Platze oder schießt sie über ihr Ziel hinaus, wenn "nur" Standardsoftware eingeführt werden soll? Bereitet die Analyse-Methode eventuell notwendige Reorganisationsmaßnahmen angemessen vor?
Mit Standardsoftware können andererseits zum Teil bestimmte Dinge nicht realisiert werden, ohne dass die Organisation stark umgestellt werden muss. In der Praxis gab es Fälle, in denen sich die Unternehmen dazu entschlossen, die Standardsoftware wieder durch eine Neuentwicklung abzulösen. Dann stellt sich die Frage, inwieweit Daten einer Analyse für die Neuentwicklung wiederverwendet werden können. Eine erneute Durchführung von z.B. Benutzer-Interviews wird auf Unverständnis der Benutzer stoßen.

1.2 Fragestellung

Die Fragestellung lautet:

> Wie lassen sich Analyse-Methoden klassifizieren, um für die Praxis eine je nach Softwareprozess angemessene Auswahl zu ermöglichen?

1.3 Methodik der Arbeit

Zur Beantwortung der Fragestellung wird in dieser Arbeit die Software-Praxis und Literatur aus der Perspektive von Softwareprozessen betrachtet (s. Abb. 1).
In der Literatur werden verschiedene Vorgehensmodelle angeboten, die jeweils bestimmte Softwareprozesse in der Praxis unterstützen wollen. In diesen Modellen kann Analyse als ein Teil identifiziert werden, an den bestimmte Anforderungen gestellt werden: zum einen werden bestimmte Vorgaben gemacht (Gegenstandsbereich, Methoden, Ressourcen etc.), zum anderen werden bestimmte Ergebnisse der Analyse erwartet.

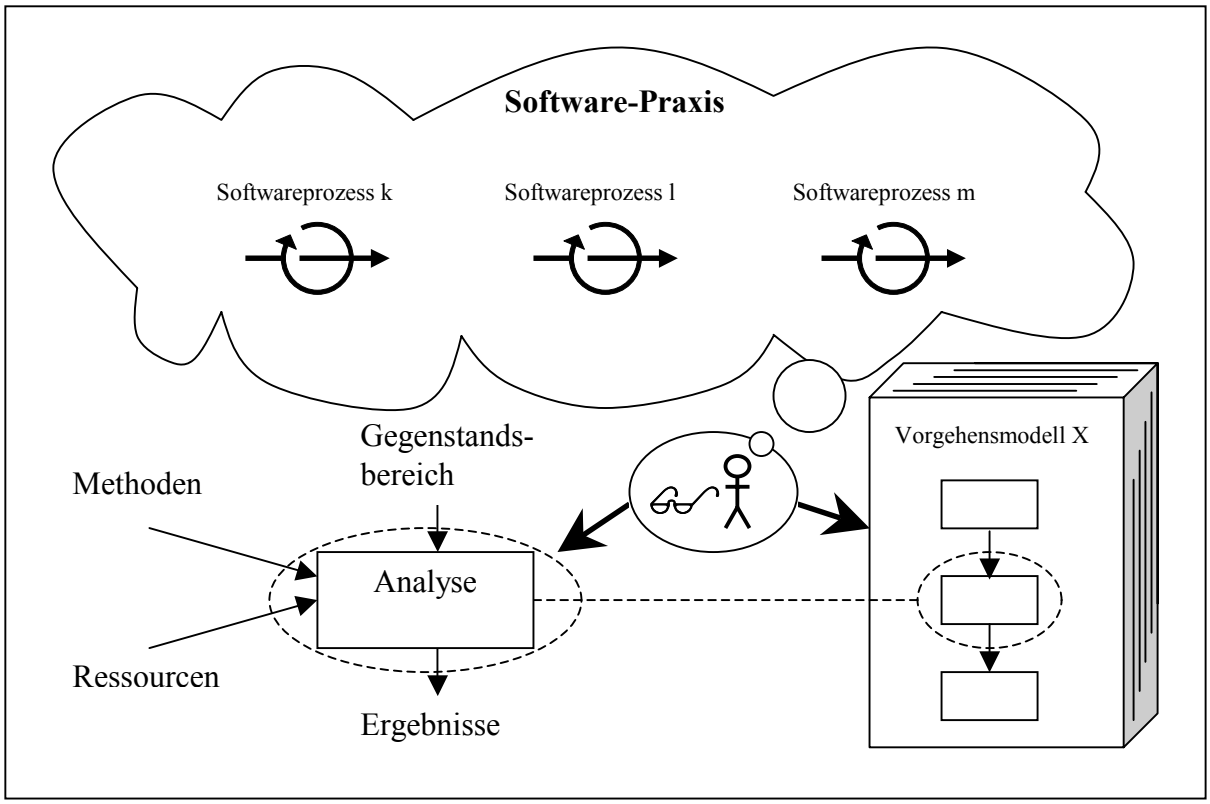


Abb. 1: Vorgehensmodelle und Analyse werden durch die „Softwareprozessbrille“ betrachtet.

Nach der Klärung der wichtigsten Begriffe werden im 2. Kapitel dieser Arbeit für fünf Vorgehensmodelle die an Analyse gestellten Anforderungen (Vorgaben und Ergebnisse) festgestellt. Die Analyse wird dabei als Blackbox betrachtet. Es entsteht eine Liste von Anforderungen für jedes Vorgehensmodell. Anhand der festgestellten Anforderungen lassen sich tendenzielle Unterschiede zwischen Vorgehensmodellen der Standardsoftwareeinführung und Vorgehensmodellen der Softwareneuentwicklung erkennen. Diese Unterschiede dienen später zur Zuordnung von Analyse-Methoden zu den Softwareprozessen.
Im 3. Kapitel wird in Analyse-Methoden hineingeschaut. Es werden Grundlagen, Ergebnisse und Vorgehensweisen von Analyse-Methoden betrachtet, ohne zu beachten, welchem Vorgehensmodell eine spezielle Analyse-Methode vielleicht zugeordnet wird. Es wird also im Gegensatz zu Kapitel 2 aus Sicht der Analyse-Methoden vorgegangen und geschaut, was diese von „außen“ als Grundlage erwarten bzw. als Ergebnis abliefern. So entsteht für jede Analyse-Methode eine Liste mit Grundlagen und Ergebnissen. Aus den Grundlagen und Ergebnissen lassen sich bestimmte Aspekte ableiten, die die Analyse-Methoden voneinander unterscheiden.
Das Ergebnis der beiden Sichtweisen sind zum einen Anforderungen an Analyse-Methoden aus Sicht von Vorgehensmodellen, die je nach Softwareprozess unterschiedliche Tendenzen haben. Und zum anderen Grundlagen und Ergebnisse von Analyse-Methoden, die sich in mehreren Aspekten unterscheiden. In Kapitel 4 wird gezeigt, dass die Aspekte der Analyse-Methoden den Tendenzen der Softwareprozesse entsprechen. Die Aspekte und Tendenzen machen dadurch eine Klassifizierung der Analyse-Methoden nach Softwareprozessen möglich. Verschiedene Erweiterungen der Methoden zeigen aber, dass diese zunehmend auch auf Softwareprozesse abzielen, die sie ursprünglich nicht unterstützen wollten.
Am Ende steht ein Methodenkatalog, der eine Auswahl der Analyse-Methoden in Abhängigkeit vom Softwareprozess ermöglicht. Dabei wird auch betrachtet, welche

Ergebnisse von Analyse weiterverwendet werden können, wenn sich der Softwareprozess ändert.
Hauptproblem der Arbeit war die Vielfalt der verschiedenen wissenschaftlichen Forschungsrichtungen. Softwaretechnik ist für sich alleine genommen schon sehr umfangreich. Die betrachtete Fragestellung fällt aber gleich noch in mehrere weitere sehr ausgedehnte wissenschaftliche Diskussionen:
Der Analyse-Begriff selbst ist zwar in der Informatik eher wenig ausgearbeitet, stattdessen gibt es aber Diskussionen unter den Begriffen Requirements Engineering (RE), Change Analysis oder auch Human Computer Interfaces (HCI), die sich alle in der einen oder anderen Form mit Analyse beschäftigen.
Auch hinter dem Begriff des Softwareprozesses steht eine umfangreiche Diskussion über Software Prozess Technologie, (z.B. Lecture Notes: Software Process Technology) Software Prozess Verbesserung (Improvement) etc.

Die Seite der Standardsoftwareeinführung befindet sich, was den theoretischen Teil angeht, im Vergleich dazu in einem klaren Untergewicht. Die methodischen Grundlagen werden erst in diesen Jahren geschaffen. So ist der Begriff „standard software“ in wissenschaftlichen Publikationen z.B. im amerikanischen Raum nur selten wahrzunehmen und scheint eher von Deutschland, dem Ursprungsland der „Standardsoftware Nr.1“ von SAP, auszugehen. Er fällt auch meistens im Zusammenhang mit Enterprise Resource Planning Software (ERP), ein Begriff, der aber auch erst in jüngerer Zeit verstärkt benutzt wird. ERP-Software gibt es mittlerweile von zahlreichen Anbietern und die einzelnen Produkte sind für sich genommen wiederum extrem komplex.
Hand in Hand mit Standardsoftware geht auch die Diskussion um Geschäftsprozessorientierung. Mit dem Organisationskonzept der Prozessorientierung könnte auch die Tür zum riesigen Bereich der Organisationstheorien aufgestoßen werden (was z.B. von Rolf für die Wirtschaftsinformatik vorgeschlagen wird [ROL98]). Das Untergewicht der theoretischen Grundlagen im Bereich Standardsoftwareeinführung wird sich aber vermutlich in den nächsten Jahren schnell ändern, da sich die in den letzten Jahren entstandene Wirtschaftsinformatik Geschäftsprozesse und Standardsoftware auf die Fahnen geschrieben hat.

Die hier betrachtete Fragestellung fällt auch in den Forschungsbereich Information Systems, der ein sehr ausdifferenziertes Verständnis von Informationssystemen in Organisationen hat. Zur Problematik der Standardsoftwareeinführung um ERP-Systeme hat dieser Forschungsbereich aber bisher, im Vergleich zur deutschen Wirtschaftsinformatik, wenig beigetragen. Andererseits erscheint die Ausdifferenzierung von Information Systems in manchen Bereichen für die Betrachtung des mit dieser Arbeit betretenden Neulandes zu umfangreich. Deswegen wurde auch von der Verwendung bereits bestehender Rahmen zur Evaluierung von Methoden abgesehen (z.B. [JAY94]) und eine eigene Methode verwendet, die sich auf die Vereinigung von Teilen der Wirtschaftsinformatik und Softwaretechnik beschränkt. Dennoch ist der Gegenstandsbereich dieser Arbeit immer noch sehr groß und kann nur durch die gewählte abstrakte Betrachtung bewältigt werden.

1.4 Gliederung

Tab. 1 skizziert die Zuordnung der Analyse-Methoden zu Softwareprozessen über den Verlauf der Arbeit.
In Kapitel 2 werden zuerst die Begriffe Softwareprozess, Vorgehensmodell und Analyse-Methode betrachtet. Dann wird für fünf Vorgehensmodelle festgestellt, welchem Softwareprozess sie sich selbst zuordnen und welche Anforderungen sie an Analyse stellen. Es werden

dann die Anforderungen an Analyse-Methoden festgestellt, die Vorgehensmodelle der Standardsoftwareeinführung von Vorgehensmodellen der Softwareneuentwicklung trennen.
In Kapitel 3 werden die Grundlagen und Ergebnisse von Analyse-Methoden festgestellt. Es ergeben sich Listen von Grundlagen und Ergebnissen. Es werden dann Aspekte hervorgehoben, in denen sich die Analyse-Methoden unterscheiden. Desweiteren werden verschiedene Erweiterungen der Methoden betrachtet.

2.1	Software-prozesse	**Softwareneuentwicklung**	**Standardsoftwareeinführung**
2.2	Vor-gehens-modelle	Modell nach Helmut Balzert RUP STEPS	IBSIS-Modell IMG
2.3	Anforde-rungen	Tendenz: Softwarekonstruktion Organisation → Software	Tendenz: Reorganisation Organisation ← Software
3.2	Aspekte von Analyse-Methoden	Klassenmodell Anwendungsfälle Softwareabläufe	Akteursübergrei-fende Abläufe Organisations-brüche
3.1	Analyse-Methoden	OBA OOA nach Heide Balzert	SA GPA

Tab. 1: Zuordnung von Analyse-Methoden zu Softwareprozessen.

In Kapitel 4 werden die Analyse-Methoden schließlich nach Softwareprozessen klassifiziert. Dies geschieht, in dem die Anforderungen aus Sicht der Vorgehensmodelle und die daraus abgeleiteten Tendenzen von Softwareprozessen den Aspekten der Analyse-Methoden gegenübergestellt werden. Die Aspekte lassen sich dabei einer Seite des Spannungsfeldes zwischen Standardsoftwareeinführungsprozessen und Softwareneuentwicklungsprozessen zuordnen. Auf dieser Basis entsteht eine Klassifikation der Analyse-Methoden in Abhängigkeit zu Softwareprozessen. Es zeigt sich eine klare Abgrenzung von Analyse-Methoden der Standardsoftwareeinführung und Softwareneuentwicklung. Die Betrachtung der Erweiterungen der Analyse-Methoden zeigt dann ein Verwaschen dieser Abgrenzung. Das Kapitel liefert schließlich einen Methodenkatalog, der dem Praktiker die Auswahl einer Analyse-Methode in Abhängigkeit vom Softwareprozess erlaubt. Das Kapitel übt außerdem eine ausführliche Methodenkritik.
Im Schlusswort (Kapitel 5) wird das Vorgehen noch einmal zusammengefasst und ein Ausblick auf Analyse-Methoden und Vorgehensmodelle der Zukunft gegeben.

2 Anforderungen an Analyse-Methoden im Rahmen von Softwareprozessen

2.1 Softwareprozess
Begriffe:
- Softwareprozess
- Vorgehensmodell
- Analyse-Methode
- Standardsoftwareeinführungsprozessmodell
- Softwareneuentwicklungsprozessmodell

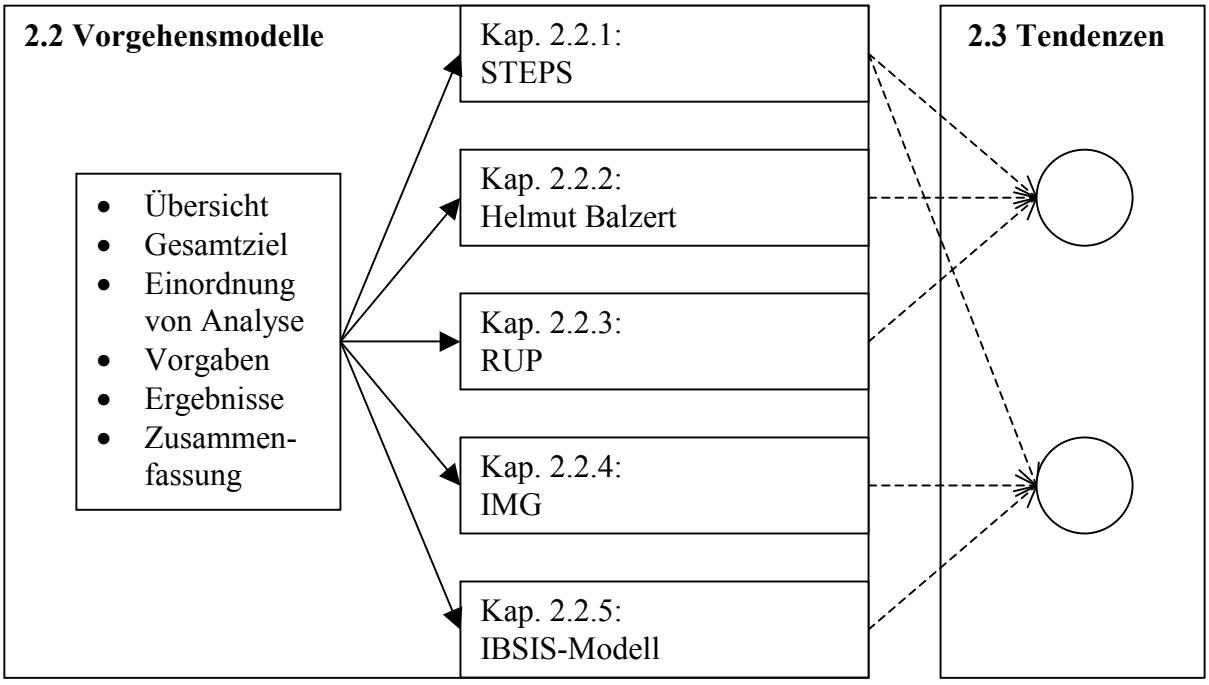


Abb. 2: Gliederung Kapitel 2

In diesem Kapitel wird für verschiedene Vorgehensmodelle betrachtet, welche Vorgaben sie Analyse-Methoden machen und welche Ergebnisse sie erwarten. Dazu werden als Erstes (Kap. 2.1) die Begriffe Softwareprozess, Vorgehensmodell und Analyse-Methode näher beleuchtet. Dann werden in Unterkapitel 2.2 für fünf Vorgehensmodelle aus der Literatur die von ihnen gestellten Ansprüche an Analyse-Methoden festgestellt. Das Unterkapitel 2.3 beschreibt schließlich die Unterschiede der Vorgehensmodelle im Spannungsfeld von Softwareprozessen der Standardsoftwareeinführung und der Softwareneuentwicklung.

2.1 Softwareprozess

Im Folgenden werden die Begriffe Softwareprozess, Softwareprozessmodell und Vorgehensmodell näher beleuchtet. Dabei soll zuerst ein Begriff des Softwareprozesses gefunden werden, unter den sowohl Softwareneuentwicklungsprozesse als auch Prozesse zur Einführung von Standardsoftware fallen. Dann wird auf Softwareprozessmodelle und Vorgehensmodelle eingegangen.
In den darauf folgenden Unterkapiteln wird dann der Begriff des Vorgehensmodells im Kontext der beiden betrachteten Softwareprozesse Softwareneuentwicklung (2.1.2) und

Standardsoftwareeinführung (2.1.3) betrachtet. Im letzten Unterkapitel wird der Begriff Analyse bzw. Analyse-Methode beleuchtet.
Für die Begriffe werden zunächst verschiedene Definitionen aus der Literatur betrachtet und anhand dieser eine Abgrenzung zum Begriffsverständnis dieser Arbeit entwickelt.

Tab. 2 gibt einen ersten Überblick über verschiedene Definitionen für Softwareprozesse.

1. A process is understood as a set of logically related activities which have to be executed by different people and in a certain order. [GRU94, S.251]
2. The primary goal of managerial processes is to provide an environment and services for the development of software which meets the product requirements and project goals. Knowledge about these product engineering processes can be captured in models in order to allow process engineering processes to improve them systematically. [ROV95, S.5]
3. A process defines *who* is doing *what when* and *how* to reach a certain goal. In software engineering the goal is to build a software product or to enhance an existing one. [JBR98, S.xviii]
4. The software process defines the way in which software development is organised, managed, measured, supported and improved (independently of the type of support technology exploited in the development). [DKW99, S.1]
5. Nach [PFL98, S.44/45] hat jeder Prozess folgende Charakteristik:
   - Er beschreibt alle Hauptprozessaktivitäten
   - Er benutzt Ressourcen, steht unter einer Menge von Einschränkungen (z.B. Zeitvorgabe) und produziert End- und Zwischenergebnisse.
   - Er hat eine Menge von Leitprinzipien, die das Ziel jeder Aktivität erklären.
   - Jede Prozessaktivität hat Eingangs- und Ausgangskriterien, so dass Ende und Anfang der Aktivität bekannt sind.
   - Die Aktivitäten sind in einer Sequenz organisiert, die angibt wann eine Aktivität relativ zu einer anderen durchgeführt wird.

Tab. 2: Definitionen für Softwareprozesse

Die meisten Quellen betrachten Softwareprozesse und deren Abbildung durch Prozessmodelle aus dem Blickwinkel eines bestimmten Forschungsbereichs. In [TOC95] werden unterschiedliche Forschungsgemeinschaften (research communities) ausgemacht, die unterschiedliche Sichtweisen auf Softwareprozesse haben:

- Software Engineering
- Information Systems Engineering
- Enterprise Modeling
- Business Process Reengineering
- Organizational Design
- Workflow Management
- Concurrent Engineering
- Computer Integrated Manufacturing

Die Fragestellung dieser Arbeit ist übergreifend und fällt sowohl in den Bereich Software Engineering, als auch in den Bereich der Wirtschaftsinformatik, die als vorwiegend deutsche Forschungsrichtung in der obigen Liste nicht auftaucht (sie überschneidet sich zum Teil mit

den Bereichen Business Process Reengineering, Information Systems Engineering, Organizational Design und Computer Integrated Manufacturing).
Gruhn liefert in [GRU94] eine Definition für Prozesse, die sowohl von der Forschung im Bereich Softwareentwicklung, als auch von Bereichen, die er „business organization and office automation“ nennt, getragen wird (Tab. 2, Nr.1). Diese Definition hat den Nachteil, dass sie Software in keiner Weise erwähnt. Andere Definitionen definieren Softwareprozesse, die aber stets nur Softwareentwicklung betreffen (Tab. 2, Nr. 4). Gruhns Definition soll deswegen hier auf Software eingeschränkt werden. Interessant sind alle Prozesse, die irgendeinen Abschnitt im Lebenszyklus von Software betreffen. Im Folgenden soll unter Software-Prozess verstanden werden:

> Ein Software-Prozess ist eine Menge von geordneten, logisch in Beziehung stehenden Aktivitäten, die von verschiedenen Personen ausgeführt werden. Der Prozess beinhaltet mindestens einen Abschnitt im Lebenszyklus von Software.

Die logische Verwandtschaft unter den Aktivitäten soll hier insbesondere bedeuten, dass Aktivitäten Ergebnisse an andere Aktivitäten liefern können bzw. auf Ergebnisse von Aktivitäten aufbauen. Ein Abschnitt im Lebenszyklus von Software, soll in dieser Arbeit insbesondere die Entwicklung und Einführung von Software meinen, in anderen Kontexten wäre aber z.B. auch Wartung oder gar Entsorgung von Software denkbar.

In dieser Arbeit wird durch die Wahl der beiden Softwareprozesse Standardsoftware-einführung und Softwareneuentwicklung ein Spannungsfeld aufgebaut, in das andere Prozesse, wie die Erstellung von Software auf Basis von Komponenten oder Kombination von verschiedenen Standardprodukten, eingeordnet werden können. Die betrachteten extremen Randprozesse sind also die völlige Neuentwicklung auf der einen Seite und die Einführung einer "off-the-shelf"-Software, die kaum noch konfiguriert werden muss, auf der anderen Seite. Durch die Wahl der das Spannungsfeld aufziehenden Randprozesse, sollen die methodischen Unterschiede bei der Analyse besonders klar deutlich werden. Dabei ist zu beachten, dass die Prozesse real nie in dieser extremen Form auftreten.

Von Interesse sind weiterhin nur Softwareprozesse, die sozial eingebettete Software betrachten und übergreifende Aktivitäten zwischen verschiedenen Arbeitsplätzen unterstützen wollen, also keine Einzelplatzanwendungen (Stand-Alone-Textverarbeitungen etc.). Auf der Seite der Standardsoftware kommen hier im Prinzip alle Arten von Standard-Groupware, CSCW-Standardsoftware und betriebswirtschaftlicher Standardsoftware in Frage. Im Verlauf der Arbeit hat sich aber herausgestellt, dass betriebswirtschaftliche Standardsoftware dem „off-the-shelf“-Prinzip am ehesten entspricht. Auch um den Aufwand zu beschränken konzentriert sich diese Arbeit deswegen auf betriebswirtschaftliche Standardsoftware, die mittlerweile vor allem unter dem Stichwort Enterprise Resource Planning (ERP) gehandelt wird.

Softwareprozessmodelle bilden bestimmte Aspekte von Softwareprozessen ab. Rombach und Verlage [ROV95, S.2ff] sehen Softwareprozessmodelle als einen Bereich des Software Know-Hows, der in Software Engineering Prozessmodelle und Non-Software-Engineering Prozessmodelle unterteilt werden kann (Abb. 3). Die Software Engineering Prozessmodelle befassen sich mit technischen Aspekten der Produktentwicklung (requirements engineering, design, coding, verification, integration, validation) und Managementaktivitäten (product management, project management, quality assurance, project data management). Process

Engineering Prozessmodelle beschreiben Aspekte der Prozessverbesserung durch zyklische Planung, beobachtende Messungen und Feedback. Non-Software-Engineering Prozessmodelle betreffen Geschäftsprozesse (banking, acquisition, traffic, plant management etc.) und soziale Prozesse (communication, training, supervising etc.).

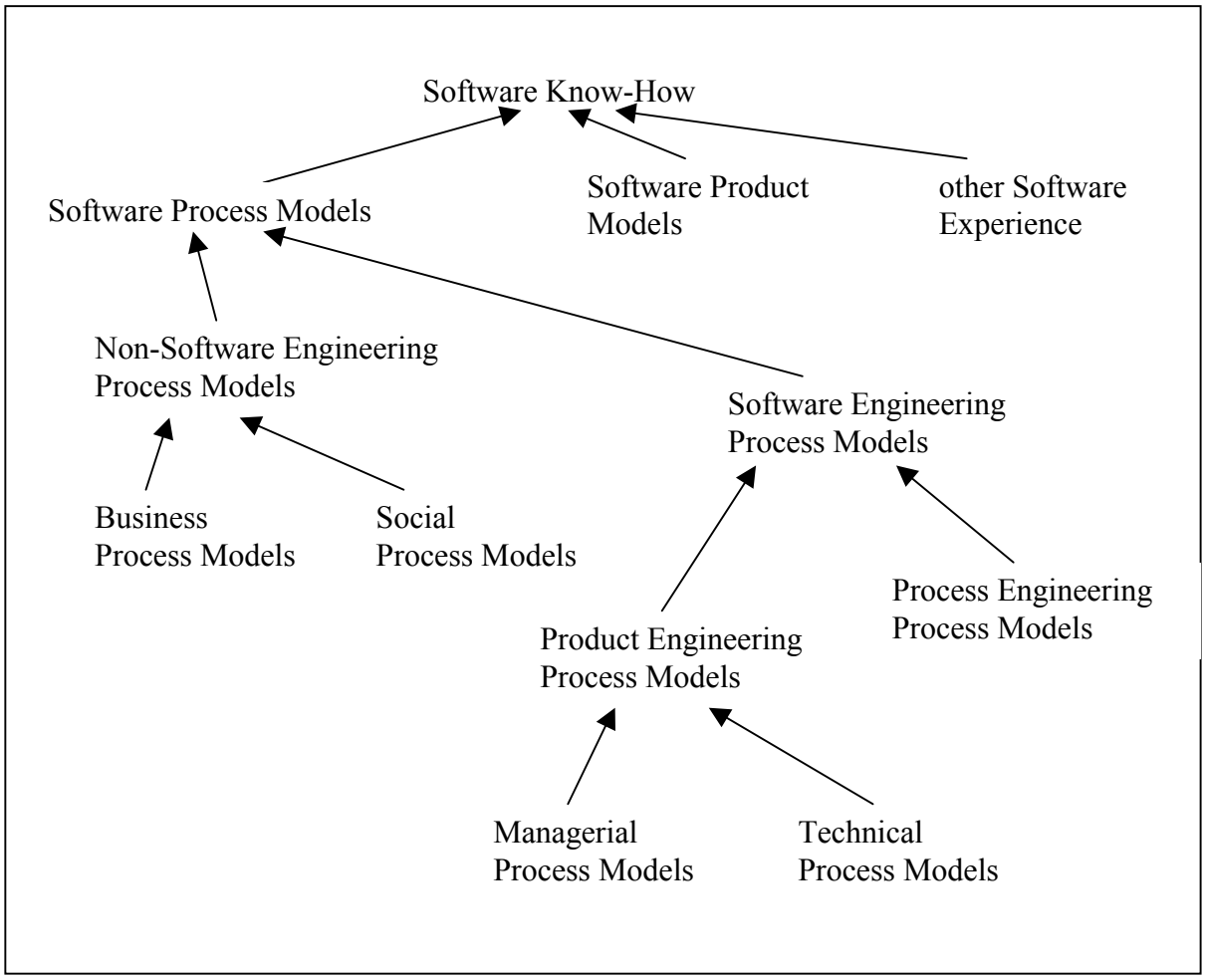


Abb. 3: Software Prozess Modelle nach [ROV95, S.3]

In der Software-Technik wurde im Laufe der Jahre erkannt, dass sich eine rein ingenieurmäßige Entwicklung von Software nicht bewährt und Softwareentwicklung nicht von Aspekten des Non-Engineering-Bereichs zu trennen ist. Immer mehr wurde auch der Non-Engineering-Bereich integriert: “Engineering and nonengineering-related processes are highly interrelated …” [ROV95, S.3].

1. In [CHR94, S.119] ist ein Prozessmodell ein Tupel PM = <A, R, i, o, ...>
   A: set of activity classes. An instance is represented by a_i
   R: set of result classes. An instance is represented by r_i
   i: $i(r_i,a_j)$ defines the inputs r_i of activity a_j.
   o: $o(r_i,a_j)$ defines the outputs r_i of activity a_j.
2. Software process models should represent as much as possible of a project, and thereby support as many project members as possible. [VER94, S.123]
3. A software process model is a set of high level goals and a description of how to decompose these goals into sub-goals. [GOD95, S.225]
4. "A software process model … describes how a class of processes is expected to be performed. … [It] represents one or more aspects. … these aspects consist of structure, resource consumption, development artifacts, and constraints. In addition, a process model can describe goals and effects of a process on its own properties and on the status of development artifacts and resources" [BRÖ95 ,S.10-11]

Tab. 3: Definitionen für Softwareprozessmodelle

Prozessmodelle zur Standardsoftwareeinführung sind in Abb. 3 nicht explizit eingeordnet. Zumindest reine „off-the-shelf"-Software wäre unter die Non-Software-Engineering Prozessmodelle einzuordnen, da hier keine Entwicklung der Software gefordert ist. Sobald der Extremfall der reinen „off-the-shelf"-Lösung aber verlassen wird, kommen auch Software-Engineering-Aspekte hinzu. Besonders interessant für Standardsoftwareeinführung ist auch der Bereich der Geschäftsprozessmodelle. Hierzu gehören z.B. auch Modelle zur Geschäftsprozessgestaltung (z.B. Business-Process-Reengineering Modelle).

2.1.1 Vorgehensmodelle

Vorgehensmodelle sind Softwareprozessmodelle, die die Durchführung eines Softwareprozesses unterstützen wollen. Andere vorstellbare Softwareprozessmodelle wären z.B. reine Erklärungsmodelle, die kein bestimmtes Vorgehen unterstützen wollen. Korrekter aus dem Englischen übersetzt, wo der Begriff Life-Cycle Models benutzt wird, wäre ein Begriff wie Lebenszyklus-Modell. Dieser spricht den Lebenszyklus einer Software von der Entwicklung bis zur Wartung an. Auch in diesem Fall wären andere Softwareprozessmodelle denkbar, die nicht den Lebenszyklus einer Software modellieren wollen. In dieser Arbeit soll jedoch der Begriff Vorgehensmodell benutzt werden. Für Vorgehensmodelle gibt es zahlreiche Definitionen (Tab. 4).

1. Life-cycle models describe activities over a project's lifetime on a high level of abstraction. They divide a project into several phases and describe temporal relationships. … development process models are more detailed and cover more aspects than life-cycle models. [ROV95, S.10-11]
2. Ein … Vorgehensmodell regelt den Ablauf des Lösungsprozesses. Es unterteilt ihn in überschaubare Abschnitte und soll dadurch eine schrittweise Planung, Durchführung, Entscheidung und Kontrolle ermöglichen. [POB93, S.17]
3. Vorgehensmodelle … dienen zur Benennung und Ordnung von produktbezogenen Aktivitäten bei der Softwareentwicklung. [FLZ97, S.657]
4. Vorgehensmodelle beschreiben in idealisierender und von Details abstrahierender Weise den Software-Entwicklungsprozess, die dabei auszuführenden Tätigkeiten und die zu erbringenden Ergebnisse. [HMF92, S.104]

Tab. 4: Definitionen für Vorgehensmodelle

Die hier zu betrachtenden Vorgehensmodellen ist gemeinsam, dass sie einen bestimmten Softwareprozess unterstützen wollen. Dazu gliedern sie den Prozess in eine Anzahl abstrakter Aktivitäten (respektive Phasen oder Workflows), die wiederum in Unteraktivitäten eingeteilt sein können. Die Aktivitäten sind durch Maßnahmen zur Einführung einer bestehenden oder neu zu schaffenden Software beschrieben. Die Maßnahmen können außerdem mit Organisationsgestaltung einhergehen. Die Vorgehensmodelle gehen von einem bei Prozessbeginn gegebenen Untersuchungsfeld oder Problemfeld aus, das eventuell noch weiter abgegrenzt werden muss. Auch Anforderungen sind zum Teil bereits gegeben, werden zum Teil aber auch aus dem Untersuchungsfeld gewonnen. Der Gegenstandsbereich von Analyse-Aktivitäten bezieht sich dabei auf das Untersuchungsfeld, aber auch auf bereits erstellte Beschreibungen bzw. Modelle des Untersuchungsfeldes (Tab. 5). Der Gegenstandsbereich wird durch das Lesen und Betrachten von Modellen (Dokumenten, Diagrammen etc.) sowie Durchführung von Interviews, Workshops und Umfragen erkundet. Dabei werden bestimmte Sichten eingenommen bzw. das Vokabular einer bestimmten Modellierungssprache angewendet (z.B. UML). Die Betrachtung des Gegenstandsbereichs unter diesen Sichten muss auch beachten, dass sich der Gegenstandsbereich verändert (z.B. wurde ein neuer Prototyp erstellt oder Anwender haben neue Wünsche). Deswegen wird in allen Softwareprozessen eine iterative bzw. zyklische Durchführung von Analyse-Aktivitäten berücksichtigt. Die durch Anwendung der Sichten entstehenden Modelle werden dokumentiert (Text, Diagramm, Grafik), entstehen aber auch zum Teil nur in den Köpfen der Beteiligten (Kognitive Karten [BAR96, S.167], Verständnis [KEI89, S.124] etc.). Von den Sichten bzw. Modellierungssprachen wird außerdem Durchgängigkeit in Bezug auf den Gesamtprozess gefordert.

Prozess	Gegenstandsbereich der Analyse
RUP	In der *Analyse des Problemfeldes* nimmt der Systemanalytiker in enger Zusammenarbeit mit dem Projektmanager die Anforderungen *von allen externen Beteiligten* des Projektes auf: Stakeholder, Endanwender, Kunden.
Helmut Balzert	*Beschriebene Anforderungen* sind auf Vollständigkeit, Konsistenz, Eindeutigkeit und Durchführbarkeit zu *analysieren*.
IBSIS	In der Analyse müssen die Mitarbeiter die *Ist-Prozesse analysieren*, um ein Modell der Realität zu erstellen. Es dürfen nicht nur Daten, Funktionen und Organisationseinheiten beschrieben werden, sondern es müssen auch die *Regeln und Prinzipien analysiert* werden, die zu der Ist-Situation geführt haben.
IMG	In der *Analyse der Ausgangsituation* werden die strategischen Anforderungen die generellen Rahmenbedingungen, die Projektziele und der Handlungsbedarf definiert.

Tab. 5: Gegenstandsbereich der Analyse in Softwareprozessen (Beispiele)

Im Folgenden sollen die Vorgehensmodelle der das Spannungsfeld aufziehenden Softwareprozesse Standardsoftwareeinführung und Softwareneuentwicklung näher betrachtet werden.

2.1.2 Vorgehensmodelle der Softwareentwicklung

In der Softwareentwicklung stehen sich Vorgehensmodelle gegenüber, die entweder das Produkt oder den Prozess der Entwicklung in den Mittelpunkt stellen.
Für Helmut Balzert steht bei der Softwareentwicklung das Produkt im Vordergrund: „Die Software-Entwicklung hat die Aufgabe, aus einem geplanten Produkt ein fertiges Produkt zu entwickeln, das die geforderten Qualitätseigenschaften besitzt. ... Eine Entwicklung läuft nicht von selbst ab, sondern Software-Management ist erforderlich, um den Entwicklungsprozess zu planen, zu organisieren, zu leiten und zu kontrollieren.“ [BALZ96, S.39]
Für [FLZ97, S.646] bezeichnet „Softwareentwicklung ... die Gesamtheit aller Aktivitäten, die zu einem Softwaresystem im Einsatz führen. Die Softwaretechnik befasst sich damit, einzelne Aktivitäten und ihre Ergebnisse zu identifizieren und zu ordnen.“ Hier steht also der Prozess der Entwicklung im Vordergrund. Je mehr bei Softwareentwicklung über die Aktivitäten reflektiert wird, umso mehr verschwimmen die Grenzen zwischen Softwareentwicklung und Softwaretechnik. In der Tat scheinen viele Autoren diese Begriffe auch gleichzusetzen. Sie betten ihre Methoden und Vorgehensweisen gleich in den Forschungsbereich Softwaretechnik bzw. Software-Engineering ein, ohne den Begriff Softwareentwicklung zu definieren. Betrachtet man Definitionen für die Forschungsbereiche Software-Engineering bzw. Softwaretechnik, so erscheint es nicht ungerechtfertigt, diese auch als Definition für Softwareentwicklung zu sehen.
Der Informatik-Duden definiert Software-Engineering folgendermaßen:

> Anwendung von Prinzipien, Methoden und Techniken auf den Entwurf und die Implementierung von Programmen und Programmsystemen. [DUD93, S.655]

[POB93, S.2] definiert:

> Software Engineering ist die praktische Anwendung wissenschaftlicher Erkenntnisse für die wirtschaftliche Herstellung und den wirtschaftlichen Einsatz qualitativ hochwertiger Software. [POB93, S.3]

Für Helmut Balzert ist Software-Technik die

> zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen Software-Systemen. Zielorientiert bedeutet die Berücksichtigung z.B. von Kosten, Zeit, Qualität. [BALZ96, S.36]

Für [FLZ97, S.642] ist der Gegenstand der Softwaretechnik die Entwicklung großer Softwaresysteme, besonders Anwendungssysteme. Von zentraler Bedeutung ist die Komplexitätsreduktion. „Dies erfordert eine Architektur als System von miteinander verknüpften Komponenten, wobei sowohl die Verständlichkeit des Gesamtsystems als auch die der einzelnen Komponenten anzustreben ist.“

Floyd und Züllighoven geben eine Beschreibung von Softwaretechnik der an Umfang und Prägnanz nichts hinzuzufügen ist:

> „... um fachliche Aufgaben in einem oder mehreren Anwendungsbereichen zu erledigen... werden vorhandene oder neue Problemlösungsstrategien durch Software realisiert. Sie modelliert einen Gegenstandsbereich der realen Welt und orientiert sich an einem Einsatzkontext, der fest sein kann (dedizierte Systeme) oder allgemein gehalten wird (Standardsoftware). Anwendungssoftware dient der Steuerung von technischen Prozessen oder der Unterstützung von Arbeitsprozessen. Bei ihrer Entwicklung ist auch die Gestaltung der Interaktion zwischen den Anwendungsfachleuten und dem eingesetzten Anwendungssystem zu beachten. ... Softwaresysteme werden professionell durch Entwicklerteams unter Einsatz von Methoden und Werkzeugen hergestellt. Dabei sind Aspekte der Arbeitsteilung und der Zusammenarbeit zu berücksichtigen. Da meist die Entwickler und die Anwender verschiedene Gruppen sind, findet wechselseitiges Lernen statt. Unterschiedliche Ziele, Interessen, Kenntnisse und Erfahrungen beeinflussen den Entwicklungsprozess. Daher sind moderierte kommunikative Arbeitsformen wichtig. ... Organisation und Management von Projekten setzt ein tragfähiges Verständnis des Softwareentwicklungsprozesses, seiner Teilaufgaben und deren Abhängigkeiten voraus, das durch Vorgehensmodell, Arbeitskonventionen und Qualitätssicherungsmaßnahmen umgesetzt wird.“ [FLZ97]

Als Vorgehensmodelle für Softwareentwicklungsprozesse gibt es verschiedene Modellarten, die sich in Phasenmodelle, Spiralmodelle oder zyklische Modelle unterteilen lassen.

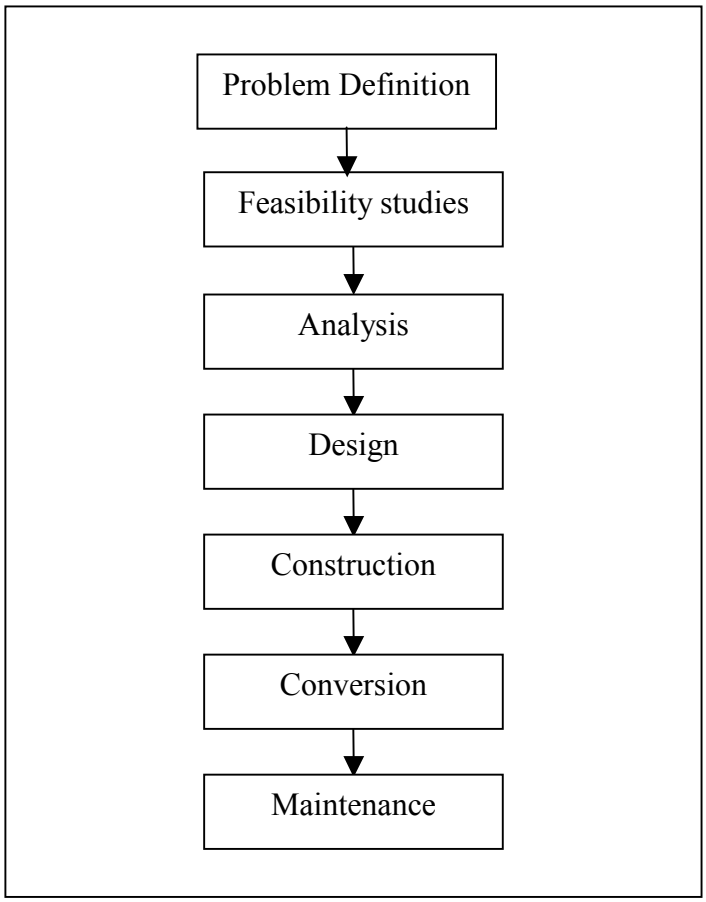


Abb. 4: Phasenmodell Softwareentwicklung [KEN92]

Die linearen Phasenmodelle (z.B. Abb. 4) teilen die Entwicklung in unterschiedliche Phasen, die benannt und standardisiert sind. Sie reichen von der Stufe, in der ein System angedacht wird, über die Analyse bis zur Implementierung und Wartung. Dabei sollte eine Aktivität vor dem Beginn der nächsten abgeschlossen sein. Einzelne Modelle unterscheiden sich in Zahl und Abgrenzung der Phasen. Jede Phase wird sequentiell durchlaufen und produziert bestimmte Artefakte (Anforderungen, Design, Quellcode etc.). Das Ergebnis des Prozesses ist ein Softwareprodukt, das eingesetzt und gewartet wird. Durch die Gliederung der Softwareentwicklung in einzelne Aktivitäten erlauben es diese Vorgehensmodelle den Entwicklungsprozess besser zu managen und zu planen. Zwischenergebnisse (sog. Meilensteine) können zu bestimmten Terminen geplant werden. Wegen ihres linearen Ablaufs heißen die Phasenmodelle auch Wasserfallmodelle. Das Einhalten der Linearität ist aber in der Praxis ein großes Problem. „In Wirklichkeit überlappen sich die ... auszuführenden Tätigkeiten, und die Wechselwirkungen zwischen den Phasen sind viel komplexer als dies durch das sequentielle Modell zum Ausdruck kommt.“ [POB93, S.22]

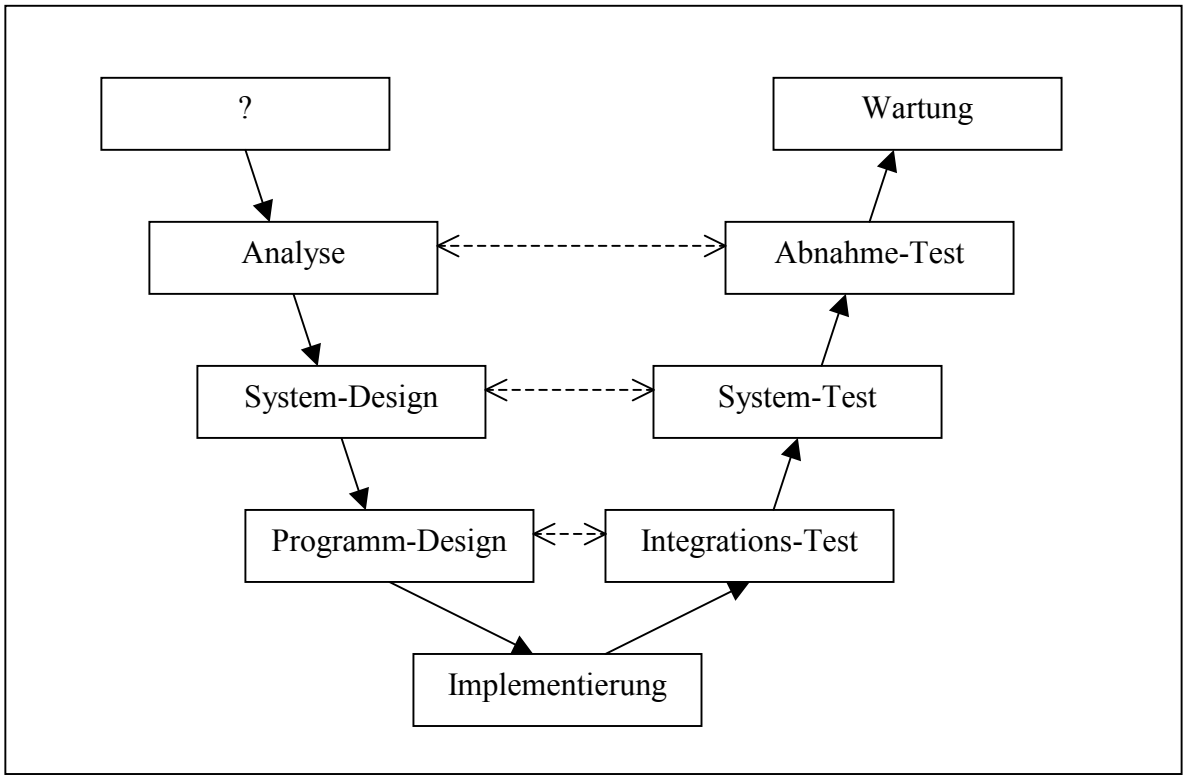


Abb. 5: V-Modell Softwareentwicklung [GMD92]

Das V-Modell [GMD92] ist eine Variation des Wasserfallmodells. (Es wurde ursprünglich im Auftrag des deutschen Verteidigungsministeriums entwickelt und schließlich als einheitlicher Standard für den gesamten öffentlichen Bereich übernommen.) Beim V-Modell werden die Analyse- und Design-Aktivitäten den Test-Aktivitäten zugeordnet (Abb. 5). Der Integrations-Test dient gleichzeitig der Verifikation des Programm-Designs, der System-Test dient der Überprüfung der Aspekte des System-Designs. Die Überprüfung der Anforderungen aus der Analyse wird in der Abnahme vom Kunden durchgeführt. Die Unterteilung von Aktivitäten in eine linke und eine rechte Seite impliziert, dass wiederholt auf der linken Seite begonnen werden kann, um auftretende Fehler zu beseitigen. Damit macht das V-Modell eine Iteration explizit, die im Wasserfallmodell nicht berücksichtigt wird.

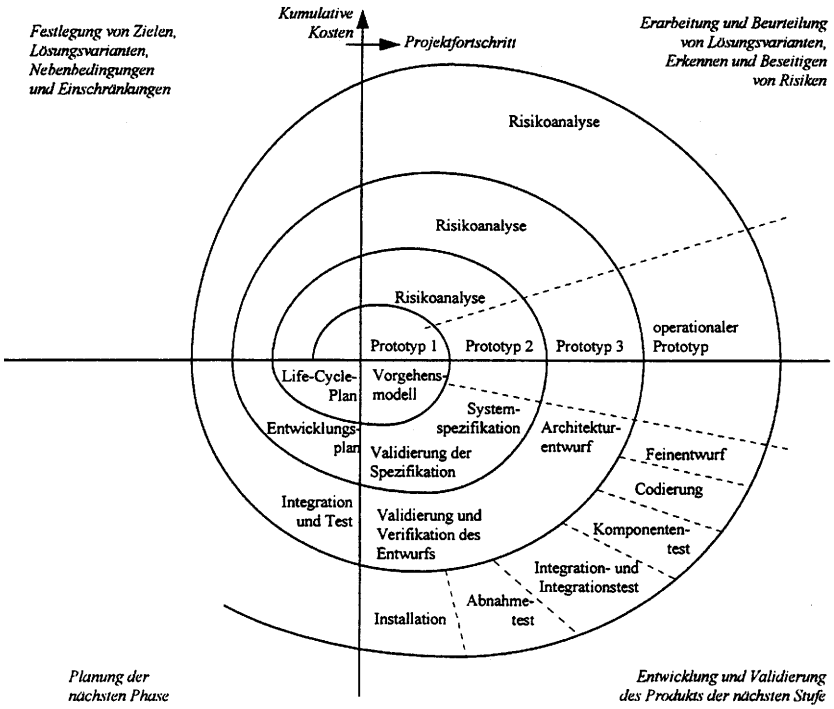


Abb. 6: Spiral-Modell nach Boehm [BOE88]

Auch Spiralmodelle (z.B. Abb. 6) sehen die Wiederholung von Aktivitäten nicht mehr als Ausnahme: Sie ordnen die Entwicklungsschritte in einem zyklisch angeordneten Prozess. Der Prozess beginnt im Mittelpunkt der Spirale und bewegt sich spiralförmig immer weiter vom Mittelpunkt weg. Nach jeder vollständigen Umdrehung wird durch dieselbe Folge von Aktivitäten eine neue Ausarbeitungsstufe erreicht. So bringen diese Modelle zum Ausdruck, dass sich die Durchführung von Aktivitäten zwar wiederholt, das Produkt aber (je weiter man sich vom Mittelpunkt der Spirale entfernt) kontinuierlich ausgebaut und verbessert wird.

Zyklische Modelle (z.B. STEPS [FRS89]) betrachten die Software nicht mehr als ein Produkt, welches kontinuierlich ausgebaut wird, sondern als Folge von Versionen. Die Entwicklung besteht aus einer Folge von Zyklen, in denen, aufbauend auf der letzten Version eine neue Version erstellt und eingesetzt wird. Herstellung und Einsatz sind verschränkt (Prototyping). Die Phasen der Entwicklung werden pro Zyklus inkrementell durchlaufen.

2.1.3 Vorgehensmodelle der Standardsoftwareeinführung

Standardsoftware unterscheidet sich von Individualsoftware dadurch, dass sie mehrfach in gleicher Form vertrieben bzw. verwendet wird. Dazwischen kann Komponentensoftware eingeordnet werden, bei der Teile wiederverwendet werden können und andere Teile bzw. die Zusammensetzung der Komponenten individuell auf ein Unternehmen abgestimmt werden muss. Standardsoftware kommt sozusagen „off-the-shelf“, eine Eigenschaft, die in dieser Arbeit besonders betont werden soll. Von Interesse ist hier außerdem nur Anwendungssoftware, die eine fachliche Anwendung unterstützt und nicht Systemsoftware, die die Grundfunktionalität des Systems gewährleistet. Desweiteren werden keine Einzelplatzanwendungen betrachtet. Mit dem Begriff Standardsoftware sind in der neueren Literatur meistens Programme gemeint, die einen möglichst großen Bereich betriebswirtschaftlicher Belange eines Unternehmens abdecken. Ein neuerer Begriff für diese

Produkte ist Enterprise Resource Planning Software (ERP). Hierzu zählen z.B. Standardsoftwareprodukte der Firmen SAP, Baan, Oracle, Peoplesoft, PSI, SSA, Marcam, JDEdwards, usw. [HAN96, S.210]. Dass in dieser Arbeit vor allem ERP-Produkte im Vordergrund stehen liegt auch an deren Größe. Für kleinere Standardsoftwareprodukte wird der Prozess der Einführung als vergleichsweise unproblematisch angesehen. Das spiegelt sich in der Literatur dadurch wieder, dass für kleinere Standardsoftwareprodukte keine Prozessmodelle angeboten werden. Bei ERP-Software gab und gibt es dagegen häufig Schwierigkeiten mit der Einführung und dementsprechend wird auch der Software-Prozess zur ERP-Einführung immer stärker in den Blick genommen.
Für den Begriff ERP-Software gibt es wenige Definitionen, die ohne Aufzählung von konkreten Systemen auskommen:

> "ERP software can be tentatively and crudely defined as customizable, standard applications software that includes integrated business solutions for the core processes and the main administrative functions of an enterprise." [GCT99, S.1039]

oder

> „Enterprise Resource Planning (ERP) systems are integrated enterprise-wide standard information systems that automate all aspects of an organisations' business processes." ERP systems "... prescribe information blueprints of how organisation's business processes should operate." [HLG99, S.273]

Im folgenden soll unter Standardsoftware deswegen die Definition aus [TEW98, S.14] übernommen werden:

> Standardsoftware ist ganzheitlich betriebswirtschaftliche Software, in der die Planung und Steuerung der Aufgabenbereiche Konstruktion, Vertrieb, Einkauf, Teilefertigung, Montage, Versand, Controlling und Personalverwaltung in ihrer Gesamtheit abgedeckt ist. Dabei steht die unternehmensweite und übergreifende Funktionalität im Vordergrund.

Die Problematik der Einführung von Standardsoftware ist wesentlich jünger als die der Softwareentwicklung, welche die Entwicklung großer Standardsysteme erst möglich machte. Die Literatur orientiert sich deswegen auch weitgehend an dem System, das lange Zeit Vorreiter war und fast Synonym für Standardsoftware stand: SAP R2 bzw. R3. Auch Literatur, die von diesen Systemen abstrahieren will, ist dies noch anzusehen.
Als Definition der Einführung von Standardsoftware findet man:

> Unter Standardsoftwareeinführung wird ... das Bündel aller Maßnahmen verstanden, die im Vorfeld und bis zur produktiven Nutzung eines für einen anonymen Markt entwickelten Anwendungssystems durchgeführt werden müssen. [HRS99, S.1]

Obwohl einige Vorgehensmodelle (vgl. Tab. 6) angeboten und in der Literatur beschrieben werden, findet man solche expliziten Definitionen für die Einführung von Standardsoftware kaum.

Autor / Anbieter	System	Vorgehensmodell
Barbitsch [BAR96]	unabhängig	IBSIS-Einführung
FIR [SCH98]	unabhängig	Aachener PPS-Modell
Kirchmer [KIR96]	unabhängig	GES
Wildemann [WIL90]	unabhängig	CIM-Einführung
Heinrich, Burgholzer [HEB90]	unabhängig	Systemplanung
BaaN	Triton	BaanWise
IBM	R2, R3	ISIM
IDS	R3	ASAP
PROMATIS	Oracle Applications	Oracle AIM Advantage & INCOME
SAP	R2, R3	IMW, IMG, AcceleratedSAP (ASAP)
Siemens Nixdorf	R3	R3 LIVE METHOD / Chestra

Tab. 6: Vorgehensmodelle für Standardsoftwareeinführung (s.a. [HRS99], [KIR96])

Eine übergreifende Diskussion von Vorgehensmodellen in Bezug auf die Einführung von Standardsoftware gab es, im Gegensatz zur ausführlichen Diskussion über Vorgehensmodelle zur Softwareentwicklung, bisher nicht.
Betrachtet man die angebotenen Vorgehensmodelle, so findet ein Wandel von linearen Phasenmodellen (vgl. Abb. 12) zu iterativen und zyklischen Modellen mit Prototyping statt (Abb. 7).

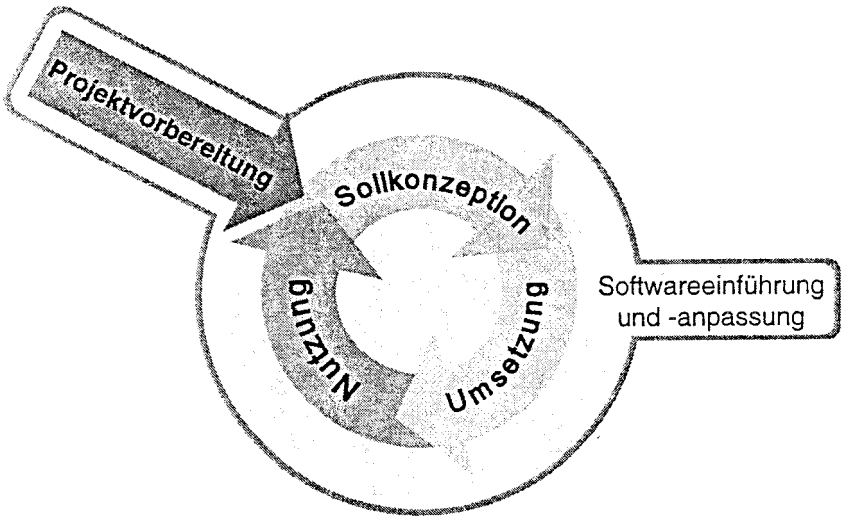


Abb. 7: Generalisiertes Vier-Phasen-Modell für Standardsoftwareeinführung [HRS99, S.17]

Hansen beschreibt einen allgemeinen Rahmen für Vorgehensmodelle der Standardsoftware-einführung. Für ihn sind die von Standardsoftwareanbietern genannten Vorteile von Standardsoftware lediglich Potentiale, die wesentlich vom Einführungsprozess abhängen. Gerade in den Anfangsphasen müsse das betriebswirtschaftliche Interesse im Vordergrund stehen. „Der Nutzen einer Standardsoftware-Einführung resultiert primär aus der Reorganisation des Unternehmens." [BAR96, Geleitwort S.5] Grob könne man ein Neuorganisations- und Standardsoftware-Einführungsprojekt in drei Teilschritte zerlegen:

1. Neuorientierung: Vorbereiten auf die Zukunft und Festlegen des Weges. Durchführung von Business Process Reengineering (BPR) und Suche nach geeignetem Informationssystem (Entscheidung ob Standardsoftware oder Neuentwicklung).
2. Standardsoftware-Abgleich: Testlabor mit grobem Prototyp, der dann immer mehr verfeinert wird.
3. Echtbetrieb

In [HRS99] fassen die Autoren vier Vorgehensmodelle (Anbieter: Baan, SAP, Siemens Nixdorf, IDS Scheer) zur Einführung von Standardsoftware zu einem generalisierten Vier-Phasen-Modell zusammen (Abb. 7):

> „Im Rahmen der Projektvorbereitung erfolgt im wesentlichen die Festlegung der Ziele des Einführungsprojektes, eine Schätzung von Zeit-, Personal- und Budgetbedarf und die Etablierung des Projektteams. Die sich anschließende Sollkonzeptionsphase beinhaltet die detaillierte Abbildung von Geschäftsprozessen, Organisationsstrukturen und die Formulierung von Anforderungen an die einzuführende Lösung. Entsprechend der erhobenen Anforderungen werden in der Umsetzungsphase das Customizing des Systems betrieben, funktionale Erweiterungen programmiert, Tests durchgeführt, Berechtigungen eingerichtet und Anwender geschult. Die im Rahmen der Nutzungsphase auftretenden Probleme und entstehenden neuen Anforderungen führen zu einem neuerlichen Anstoß der Sollkonzeption. Parallel zu allen genannten Phasen läuft die Schulung der Beteiligten sowie das Projektmanagement." [HRS99, S.17]

2.1.4 Analyse-Methoden

Analyse bedeutet etymologisch soviel wie Auflösung, Zergliederung oder Untersuchung und war „... im Griechischen zunächst ein Terminus der mathematischen und philosophischen Methodenlehre (z.B. etwas auf die Bestandteile zurückführen, aus denen es zusammengesetzt ist). In der Neuzeit [gab es] dann [eine] Ausweitung der Bedeutung auf wissenschaftliche Untersuchung." [KLU89, S.27/28]
Es gibt zahlreiche Begriffe und Definitionen für Analyse (Abb. 8).

1. SA: “Analysis is the study of a problem, prior to taking some action.” [DEM79, S.4]
2. SADT: “Requirements definition is a careful assessment of the needs that a system is to fulfill. It must say why a system is needed, based on current or foreseen conditions, which may be internal operations or an external market. It must say what system features will serve and satisfy this context. And it must say how the system is to be constructed.“ [ROS79, S.366]
3. „The aim of domain engineering is to develop, together with stake-holders of, or in, the selected domain, a precise set of concordant descriptions of the domain, a set that the parties, developers and stake-holders, can agree upon.” [BJO97, S.5]
4. “Business Reengineering nach unserem Verständnis beschäftigt sich mit den einzelnen Abläufen im Unternehmen und versucht, diese für das Geschäft zu optimieren. ... Ziel des Business Reengineering ist es, die organisatorischen Abläufe eines Unternehmens neu zu gestalten.“ [BRH95, S.18]
5. Ist-Analyse: In einer Ist-Analyse müssen ein Verständnis über die Arbeitsabläufe der tätigen Personen erarbeitet und in einer übergeordnete Sichtweise die Kooperation der Teilbereiche einer Organisation betrachtet und verstanden werden. Dazu werden verschiedene Dokumenttypen benutzt. (Ist-Analyse für ein Krankenhaus [RAT97, S.11])
6. OBA: Analysis is the study and modeling of a given problem domain, within the context of stated goals and objectives. It focuses on the what a system is supposed to do, rather than how it is supposed to do it (which we consider the design aspects). In addition, it must embody the rule of traceability (why), which justifies the existence of a given result by tying it back to the stated goals and objectives. [RUG92, S.48]
7. OOA: Analyse ist die Untersuchung eines Anwendungsgebietes, die zur Spezifikation eines von außen beobachtbaren Verhaltens führt. Die Analyse definiert vollständig, greifbar und in sich stimmig alle erforderlichen Faktoren. Sie umfasst die funktionellen und quantitativen Betriebseigenschaften (wie z.B. Zuverlässigkeit, Verfügbarkeit, Leistung). Analyse ist der Prozess, in dem herausgefunden wird, was ein System können muss - was das System tun muss, um den Kunden zufrieden zu stellen. Wie das System implementiert wird, ist für den Analytiker nicht wichtig. [CY90]
8. OOA: Das Ziel einer Analyse ist, die wahren Wünsche und Anforderungen eines Auftraggebers an ein neues Softwaresystem zu ermitteln und zu beschreiben. Dazu werden in einem kontinuierlichen Prozess Modelle und Dokumente bzw. Prototypen erstellt, die auch für den Auftraggeber verständlich aufbereitet werden. [BAL99, S.8+9]
9. OOSE: The purpose of analysis is to obtain an understanding of the application; an understanding depending only on the system’s functional requirements. [JAC92, S.77]

Abb. 8: Analyse-Definitionen aus der Literatur

Nach dem Fremdwörterlexikon ist eine Methode eine „Art und Weise des Vorgehens“ [DWF84, S.193]. Für Bjørner ist eine Methode folgendes:

> „A method is a set of principles, techniques and tools for analysing problems and for selecting and applying techniques and tools in order efficiently to construct an efficient artefact.” [BJO97, S.3]

Diese Definition scheint für diese Arbeit wenig hilfreich, da sie selbst wieder viele ungeklärte Begriffe enthält, im besonderen Analyse. Besser geeignet erscheint diese Definition: Eine Methode ist „... ein System von Regeln, mit deren Anwendung man von einer Ausgangssituation zu einem gewünschten Ziel gelangt." [SCHW96, S.13]

Die sehr unterschiedlichen Vorstellungen und Ziele von Analyse führen dazu, dass eine Definition, die möglichst keine Sichtweise ausschließen will, sehr allgemein gefasst sein muss. Unter Analyse-Methode soll hier deswegen verstanden werden:

> Eine Analyse-Methode ist ein System von Regeln, bei dessen Anwendung ein gegebener organisatorischer Gegenstandsbereich aufgelöst, zergliedert und untersucht wird und ein Ergebnis erstellt wird (Abb. 9).

Wenn hier die Rede von einem Gegenstandsbereich ist, soll dabei nicht angenommen werden, dass man tatsächlich objektiv die Realität abbilden kann, sondern dass es sich um einen subjektiv wahrgenommenen Bereich handelt.

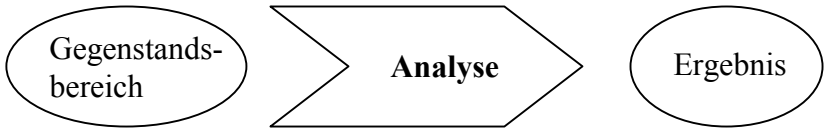


Abb. 9: Analyse

Die Einschränkung *organisatorischer* Bereich soll den Fokus auf die Analyse des Unternehmens bzw. der Organisation setzen, in die neue Software eingeführt werden soll (im folgenden Anwender-Organisation genannt). Ausgeschlossen aus der Betrachtung werden sollen Analyse-Methoden der Marktanalyse (z.B. Auswahl von Software) und der Analyse von Software (z.B. Tests). Nicht ausgeschlossen wird die Analyse von bereits erstellten Dokumenten bzw. Modellen eines organisatorischen Bereichs (z.B. Extraktion von Use-Cases aus einer textuellen Beschreibung).

2.2 Anforderungen einzelner Vorgehensmodelle

Im Folgenden werden für fünf Vorgehensmodelle die Anforderungen an Analyse-Methoden näher betrachtet. Die Vorgehensmodelle wurden nach Relevanz ausgewählt. Im einzelnen sind dies:

- STEPS (Softwaretechnik für Evolutionäre Partizipative Systemgestaltung)
- Vorgehensmodell nach Helmut Balzert
- Rational Unified Process (RUP)
- IMG von SAP (Implementation Management Guide)
- Vorgehensmodell nach Barbitsch (Im Folgenden IBSIS-Modell)

In den folgenden Unterkapiteln wird für jedes Modell zuerst eine kurze allgemeine Übersicht geliefert und sein Gesamtziel beschrieben. Dann wird bestimmt wie das Vorgehensmodell Analyse in den Gesamtprozess einordnet. Darauf werden zum einen die Vorgaben an die Analyse festgestellt und zum anderen die erwarteten Ergebnisse der Analyse. Am Ende jedes Unterkapitels gibt es eine kurze Zusammenfassung.

2.2.1 Evolutionäre Systementwicklung (STEPS)

Übersicht

STEPS (Softwaretechnik für Evolutionäre Partizipative Systemgestaltung) ist ein zyklisches Vorgehensmodell, das einen Methodenrahmen vorgibt, der durch verschiedene an evolutionäre Sicht angepasste Methoden ausgefüllt werden kann. „STEPS vertritt als grundlegende Sichtweise einen menschenzentrierten und anwendungsnahen Ansatz, der Softwareentwicklung als Lern- und Kommunikationsprozess begreift.“ [GRY96, S.2]

Gesamtziel

„Da von einem Zusammenspiel von Softwareeinführung und organisatorischer Veränderung ausgegangen wird, wird Software-Entwicklung als integrativer Teil einer übergreifenden Organisationsentwicklung gesehen.“ [FKRW97, S.14]. STEPS befasst sich insbesondere mit der Einbettung von Software in Arbeits- und Kommunikationsprozesse. Evolutionäre Systementwicklung soll insbesondere experimentelle Arbeitsformen in der Entwicklung etablieren, die Zusammenarbeit mit Benutzern verbessern und gebrauchsangemessene Software für den Einsatz entwickeln. STEPS beschränkt sich dabei nicht nur auf die reine Entwicklung, sondern nimmt auch die Anpassung bzw. Einführung von Standardsoftware in den Blick. Diese übergreifende Sicht wird auch durch den hohen Abstraktionsgrad des Methodenrahmens ermöglicht. In der Praxis hat es sowohl Entwicklungsprojekte, als auch Standardsoftwareauswahl- und Standardsoftwareeinführungsprojekte gegeben, die sich an STEPS orientierten.

„Technisches Kernanliegen der evolutionären Systementwicklung ist eine änderbare Softwarearchitektur, die die anwendungsfachlichen Gegebenheiten direkt widerspiegelt. Dies ist mit einer modularen, besser noch mit einer objektorientierten Konstruktionstechnik leistbar...“. [S.15]

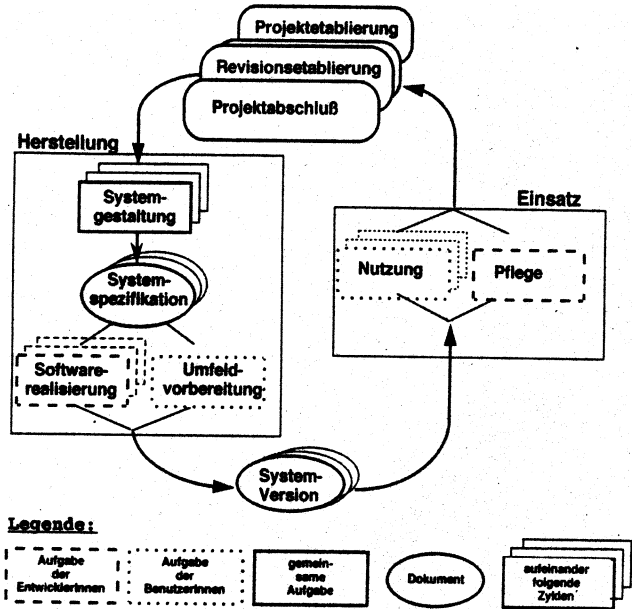


Abb. 10: Modellrahmen STEPS [FKRW97, S.14]

Einordnung von Analyse

Zur Erarbeitung des Zusammenhangs von Software-Entwicklung und Organisationsentwicklung werden in STEPS insbesondere Methoden der aufgabenbezogene Anforderungsermittlung verlangt [FKRW97, S.14].

Vorgaben

Neben der geforderten Einbettung der Software in Arbeits- und Kommunikationsprozesse beschreibt Keil-Slawik, wie die aufgabenbezogene Anforderungsermittlung in den Entwicklungsprozess einzubetten ist [KEI89]. „Verfahren und Techniken zur Anforderungsermittlung müssen ... sowohl aufgabenbezogen, kommunikationsunterstützend und verständnisfördernd, als auch im Hinblick auf die weiteren Entwicklungsschritte effizient nutzbar sein. Was jedoch diese Kriterien im einzelnen bedeuten, lässt sich nur aufgrund der Analyse praktisch durchgeführter Entwicklungsprojekte feststellen und bewerten.“ [KEI89, S.124]

Ergebnisse

„Anforderungen stecken den Rahmen für die Systementwicklung ab ... Wesentlich dabei ist der Zusammenhang zwischen Anforderungen an das DV-System und der Einpassung des Systems in das menschliche Arbeitshandeln.“ [KEI89, S.124] In der Anforderungsermittlung müssen folgende Anforderungen erarbeitet werden:

- Funktionelle Anforderungen (beschreiben das Ein-/ Ausgabeverhalten des DV-System, d.h. welche Eingaben das System erwartet, welche Listen erzeugt werden sollen usw.)
- Leistungsanforderungen (zur Verfügung stehende Ressourcen, optimale Ausnutzung der Betriebsmittel, Angabe von Mengengerüsten)
- Handhabungsanforderungen (betreffen das Zusammenwirken von Mensch und Rechner; z.B. GUI, Dialogführung, Fehler- und Ausnahmebehandlung etc.)
- Einbettungsanforderungen (berücksichtigen spezifische Gegebenheiten einer Organisation, die Randbedingungen für die Software-Entwicklung liefern, z.B. existierende Hardware und Software oder Verfahrensvorschriften, Standards, usw.)

Die Anforderungen müssen für die darauf aufbauende Systemgestaltung bereit gestellt werden. Außerdem „soll einerseits den Benutzern frühzeitig die Einflussnahme auf Entwurfsentscheidungen ermöglicht werden und andererseits das Verständnis der Entwickler dafür gefördert werden, ob und welche DV-Leistungen für eine angemessene Unterstützung der Arbeitsaufgaben zur Verfügung gestellt werden sollen.“ Dabei müssen Entwurfsentscheidungen explizit vollzogen werden und Kommunikation sowie differierende Sichten dokumentiert werden. „Für die Entwickler ist es darüber hinaus wichtig, dass die zu erstellenden Modelle und Dokumente die Entwicklungs- und Implementierungsarbeit erleichtern und nicht nur einen für ihre weitere Arbeit unbrauchbaren zusätzlichen Dokumentationsaufwand bedeuten.“ [S.124]

Gegenstandsbereich	Organisation, Arbeitsprozesse, Kommunikationsprozesse, Aufgaben
Vorgaben	• **aufgabenorientiert** • **kommunikationsunterstützend** • **verständnisfördernd** • **im Hinblick auf weitere Entwicklungsschritte effizient**
Methoden	**Aufgabenorientierte Anforderungsermittlung**
dokumentierte Ergebnisse	• **Anforderungen dokumentiert (Funktionelle, Leistungsanforderungen, Handhabungsanforderungen und Einbettungsanforderungen)** • **Entwurfsentscheidungen explizit gemacht** • **Kommunikation und unterschiedliche Sichten sind dokumentiert** • **Rahmen für Systementwicklung festgelegt**
„mentale“ Ergebnisse	• **früher Einfluss der Benutzer auf Entwurf** • **Verständnis der Entwickler für Angemessenheit** • **Implementierungsarbeit wird erleichtert**

Tab. 7: Analyse in STEPS

Zusammenfassung

Das Vorgehensmodell STEPS will einen allgemeinen Rahmen für verschiedenste Softwareprozesse geben, der den Menschen, Anwendungsnähe und evolutionäre Vorgehensweise in den Mittelpunkt stellt. Die Softwareentwicklung, -anpassung oder -einführung wird als Teil einer übergreifenden Organisationsentwicklung gesehen. Analyse soll insbesondere Arbeits- und Kommunikationsprozesse untersuchen. Im Ergebnis soll sie aufgabenbezogen, kommunikationsunterstützend und verständnisfördernd sein sowie weitere Entwicklungsschritte effizient vorbereiten.

STEPS hat im Vergleich zu den anderen hier betrachteten Vorgehensmodellen den höchsten Abstraktionsgrad. Im Gegensatz zum RUP (Kap. 2.2.3) und dem Vorgehensmodell von Helmut Balzert (Kap. 2.2.2), die viele etablierte Methoden der Software-Entwicklung zu einem Ansatz kombinieren, geht STEPS von wenigen Leitbildern aus, die einen ganzheitlichen Rahmen zur Einbettung von Methoden bieten. Die anderen Vorgehensmodellen liefern zwar „nach oben hin“ keine allgemeingültigen Leitbilder, schlagen aber konkrete Analyse-Methoden vor. STEPS lässt dagegen nach „unten hin“ die Wahl der Analyse-Methoden weitgehend offen. Auf der Abstraktionsebene von STEPS werden auch keine unterschiedlichen Anforderungen an Analyse-Methoden für Standardsoftwareeinführung oder Softwareneuentwicklung gestellt. Leitbild für die Auswahl von Analyse-Methoden ist die aufgabenorientierte Anforderungsermittlung. Floyd vermeidet aber die Verwendung des Begriffs Analyse.

Der Unterschied des evolutionären Ansatzes STEPS im Vergleich zu einer Produktentwicklung tritt besonders im Vergleich zu Helmut Balzerts Vorgehensmodell hervor. Alle Autoren der hier vorgestellten Vorgehensmodelle bemühen sich aber zumindest zu erwähnen, dass ein iteratives Vorgehen bei der Analyse notwendig ist. Zum Teil verhindern die Modelle aber eine Iteration durch die wasserfallartige Struktur (insbesondere die Gateways des IBSIS-Modells erinnern sehr an Meilensteine).

2.2.2 Softwareentwicklung nach Helmut Balzert

Übersicht

In seinem „Lehrbuch zur Software-Technik“ bietet Helmut Balzert ein umfangreiches Vorgehensmodell an, das an mehreren deutschen Universitäten als Lehrbuch für Software-Entwicklung dient [BALZ96]. Balzert berücksichtigt drei Ebenen der Software-Technik:

1. Produktebene
2. Arbeitsplatzebene
3. Unternehmensebene

Auf Produktebene werden vom Auftraggeber Anforderungen an das Produkt gestellt. Die Arbeitsplatzebene unterstützt dagegen die Sicht des Benutzers und wie er seine Arbeitsaufgaben lösen kann. Die Unternehmensebene befasst sich mit Geschäftsprozessen. „Da die Entscheidungen, die auf der Unternehmens- und der Arbeitsplatzebene getroffen werden, gravierende Auswirkungen auf die Software-Produktebene haben können, sind die Methoden, die im Vorfeld der eigentlichen Softwareentwicklung angewandt werden, auch für den Software-Ingenieur relevant.“ [BALZ96, S.43]
Dies gilt auch umgekehrt: "Da ein Unternehmen heute ohne Informations- und Kommunikationstechnik nicht bestehen kann, spielt letztendlich die Software eine entscheidende Rolle bei der Realisierung eines Unternehmensmodells." [BALZ98, S.691] Balzert sieht die Unternehmensmodellierung als ingenieurmäßige Tätigkeit. [S.718]

Gesamtziel

Das Vorgehensmodell behandelt die ingenieurmäßige und arbeitsteilige Entwicklung von umfangreicher Software auf Basis von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen. Dabei wird versucht die Ziele eines Software-Kunden zu erreichen. [BALZ96, S.36] Balzert stellt fest, dass in der Software-Technik die Denkmuster strukturiertes Paradigma, objektorientiertes Paradigma und wissensbasiertes Paradigma getrennt betrachtet werden. Er sieht diese Paradigma dagegen als sich gegenseitig ergänzend an und betrachtet sie in seinem Vorgehensmodell deswegen integriert: „Jeder Software-Ingenieur muss wissen, welches Paradigma für welches Problem oder Teilproblem am besten geeignet ist. Es geht nicht um ‚Entweder – Oder’ sondern ‚Sowohl als auch’“. [S.41]

Einordnung der Analyse

Balzert strukturiert Softwareentwicklung in fünf Phasen:

1. Planungsphase
2. Definitionsphase
3. Entwurfsphase
4. Implementierungsphase
5. Abnahme & Einführungsphase

Analyse-Methoden werden dabei in der Planungs- und der Definitionsphase gebraucht.
Die Arbeitsplatzebene und die Unternehmensebene betrachtet Balzert getrennt vom übrigen Vorgehensmodell im zweiten Band seines Lehrbuches [BALZ98].

Vorgaben

Balzert nennt eine Reihe von Dingen, die die Grundlage für die *Planungsphase* liefern sollen [BALZ96, S.56]: Trendstudien, Marktanalysen, Forschungsergebnisse, Vorentwicklungen, Kundenanfragen.
Die *Definitionsphase* basiert zum einen auf den „Vorgaben und Rahmenbedingungen aus der Planungsphase“ und zum anderen aus vagen, verschwommenen, unzusammenhängenden unvollständigen und widersprüchlichen Anforderungen. [BALZ96, S.93]

Bei der Unternehmensmodellierung unterscheidet Balzert in "vier wesentliche Blickwinkel: Daten, Funktionen, Geschäftsprozesse, Arbeitsplätze" [BALZ98, S.716]
"An die Konzepte und Notationen zur Unternehmensmodellierung ist insbesondere die Anforderung der guten Verständlichkeit zu stellen. ... Außerdem muss das Unternehmensmodell leicht änderbar sein, damit es an die wandelnden Anforderungen schnell angepasst werden kann. ... Da ein großer Teil des Unternehmensmodells durch Anwendungssoftware realisiert werden muss, müssen die Konzepte und Notationen kompatibel zu Konzepten und Notationen der Software-Entwicklung sein." [BALZ98, S.717]
"Ausgangspunkt für die Umgestaltung oder Neugestaltung eines Unternehmens ist eine dokumentierte Unternehmensvision. Da in der Regel die Basis für eine Umgestaltung ein existierendes Unternehmen ist, entwickelt sich eine Unternehmensvision in Wechselwirkung mit der Analyse des existierenden Unternehmens.“ [BALZ98, S.722]

Ergebnisse

Ergebnis der Planungsphase ist die Entscheidung über die weitere Vorgehensweise: „weitermachen oder beenden (Stop or go).“ [BALZ96, S.56] Ziel der Planungsphase ist zu prüfen, ob ein Produkt entwickelt werden soll. Dabei muss eine gezielte Ist-Aufnahme durchgeführt werden, wenn bereits ein Vorgängerprodukt vorhanden ist. Anschließend wird eine Ist-Analyse durchgeführt. Es werden die Hauptanforderungen festgelegt (Funktionen, Daten, Leistungen, Aspekte der Benutzungsschnittstelle, Qualitätsmerkmale), die softwaretechnische Realisierbarkeit geprüft und eine Aufwandsschätzung (Zeit, Wirtschaftlichkeit) durchgeführt. Dann werden alternative Lösungsvorschläge (z.B. Standardsoftware) geprüft und geschaut, ob das benötigte Personal vorhanden ist. Schließlich werden die Risiken evaluiert. „Die Ergebnisse dieser Tätigkeiten münden in eine Durchführbarkeitsstudie (feasiblity study)...“, die aus Lastenheft, Projektkalkulation und Projektplan besteht.
Das Lastenheft soll „eine Zusammenfassung aller fachlichen Basisanforderungen, die das zu entwickelnde Software-Produkt aus der Sicht des Auftraggebers erfüllen muss“ beinhalten. Mit Basisanforderungen ist eine „bewusste Konzentration auf die fundamentalen Eigenschaften des Produktes und ihre Beschreibung auf einem hinreichenden Abstraktionsniveau“ gemeint. Das Lastenheft ist an die Auftraggeber, sowie die Auftragnehmer adressiert. Es beschreibt das „Was“ nicht das „Wie“. Das Lastenheft hält sich an ein „vorgegebenes, standardisiertes, grobes Gliederungsschema mit festgelegten Inhalten“. Es beschreibt die einzelnen Anforderungen in verbaler Form. Das Lastenheft ist außerdem das zeitlich erste Dokument, das die Anforderungen an das neue Produkt beschreibt. Der Umfang sollte auf wenige Seiten beschränkt werden. [BALZ96, S.57/85]
Die geforderte Aufwandsschätzung soll vor allem eine Vorstellung von der Anzahl gebrauchter Mitarbeitermonate und somit der geschätzten Personalkosten erbringen. In die Aufwandsschätzung können dabei, wenn vorhanden, Erfahrungswerte aus anderen Projekten einfließen.
Auf die erwarteten Ergebnisse von Marktanalyse und Ist-Analyse geht Balzert nicht genauer ein.

Das Ergebnis der Definitionsphase sind „vollständige, konsistente, eindeutige und durchführbare Produktanforderungen“. Die beschriebenen Anforderungen sind auf diese Qualitätsziele hin zu analysieren. Diese Produktdefinition legt qualitative und quantitative Eigenschaften des Produkts aus Sicht des Auftraggebers fest. Die Produktdefinition dient später auch der Abnahme des fertigen Produkts. „Vollständigkeit beschreibt den Grad, in dem das Produkt dem Benutzer alle notwendigen Funktionen und Daten selbst zur Verfügung stellt, um die geforderten Produktziele zu erreichen.“ [S.94] „Konsistenz beschreibt den Grad, in dem die definierten Anforderungen untereinander widerspruchsfrei sind.“ [S.95] „Eindeutigkeit beschreibt den Grad, in dem die definierten Anforderungen genau eine Interpretation erlauben.“ [S.95] Technische „Durchführbarkeit kann man durch Prototyp-Entwicklung überprüfen.“ [BALZ96, S.95]
Das Ergebnisdokument der Anforderungsdefinition ist ein Pflichtenheft. Es „... enthält eine Zusammenfassung aller fachlichen Anforderungen, die das zu entwickelnde Software-Produkt aus der Sicht des Auftraggebers erfüllen muss.“ [S.104] Adressaten sind Auftraggeber, Auftragnehmer repräsentiert durch den Projektleiter und die Systemanalytiker, Entwerfer, Qualitätskontrolle, Benutzerrepräsentant oder ausgewählte potentielle Benutzer. Das Pflichtenheft muss „so abgefasst sein, dass es als Basis eines juristischen Vertrages dienen kann. Das Pflichtenheft stellt also die vertragliche Beschreibung des Lieferumfanges dar.“ [S.105] Es muss außerdem in einer vorgegebenen, standardisierten Form und einem groben Gliederungsschema mit festgelegten Inhalten strukturiert sein, damit es gut lesbar und vergleichbar ist. Sprachlich sollte es detailliert sein und mit Numerierung einzelner Anforderungen, auf die sich dann in späteren Dokumenten bezogen werden kann.
„Die notwendigen Anforderungen müssen in ausreichender Detaillierung und ausreichendem Präzisierungsgrad beschrieben werden. Der Funktions-, Daten- und Leistungsumfang muss aus Sicht des Benutzers bzw. Auftraggebers auf hinreichendem Abstraktionsniveau vollständig beschrieben sein.“ [BALZ96, S.106]
„Das Pflichtenheft ist das erste Dokument, das nach Abschluss der Planungsphase erstellt wird. Ergibt sich die Notwendigkeit, Anforderungen im Pflichtenheft zu ändern ..., dann sind diese Änderungen vom Auftraggeber schriftlich zu bestätigen.“ [S.106] „Das Erstellen eines verbalen Pflichtenheftes reicht nicht aus. Es muss ein semiformales bzw. formales Produktmodell erstellt werden.“ Nur dies erzwingt eine fachliche Problemlösung. [BALZ96, S.112]

„Die systematische Vorgehensweise, um Anforderungen in einem iterativen Prozess zu ermitteln, bezeichnet man als Systemanalyse (requirements engineering).“ [S.92] Dazu müssen die Anforderungen ermittelt, festgelegt, beschrieben, analysiert, animiert, simuliert, ausgeführt und verabschiedet werden. Dies muss iterativ mehrmals wiederholt werden.
Der Gegenstandsbereich zur Ermittlung der Anforderungen wird in Gesprächen erkundet. „Je nach Situation kommen ein oder mehrere Gesprächspartner in Frage: der Auftraggeber, die Marketingabteilung, die Fachabteilung, der Benutzer, ein oder mehrere Repräsentanten der Endbenutzer.“ [S.92] Desweiteren wird bei bereits vorhandenen Software-Systemen eine Ist-Analyse durchgeführt, die Schwachstellen des vorhandenen Systems aufdecken soll. „Diese Mängel können dann konstruktiv bei der Formulierung der neuen Anforderungen berücksichtigt werden.“ [BALZ96, S.93]
Der Systemanalytiker kümmert sich aktiv um die Ermittlung der Anforderungen. Zum Teil erhält er schriftlich formulierte Anforderungen. Weitere Informationen erhält er durch Interviews und Befragungen. Seine Aufgabe ist es dann, „ein vollständiges, konsistentes und eindeutiges Produkt-Modell zu entwickeln. Die eingesetzten Methoden und Werkzeuge helfen ihm, den Gesprächspartnern die richtigen Fragen zu stellen. Die Fragen wiederum

helfen den Gesprächspartnern, ihre Vorstellung von dem zu erstellenden Produkt weiterzuentwickeln.“ [S.94]
Die Anforderungen müssen dann festgelegt, beschrieben, systematisch gruppiert und klassifiziert werden. „Sie sind in unbedingt notwendige und wünschenswerte zu unterteilen. Die Funktionen, die Daten, die Leistungen, die Benutzungsschnittstelle und die Qualitätsziele sind festzulegen. Die Entwicklungs- und Zielumgebung sind zu bestimmen.“
Die in der Definitionsphase entstehende Produktdefinition kann aus verschiedenen Teilprodukten bestehen. Nach Balzert müssen vier verschiedene Sichten und ihre Zusammenhänge beschrieben werden: Daten, Funktionen, Dynamik, Benutzungsoberfläche. Die Sichten können je nach Produkt in unterschiedlicher Weise dominieren. Jeder Sicht ordnet Balzert bestimmte Modellierungskonzepte zu, die sich auf „seit längerem bekannte Basiskonzepte zurückführen“ lassen [BALZ96, S.98]. Auch in der Definitionsphase wird nur das „Was“ eines neuen Produktes beschrieben und nicht festgelegt, „Wie“ das Produkt software-technisch realisiert werden soll.
Basiskonzepte sind nach Balzert elementar und originär, d.h. sie lassen sich nicht auf andere Basiskonzepte reduzieren. Außerdem haben sie mindestens eine der Eigenschaften: konzeptionell langlebig, phasenübergreifend verwendbar, in unterschiedlichen Kontexten einsetzbar. Tab. 8 zeigt eine Auflistung der bei Balzert betrachteten Basiskonzepte.

• Data Dictionary • Jackson-Diagramm • ER-Diagramm • Zustandsautomat • Datenflussdiagramm • Funktionsbaum • Pseudocode • Programmablaufplan	• Struktogramm • Warnier-Orr-Diagramm • Entscheidungstabelle • Regeln • Interaktionsdiagramm • Petrinetz • grafische Spezifikation • Kontrollstrukturen

Tab. 8: Basiskonzepte bei Helmut Balzert [BALZ96, S.100]

Analyse-Methoden bestehen schließlich aus verschiedenen Basiskonzepten, die in geeigneter Weise kombiniert sind. Als Stand der Technik gehören für Balzert hierzu Structured Analysis (SA) und Real Time Analysis (RT). Auf dem Weg von der Forschung in die Praxis sieht er Object Oriented Analysis (OOA).
Die Wahl der passenden Basiskonzepte ist für Balzert zum einen abhängig von dem Anwendungsbereich des zu entwickelnden Produktes und zum anderen von der dominanten Komplexitätsart des Produktes (z.B. komplexe Datentypen oder komplexe Algorithmen).
„Die Informationen aus Pflichtenheft, Produktmodell und Benutzungsoberfläche gehen didaktisch-methodisch aufbereitet unter Auswahl geeigneter Beispiele in das Benutzerhandbuch ein.“ [BALZ96, S.110] Insgesamt besteht die fertiggestellte Produkt-Definition dann aus Pflichtenheft, Produktmodell, Konzept der Benutzungsoberfläche und Benutzerhandbuch.

Unternehmensmodellierung betrachtet Balzert in [BALZ98]: "Als besonders wichtig hat sich die Beschreibung und Optimierung der Geschäftsprozesse (Unternehmensprozesse, business processes) erwiesen." [BALZ98, S.718]
Ein existierendes Unternehmen wird beschrieben durch seine Umwelt sowie seine Geschäftsprozesse aus externer und interner Sicht einschließlich benötigter Objekte. „Die Analyse dieser Beschreibungen führt zusammen mit der Unternehmensvision zu Zielen für die Um- oder Neugestaltung der Geschäftsprozesse." [BALZ98, S.722]

Die Neugestaltung des Unternehmens beinhaltet die Analyse, welche Teilprozesse durch die Anwendungssoftware und welche durch Mitarbeiter erledigt werden. Das gleiche wird für die zu verwaltenden Objekte festgestellt. Zur Beschreibung werden drei Objekttypen identifiziert: Interface, Control & Entity Objekte.
"Es ergibt sich als Ergebnis ein Unternehmensmodell, in das Anwendungs-Software als Subsysteme eingebettet sind. Dadurch ergibt sich ein direkter Übergang von dem Unternehmensmodell zu einem oder mehreren OOA-Modellen für die Anwendungs-Software."

Gegenstandsbereich	• **Bereich, aus dem die fachlichen Basisanforderungen kommen,** • **fachliche Basisanforderungen (für Aufwandsschätzung)** • **Vorgaben und Rahmenbedingungen aus der Definitionsphase** • **vagen, verschwommenen, unzusammenhängenden, unvollständigen und widersprüchlichen Anforderungen** • **schriftliche Dokumente** • **vorhandene Softwaresysteme** • **Unternehmensvision** • **Umwelt** • **Geschäftsprozesse**
Methoden	• **Gespräche, Interviews, Befragungen** • **Unternehmensmodellierung (Geschäftsprozesse beschreiben und optimieren)**
dokumentierte Ergebnisse	• **Lastenheft, verbale Beschreibung der fachlichen Basisanforderungen, Aufwandsschätzung** • **vollständige, konsistente, eindeutige und durchführbare Produktanforderungen** • **qualitative und quantitative Eigenschaften des Produkts (Produktdefinition)** • **fachliche lesbare und vergleichbare Anforderungen, die als juristische Vertragsgrundlage dienen können** • **Benutzerhandbuch** • **formales oder semi-formales Produktmodell** • **Funktionen, Daten, Leistungen, Benutzungsschnittstelle, Qualitätsziele** • **Entwicklungs- und Zielumgebung stehen fest** • **Lastenheft, verbale Beschreibung der fachlichen Basisanforderungen, Aufwandsschätzung** • **Unternehmensmodell, in das Anwendungs-Software als Subsysteme eingebettet sind** • **Geschäftsprozesse, Teilprozesse, Objekte**

Tab. 9: Analyse im Vorgehensmodell von Balzert

Zusammenfassung

Das Vorgehensmodell von Balzert will die ingenieurmäßige Konstruktion einer neuen Software unterstützen und versucht dies durch eine möglichst breite und detaillierte Berücksichtigung von älteren und neuen Methoden und Modellen. Analyse soll auf drei

Ebenen durchgeführt werden: Produktebene, Arbeitsplatzebene und Unternehmensebene. Die Identifizierung von Geschäftsprozessen und Anforderungen führt im Ergebnis zu einem Produktmodell.
Balzert stellt viele unterschiedliche softwaretechnische Konzepte nebeneinander (Tab. 8). Sein Vorgehensmodell unterscheidet sich hier insbesondere vom RUP (Kap. 2.2.3), der sich einer objektorientierten Leitlinie verschreibt und dessen Konzepte aufeinander aufbauen. Die bei Balzert nötige Auswahl einiger der nebeneinanderstehenden Konzepte versucht er durch Auswahlkriterien zu erleichtern.

2.2.3 Rational Unified Process (RUP)

Übersicht

Der Rational Unified Process (RUP) vereinigt mehrere Ansätze bzw. Methoden zur Softwareentwicklung zu einem umfangreichen, iterativen Vorgehensmodell. Die Namen seiner drei Entwickler Ivar Jacobson, Grady Booch und James Rumbaugh („die drei Amigos“) verleihen ihm besonderes Gewicht. Das Vorgehensmodell verwendet die Unified Modeling Language (UML). Ihm liegt außerdem ein „reinrassiges objektorientiertes Prozessmodell“ zugrunde [VER00, S.39].
Der RUP kann auch als Framework verwendet werden, das auf spezifische Gegebenheiten eines Unternehmens angepasst werden kann.

Allgemeine Ziele

Ziel des RUP ist es, „hochwertige Software innerhalb eines vorgegebenen Zeitrahmens und Budgets so zu entwickeln, dass die Anforderungen der Endbenutzer erfüllt werden.“ [KRU99, S.xv] Besonderes Augenmerk gilt der Software-Architektur. Der RUP bezeichnet sich auch als architekturzentrierten Prozess. „Der Sinn der Analyse ist es, Anforderungen an das System in eine Form zu übertragen, die dem Software-Designer weiterhilft – das bedeutet: in eine Anzahl von Klassen und Subsystemen.“ [KRU99, S.155]

Einbettung der Analyse

Der RUP definiert verschiedene Workflows (Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration & Change Management, Project Management, Environment), die parallel über vier Phasen ablaufen (Konzeptualisierung, Entwurf, Konstruktion, Übergang). [KRU99, S.23] „Ein Workflow ist eine Abfolge von Aktivitäten, die ein Ergebnis von nachweisbarem Wert erzeugen.“ [KRU99, S.44] Eine Aktivität ist definiert als „eine in sich abgeschlossene Folge von Tätigkeiten, deren Unterbrechung kein sinnvolles Ergebnis liefern würde.“ [VER00, S.78]
Von den insgesamt neun Workflows gibt es drei, in denen Analyse stattfindet:

- Business Modeling (Geschäftsprozessmodellierung)
- Requirements (Anforderungen)
- Analysis & Design

Vorgaben

Der RUP verlangt objektorientierte Analyse-Methoden und die Verwendung von UML. Zur Modellierung der Anforderungen werden Use-Case-Diagramme, Aktivitätsdiagramme, Sequenzdiagramme oder Kollaborationsdiagramme empfohlen.
Eine besondere Rolle spielen Use-Cases: Im RUP sind „die Usecases, die für ein System definiert werden, die Basis für den gesamten Entwicklungsprozess ...“ [KRU99, S.100]

Kruchten empfiehlt die Verwendung von Techniken des Software-Engineerings zur Use-Case-Modellierung. [S.134] Von Vorteil sei auch, wenn das Usecase-Modell leicht in ein Modell des zu erstellenden Systems überführt werden kann.
Geschäftsprozessmodellierung (mit Geschäftsprozess-Use-Cases) ist bei Entwicklung weniger komplexer Software nicht unbedingt notwendig.
Vor Beginn der Analyse-Tätigkeit muss ein Problemfeld abgegrenzt sein. Wie das Problemfeld abgegrenzt wurde, ist in der Literatur aber nicht näher beschrieben.

Ergebnisse

Kruchten definiert eine „Anforderungskategorie“, der er folgende Artefakte zuordnet [KRU99, S.241]:

- Geschäftsprozess-Use-Case-Modell (Geschäftsprozess-Use-Cases, Worker)
- Geschäftsprozessobjektmodell (Entitäten, Organisationseinheiten)
- Ergänzende Geschäftsprozessspezifikation
- Visionsdokument
- Bedürfnisse der Stakeholder
- Use-Case-Modell (Use Cases, Akteure)
- Ergänzende Use-Case-Spezifikation
- Use-Case-Storyboard
- Projektglossar
- Benutzerschnittstellenprototyp

Besonders bei Software mit der später viele Personen arbeiten und über die viele Informationen ausgetauscht werden, soll eine Geschäftsprozessmodellierung vorgenommen werden. Ziel dieser Modellierung ist, „die Struktur und die Dynamik einer Organisation zu verstehen ... sicherzustellen, dass Kunden, Endanwender und Entwickler das gleiche Bild von einer Organisation haben ... [und] Systemanforderungen, die benötigt werden um eine Organisation zu unterstützen, abzuleiten. ... Um diese Ziele zu erreichen, muss ein Modell der Organisation und ihrer Geschäftsprozesse...“ erstellt werden. [KRU99, S.133] Im Einzelnen werden die zu modellierenden Organisationseinheiten grob skizziert und abgegrenzt, die existierenden Geschäftsprozesse (Geschäftsprozess-Use-Cases) und Akteure festgestellt. [KRU99, S.135] Diese Spezifikation wird dann um Workflows, Worker, Entitäten, Verantwortlichkeiten, Aktivitäten und Attribute ergänzt.
Zusätzlich werden außerdem ein ergänzendes Dokument zur Beschreibung von Besonderheiten der Geschäftsprozesse und ein Projektglossar erstellt. Das Projektglossar definiert wichtige gemeinsame Begriffe, die von den Analytikern bei Beschreibung des Systems benutzt werden. [JBR98, S.139]

Weitere Ergebnisse sind eine Übereinkunft mit den Kunden und Anwendern über die Systemleistungen, ein besseres Verständnis der System-Entwickler für die Anforderungen an das System und die Definition der Funktionalität des Systems. Außerdem soll eine Basis zur Planung der technischen Inhalte und zur Abschätzung der Kosten und benötigten Zeit geleistet werden. Schließlich soll die Benutzerschnittstelle für das System beschrieben werden. [KRU99, S.141]
Dazu wird das Problemfeld analysiert, die Bedürfnisse der Stakeholder ermittelt sowie das System definiert und verfeinert. [VER00, S.56/57]. In der Analyse des Problemfeldes nimmt der „Systemanalytiker in enger Zusammenarbeit mit dem Projektmanager die Anforderungen von allen externen Beteiligten des Projektes auf: Stakeholder, Endanwender, Kunden.“

[VER00, S.92] Ergebnis ist ein Visionsdokument, das die Anforderungen der Stakeholder, Kunden und Endanwender enthält. Es ist „eine Art Vertragsbasis“ und richtet sich an das Management von Auftraggeber- und Auftragnehmerseite. Außerdem ist es Grundlage für eine Risikoliste und für die Use-Case-Modellierung. [VER00, S.94]
Messbare Kontrollmöglichkeiten sind im Rational Unified Process als Checkpoints benannt. Die Checkpoints für das Visionsdokument sind die vollständige Klärung und korrekte Wiedergabe des Problems, eine vollständige Liste der Stakeholders („beliebige Person eines Unternehmens ..., die ein berechtigtes Interesse am Ergebnis des Projektes hat.“ [VER00, S.96]), die Einigung aller Akteure über die Systemgrenzen, die Identifizierung und Definition aller Schlüsselfunktionalitäten und die Überprüfung, dass alle Probleme durch die festgelegten Funktionalitäten abgedeckt sind.
Zur Sammlung von Anforderungen werden weiterhin Fragebögen, Workshops und Einzelinterviews durchgeführt. [VER00, S.98/99] „Eine Anforderung wird von einem Auftraggeber oder einem Endbenutzer gestellt, sie beschreibt, wie sich das zu implementierende System verhalten soll“. [VER00, S.85] „Eine Anforderung muss in schriftlicher Form als Diagramm oder Modell vorliegen. Sie muss klar formuliert und anhand von Zahlenmaterial nachprüfbar sein. Ferner muss derjenige, der sie formuliert, auch die notwendige Entscheidungskompetenz besitzen, die Anforderung stellen zu dürfen.“ [VER00, S.86] Die Qualität des zu erstellenden Systems wird in Nicht-Funktionalen Anforderungen festgehalten: Anwenderfreundlichkeit, Wartbarkeit, Performance, Zuverlässigkeit.
Aus dem Visionsdokument wird ein Use-Case-Modell vom Systemanalytiker erzeugt. „Ein Use-Case ist eine Beschreibung einer Menge von Aktionsfolgen, inklusive deren Varianten, die ein System ausführen kann, und die ein erkennbares, nützliches Ergebnis für einen Akteur bringt.“ [VER00, S.111] „Zumeist liegen Anforderungen an ein Software-System in textueller Form vor... Hier besteht die Aufgabe ... aus den Textpassagen alle entscheidenden Use-Cases zu identifizieren.“ [S.112] Ein Akteur ist dabei „... eine außerhalb eines Systems liegende Klasse, die an einer in einem Use-Case beschriebenen Interaktion mit dem System beteiligt ist.“ Das Use-Case-Modell wird in mehreren Iterationen verfeinert.
Die Anforderungen werden schließlich in Spezifikationen übersetzt, die beschreiben, wie das System zu entwickeln ist. Das Use-Case-Modell wird in ein Analysis-Modell verfeinert, aus dem später das Design-Modell entsteht (Abgrenzung siehe Tab. 10). Im Einzelnen werden die Software-Architektur definiert und überarbeitet, Komponenten spezifiziert und Objekte aus den Use-Cases identifiziert, die zu einer Klasse, Paketen oder Subsystemen zusammengefasst werden. [KRU99, S.162]

Analysis Model	**Design Model**
Conceptual model, because it is an abstraction of the system and avoids implementation issues	Physical model, because it is a blueprint of the implementation
Design-generic (applicable to several designs)	Not generic, but specific for an implementation
Three (conceptual) stereotypes on classes: control, entity, boundary	Any number of (physical) stereotypes on classes, depending on implementation language
Less formal	More formal
Less expensive to develop (1:5 ratio to design)	More expensive to develop (5:1 ratio to analysis)
Few layers	Many layers
Dynamic (but not much focus on sequence)	Dynamic (much focus on sequence)
Outlines the design of the system, including its architecture	Manifests the design of the system, including its architecture (one of its views)
Primarily created by "leg work," in workshops and the like	Primarily created by "visual programming" in round-trip engineering environments; the design model is "round-trip engineered" with the implementation model
May not be maintained throughout the complete software life cycle	Should be maintained throughout the complete software life cycle
Defines a structure that is an essential input to shaping the system – including creating the design model	Shapes the system while trying to preserve the structure defined by the analysis model as much as possible

Tab. 10: Kurzer Vergleich zwischen Analysis- und Design-Modell [JBR99, S.219]

Zusammenfassung

Der RUP bietet ein ingenieurmäßiges, objektorientiertes Vorgehensmodell, das ausgehend von Use-Cases die Konstruktion einer Softwarearchitektur unterstützt. Von Analyse-Methoden wird die Identifikation von Use-Cases gefordert, auf deren Basis ein logisches Modell der Software zurückgeliefert wird (Abb. 11).

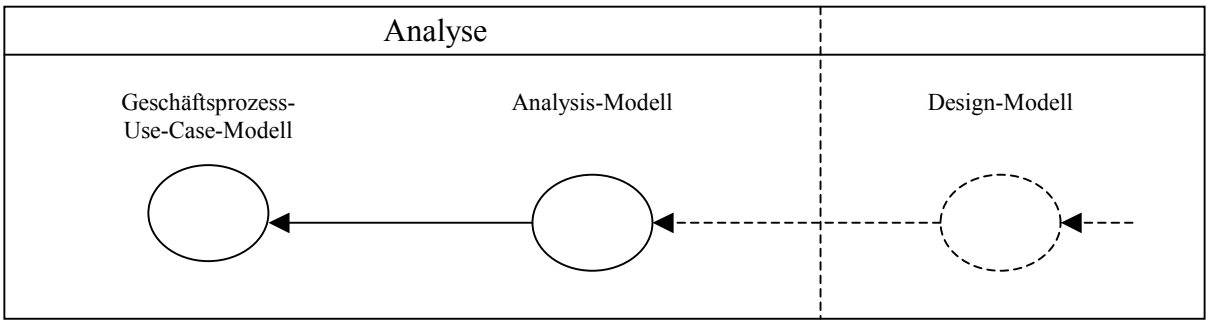


Abb. 11: Folge von Modellen im Rational Unified Process

Das Vorgehensmodell ist mit seinen über vier Phasen parallel ablaufenden Workflows sehr komplex, die Zuordnung der Analyse-Aktivitäten zu den Workflows nicht eindeutig.
Zur Modellierung soll insbesondere UML herangezogen werden.

Gegenstandsbereich	Problemfeld, Anforderungen, Beteiligte (Stakeholder, Kunden, Endanwender), bestehende Dokumente, Geschäftsprozess-Usecases, Usecases
Methoden	**Fragebögen, Interviews, Workshops, Modellierung mit UML (Usecase-, Aktivitäts-, Sequenz-, Kollaborationsdiagramme)**

dokumentierte Ergebnisse	• **Visionsdokument**
	  o **Vertragsbasis**
	  o **Funktionale Anforderungen**
	  o **Nicht-Funktionale Anforderungen**
	• **verfeinerte Use-Cases (Modelle und Storyboard)**
	• **Projektglossar**
	• **Problembeschreibung**
	• **Vollständige Liste der Stakeholders**
	• **Identifikation + Definition aller Schlüsselfunktionen**
	• **Benutzerschnittstellenprototyp**
	• **Software-Architektur**
	• **Analyse-Modell, das eine Abstraktion des Systems ist**
	  o **Klassen**
	  o **Subsysteme**

„mentale“ Ergebnisse	• **Einigung aller Akteure über Systemgrenzen** • **Problemklärung**

Tab. 11: Analyse im Rational Unified Process

2.2.4 Implementation Management Guide (IMG)

Übersicht

Die deutsche SAP-AG ist mit ihren Standardsoftwareprodukten R2 und R3 internationaler Marktführer. Ihre Produkte werden zum Teil synonym zum Begriff Standardsoftware gebraucht. Als Vorreiter auf dem ERP-Gebiet strahlen die SAP-Vorgehensmodelle auf den gesamten Bereich der Standardsoftwareeinführung aus.
Das neueste Vorgehensmodell zur Einführung von SAP R3 ist AcceleratedSAP (ASAP). Dieses basiert auf dem älteren Vorgehensmodell IMG (Implementation Management Guide), das wiederum dem IMW (Implementation Management Ware) zu Einführung von R2 ähnlich ist. Für umfangreiche Projekte wird weiterhin das IMG empfohlen [HRS99, S.9], das deswegen auch an dieser Stelle betrachtet wird.

Allgemeine Ziele

Das IMG-Vorgehensmodell unterstützt die geschäftsprozessorientierte Einführung von SAP R3. „Es unterscheidet sich von anderen Vorgehensmodellen zur Informationssystem-entwicklung darin, dass eine ‚fertige', lediglich an die Bedürfnisse des Unternehmens anzupassende Lösung, eingeführt werden soll. Der Aufwand für die Systemkonzeption, Systembeschreibung und Programmierung wird durch den Einsatz der Standardsoftware erheblich reduziert bzw. ist zum Teil überhaupt nicht notwendig.“ [MEI95, S.488]

Einbettung der Analyse

Das Prozessmodell von SAP für die Einführung der Standardsoftware R3 besteht auf oberster Ebene aus vier aufeinanderfolgenden Phasen (siehe z.B. [BAR96, S.50]):

1. Organisation und Konzeption
2. Detaillierung und Realisierung
3. Produktionsvorbereitung
4. Produktivbetrieb

[MEI95] sieht den SAP-Einführungsprozess in einen größeren Rahmen eingeordnet, den er „Implementation life cycle“ nennt. Dieser besteht aus Projektvorlauf, Einführung und Wartung.
Analyse und Planung fallen je nach Projekt in den Projektvorlauf oder in den Anfang der Einführung (Organisation und Konzeption). „Somit hat die Anforderungsanalyse eine Zwitterstellung, die Projektvorlauf und SAP-Einführung miteinander verbindet.“ [MEI95, S.490] (Im neuen Vorgehensmodell ASAP ist die Projektvorbereitung als erste Phase der Einführung inbegriffen.) Der Projektvorlauf besteht insgesamt aus Problemdefinition, Anforderungsanalyse und Projektvorbereitung.

Vorgaben

„Inhalt und Aufgabe der Problemdefinition ist oftmals eine BPR-Analyse oder die Entwicklung einer zukünftigen IT-Strategie des Unternehmens“ [MEI95, S.490] Das R3-Referenzmodell kann dabei eine „nützliche Informationsquelle zur Auffindung von ‚best business practices’ sein, die mit dem R/3-System abbildbar sind und aus langjähriger Softwareentwicklung der SAP zusammen mit ihren Kunden resultieren“. Auch in der Geschäftsprozessanalyse und Anforderungsanalyse ist das R3-Referenzmodell von „großer Bedeutung“, da es eine „... Wissensbasis über die komplette SAP-Funktionalität und die integrierten Geschäftsprozesse des R/3-Systems ...„ darstellt und auf seiner Grundlage „... kostengünstig aussagefähige Ergebnisse für die Entscheidungsfindung über eine Einführung ... erarbeitbar sind.“ [MEI95, S.490f] Das R3-Referenzmodell basiert auf sechs verschiedenen Sichten: Prozesssicht, Datensicht, Funktionssicht, Organisationssicht, Kommunikationssicht, Informationsflusssicht.

Ergebnisse

Das Ergebnis der Problemdefinition ist die Entscheidung für Individual- oder Standardsoftware. Spätestens in der ersten Phase (Organisation und Konzeption) wird die Entscheidung für R3 gefällt.
Ziel der Anforderungsanalyse ist es herauszufinden, ob das Unternehmen (Unternehmensstruktur, Geschäftsprozesse und Unternehmensziele) mit dem Einsatz von R3 adäquat unterstützt werden kann. Dabei werden Fragen gestellt wie [MEI95, S.490]:

- Werden alle wichtigen Funktionen abgedeckt?
- Können die organisatorischen Strukturen abgebildet werden?
- Werden komplette Geschäftsprozesse unterstützt?
- Welche Änderungen in der Organisation sind zu erwarten?
- Welche Kosten und Risiken sind zu erwarten?

Die Anforderungsanalyse wird in drei Schritten durchgeführt:

1. Analyse der Ausgangssituation
2. Geschäftsprozessanalyse
3. Realisierungsplanung

In der Analyse der Ausgangsituation werden die strategischen Anforderungen (Planungsstrategien, Controlling-Strategien etc.), die generellen Rahmenbedingungen (Unternehmensstruktur, Geschäftsfelder, Kern-Prozesse etc.), die Projektziele (strategische Verbesserungspotentiale) und der Handlungsbedarf definiert.
In der Geschäftsprozessanalyse werden zusammen mit den Fachabteilungen die Kern-Prozesse (auch Szenarien) herausgearbeitet und dabei alle organisatorischen und technischen Verbesserungspotentiale dokumentiert. „Die funktionalen Details der einzelnen Prozesse sind in der Regel in dieser Phase der Anforderungsanalyse nicht flächendeckend zu untersuchen." [MEI95, S.492] Kritische Prozesse sollten aber als erste R3-Prototypen abgebildet werden. Die Ergebnisse der Analysearbeit sind insgesamt:

- Prozess-/ Funktionsabdeckungsgrad von R3
- Verbesserungspotentiale
- Optimierungspotentiale der Prozesse, inklusive der erforderlichen / empfohlenen organisatorischen Änderungen
- Anzahl der Schnittstellen

In der Realisierungsplanung wird auf Basis der Analyseergebnisse die Realisierung geplant. Dazu gehören Aufwandsschätzung der Einführung, Empfehlung für eine Einführungsstrategie, Projektorganisation und Projektplan.
In der Projektvorbereitung wird der Projektauftrag formuliert und verabschiedet. Er umfasst „alle für die Einführung relevanten Vorgaben wie Projektziele, Projektstruktur, Verantwortlichkeiten und Budget." [MEI95, S.494]

Wurde die Anforderungsanalyse nicht bereits im Projektvorlauf durchgeführt, steht sie als erste Aktivität am Anfang der Phase Organisation und Konzeption. Ihre Ergebnisse werden dann detailliert und ein Soll-Konzept erarbeitet. Bei der Erarbeitung des Soll-Konzeptes werden alle funktionalen Details flächendeckend untersucht. [MEI95, S.492] Es sollte sichergestellt werden, „dass bei der R/3-Einführung die Geschäftsprozesse im Vordergrund stehen und nicht funktionale Abteilungssichten dominieren..." [MEI95, S.494]
Die ausgewählten Prozesse werden auf Vollständigkeit überprüft. Dann wird untersucht, ob die Detailfunktionalität von R3 für die Abbildung der Prozesse ausreicht bzw. zuviel ist. Außerdem wird die Zuordnung der Informationsobjekte und Organisationseinheiten zu den Geschäftsprozessen überprüft.
Abschließend werden die Veränderungen zum Ist-Zustand der Anwender-Organisation (Abwicklung und organisatorische Zuständigkeiten) festgehalten, die einerseits die Schwachstellen dokumentieren und andererseits Grundlage für die Entscheidungsfindung des Managements sind. Fehler in dieser Projektphase lassen sich später nur mit überproportionalem Aufwand korrigieren: „Daher muss bei der Erstellung des Soll-Konzeptes besonders sorgfältig vorgegangen werden." [BEI95, S.495]

Vorgaben	Ausgangssituation, Geschäftsprozesse, R3-Referenzmodell
Methoden	**Anforderungsanalyse, Geschäftsprozessanalyse, BPR-Analyse**
dokumentierte Ergebnisse	• **IT-Strategie** • **strategische Anforderungen** • **generelle Rahmenbedingungen (Unternehmensstruktur, Geschäftsfelder, Kern-Prozesse)** • **Prozess-/ Funktionsabdeckungsgrad von R3** • **Verbesserungspotentiale** • **Optimierungspotentiale der Prozesse, inklusive der erforderlichen bzw. empfohlenen organisatorischen Änderungen** • **Anzahl der Schnittstellen** • **Projektziele** • **Handlungsbedarf** • **Dokumentation der organisatorischen und technischen Verbesserungspotentiale** • **flächendeckende Dokumentation aller funktionalen Details** • **Veränderungen zum Ist-Zustand der Organisation**
„mentale“ Ergebnisse	• **Entscheidung für Individual- oder Standardsoftware** • **Adäquatheit der R3-Unterstützung für Unternehmen**

Tab. 12: Analyse im SAP-IMG-Vorgehensmodell

Zusammenfassung

Das IMG-Vorgehensmodell unterstützt die Einführung der Standardsoftware R3 und konzentriert sich dabei auf die Geschäftsprozesssicht. Analyse-Methoden sollen Anforderungen und Geschäftsprozesse identifizieren und insbesondere organisatorische und technische Verbesserungspotentiale sowie den Abdeckungsgrad der Software aufzeigen.
Das IMG-Modell betont wie das IBSIS-Modell den radikalen Ansatz des BPR, um „verkrustete Strukturen“ aufzubrechen.

2.2.5 Integrierte betriebswirtschaftliche Standardinformationssysteme (IBSIS)

Übersicht

Barbitsch bietet in seinem Buch „Einführung integrierter Standardsoftware“ ein umfangreiches Phasenmodell zur Einführung von Standardsoftware (bei ihm IBSIS = Integrierte betriebswirtschaftliche Standardinformationssysteme) an (Abb. 12), das von konkreten Produkten abstrahiert.

Allgemeines Ziel

Zentrales Anliegen einer Einführung ist für Barbitsch *Business Process Redesign (*BPR), für das die einzuführende Standardsoftware der Wegbereiter sein soll. Sie ermögliche erst, die Organisation weitgehend unabhängig von den „Restriktionen des Ist-Zustandes“ [BAR96, S.45] neu zu gestalten. Es entstehe ein Reorganisationszwang, weil von dem der Standardsoftware zu Grunde liegenden betriebswirtschaftlichen Modell und der in der Software abgebildeten Organisation nicht abgewichen werden kann. [BAR96, S.46]

Abb. 12: Phasenmodell Standardsoftwareeinführung [BAR96, S.97]

Einbettung der Analyse

Das IBSIS-Vorgehensmodell besteht aus sieben Phasen, die durch sogenannte Gateways voneinander getrennt sind. Die Gateways sind „eine Art von übergeordneten Kontrollpunkten, die vor Eintritt in die nächste Phase passiert werden müssen." [BAR96, S.96] Barbitsch geht aber auch davon aus, dass kein rein sequentieller Ablauf sondern Iterationen stattfinden. [S.95] Es wird in Gesamtprojektphasen und Teilprojektphasen unterschieden. Die Entscheidung, ob die Anforderungen an ein Gateway des Gesamtprojekts erfüllt sind „liegt grundsätzlich beim Topmanagement." [S.99] Die Gateways der Teilprojekte werden von Teilprojektleitern überwacht. Das Durchschreiten eines Gateways des Gesamtprojekts wird überall im Unternehmen verkündet, „am besten im Rahmen einer groß angelegten Informationsveranstaltung". [S.99] Das Vorgehensmodell ist als Gatewayplan aufgebaut. Die Phasen des Modells werden darin vor allem durch ihre Ergebnisse und Kontrollpunkte beschrieben.
Analyse-Aktivitäten finden in den beiden Phasen *Vorbereitung* und *Analyse und Grobdesign ohne IBSIS* statt.
Nach Barbitsch können Analyse und Design nicht „als klar getrennte aufeinanderfolgende Schritte gesehen werden. Es handelt sich hier vielmehr um einen iterativen Prozess." [BAR96, S.168]

Vorgaben

Vor Projektbeginn (Gateway Projektauftrag [BAR96, S.108]) müssen Vision, Strategien und Projektziele feststehen, die Entscheidung für Standardsoftware festliegen (noch kein bestimmtes Produkt) und der Gesamtprojektleiter sowie ein BPR-Sponsor festgelegt sein. Der BPR-Sponsor ist ein Mitglied des Topmanagements, das durch seine Macht den Innovationsprozess aktiv und intensiv fördert und die geforderten Änderungen zusammen mit dem übrigen Topmanagement vorlebt. [BAR96, S.114]. In der strategischen Informations-

systemplanung [BAR96, S.113] muss die Informationssystemlandschaft des Unternehmens für die nächsten Jahre geklärt worden sein. Dazu gehört:

- Standortbestimmung
  - Wie ist die Organisation strukturiert?
  - Welche Probleme herrschen im Informationsverarbeitungs-Bereich vor?
  - Wie ist der Entwicklungsstand der Organisation der Informationsverarbeitung?
  - Welche strategische Bedeutung hat die Informationsverarbeitung für die Organisation?
- Analyse der Bedingungslage
  - externe Bedingungen (rechtliche Bestimmungen, Entwicklungen im Bereich der Informationstechnik, usw.)
  - interne Bedingungen (Daten- und Applikationsstruktur, Ressourcen, Organisation der Informationsverarbeitung etc.)
- Bestimmung der strategischen Richtung für die Informationsverarbeitung
  - Vision, aus der Strategien abgeleitet werden
- Entwicklung von Informationssystem-Strategien
  - Daten-, Applikations- und Kommunikationsstruktur (Informationssystem-Architektur, Reihenfolgeplanung für Informationssystem-Entwicklung
  - Ressourcen für die Informationsverarbeitung (Mitarbeiter, IT, Budget etc.)
  - Organisation und Führung der Informationsverarbeitung

„Die Entscheidung, ein bestehendes Informationssystem abzulösen und durch ein IBSIS zu ersetzen, ist somit Ergebnis der strategischen Informationssystemplanung.“ [BAR96, S.114]
Zum Gateway Projektauftrag gehört auch die Erstellung von „... technologischen Basisszenarien, auf die bei der endgültigen Entscheidung zurückgegriffen werden kann.“ [S.114] In den Basisszenarien wurde vor Projektbeginn „... der Einsatz unterschiedlicher IBSIS-Alternativen bereits bewertet und dargestellt.“ [S.141] Die Auswahl eines bestimmten IBSIS geschieht erst nach der Analyse in der dritten Phase.

Ergebnisse

In der *Vorbereitungsphase* wird der organisatorische Rahmen und die Grundkonzeption für das Projekt festgelegt (z.B. Abgrenzung des Untersuchungsfeldes und Beschaffung von Ressourcen). Desweiteren dient diese Phase auch als Lernphase für BPR, indem in einem unkritischen Bereich ein Pilotprojekt durchgeführt wird.
Dann werden Teilprojekte ins Leben gerufen und für jedes Teilprojekt Projektorganisation, Projektcontrolling, Untersuchungsumfang sowie Ziele festgelegt. Außerdem muss ein Business Process Redesign-Klima geschaffen worden sein (Gateway Allgemeine Vorbereitung, [S.122]). Insbesondere wurden die Untersuchungsmethoden festgelegt, der Untersuchungsbereich bestimmt, die Geschäftsprozesse identifiziert, ausgewählt und in Teilprozesse zerlegt und Formalziele verfeinert und dokumentiert.
Die meisten Untersuchungsmethoden kann „... jeder Mitarbeiter individuell und unabhängig von den anderen Projektmitgliedern einsetzen ...“ „Daneben müssen aber im Projekt Instrumente eingesetzt werden, die von mehreren Personen auf die gleiche Art und Weise zu verwenden sind und deren Ergebnisse zu einem einheitlichen Ganzen zusammengefügt werden müssen.“ [S.130] Barbitsch nennt hier insbesondere die Kunden/ Lieferantenbeziehungsanalyse und die Prozessanalyse. Das Ziel der Kunden/ Lieferantenbeziehungsanalyse ist die kontinuierliche Aufdeckung von Reorganisationspotentialen und die

Beeinflussung der Unternehmenskultur in Richtung mehr Änderungsbereitschaft („keine Leistung ohne Kunden“). Die Prozessanalysemethode sollen für folgende Aufgaben geeignet sein [S.131]:

- Dokumentation der Analyseergebnisse der Geschäftsprozesse,
- Entwickeln eines Sollmodells der Geschäftsprozesse und
- Abgleich des Sollmodells der Geschäftsprozesse mit dem Modell des IBSIS
- wiederverwendbare Modellierung für leichte Veränderbarkeit
- Abbildung von Funktionen, Organisationseinheiten, Daten, Zeit und Kosten

Die Methoden sollten außerdem leicht beherrschbar sein (Schulungszeit von ein bis zwei Stunden). [S.132] Existieren im Unternehmen bereits Modelle sowie Methoden und Werkzeuge zur Analyse und zum Beschreiben von Prozessen, sollte auf diese zurückgegriffen werden. Barbitsch empfiehlt außerdem die erweiterten ereignisgesteuerten Prozessketten (eEPK) zur Modellierung: „Die Prozesskette enthält die wesentlichen Beschreibungselemente eines Prozesses, nämlich Ereignis, Funktion, Organisationseinheit und Daten.“ [BAR96, S.132]
Aus den durch BPR betroffenen Unternehmensbereichen werden die neuzugestaltenden Geschäftsprozesse ausgewählt. Bei der unternehmensweiten Einführung eines IBSIS „... ist nicht zu erwarten, dass alle Prozesse und Funktionen, die durch das IBSIS unterstützt werden, auch zu reorganisieren sind.“ [S.102] Es kann aber sein, dass manche „... Prozesse BPR unterzogen werden, obwohl sie vom IBSIS gar nicht berührt werden.“ [BAR96, S.103]
Zur Bestimmung der relevanten Unternehmensbereiche kann Porters Wertkette helfen [POR85, S.38ff]. Auch die vor Projektbeginn erstellten Basisszenarien (s.o.) helfen hierbei. Die Abgrenzung des Untersuchungsbereichs klärt die Frage, welche Organisationseinheiten und Funktionen wie stark durch BPR betroffen sind. „Die Definition des Untersuchungsbereiches ist eine Voraussetzung, um das Projekt gegenüber den Beteiligten transparent zu machen und damit die Basis für eine offene und ehrliche Kommunikation zu legen.“ [S.141]
Zur Identifikation von Prozessen liefert Barbitsch folgende Definition:

> Ein Prozess bezeichnet eine strukturierte Abfolge von Aktivitäten, die für einen internen oder externen Kunden aufgrund von festgelegten Eingangsgrößen ein bestimmtes Ergebnis von Wert erzeugt. Er ist ein Rahmen mit einem Anfang und einem Ende, der Aktivitäten zeitlich und örtlich ordnet. Ein Prozess kann einerseits selbst und andererseits über sein Ergebnis gemessen und beurteilt werden. [BAR96, S.23]

„Das wesentliche Potential des BPR liegt in der funktionsübergreifenden Betrachtung der Aktivitäten.“ [BAR96, S.141] Deswegen sollten keine Aktivitäten aus einem Prozess gelöst und einzeln optimiert werden. BPR sollte sich auf die Kernprozesse konzentrieren, die gefunden werden können, in dem man folgende Frage beantwortet: „Welche Produkte (Leistungen) erwarten die externen Kunden, und welche Funktionen müssen wie durchlaufen werden, um diese Leistungen zu erstellen?“ [S.142]
Die Prozesse werden dann in überschaubare Teilprozesse zerlegt [S.143] „Die Frage nach der Zerlegungstiefe kann nicht allgemeingültig beantwortet werden ... Der Detaillierungsgrad wird durch das weiche Kriterium der Beherrschbarkeit bestimmt. Ein Prozess ist dann tief genug zerlegt, wenn der Ablauf vom Teilprojektteam verstanden werden kann und die Schwachstellen, die deutliches Verbesserungspotential beinhalten, erkannt sind. Er ist zu weit

verfeinert, wenn aus dem hohen Detaillierungsgrad keine nennenswerten Verbesserungsmöglichkeiten mehr resultieren.“ [S.143]
Auf Basis der Teilprozesse werden Teilprojekte definiert, deren Aufgaben in den nachfolgenden Phasen „... von der Analyse der Teilprozesse, dem Entwickeln von Sollprozessen bis zum Abgleich der Sollprozesse mit dem IBSIS reichen.“ [S.145]
Dann werden „die Ziele in bezug auf Inhalt, Maßstab, Ausmaß und zeitlichen Bezug der Zielerreichung näher ... “ bestimmt. [S.146] Grundlagen dafür sind die vorgegebenen Projektziele, der Gatewayplan, der Untersuchungsbereich, die ausgewählten Teilprojekte und die Basisszenarien. Die Formalziele bestehen u.a. aus Leistungszielen, Terminzielen, Kostenzielen, Nutzungszielen (Akzeptanz, Benutzbarkeit, Produktivität, Sicherheit, Verfügbarkeit, Wirksamkeit, Wirtschaftlichkeit des IBSIS) und Wartungszielen (Releasefähigkeit etc.). [S.147]
Die Vorbereitungsphase eines Teilprojekts ist abgeschlossen, wenn das Team für das Teilprojekt bestimmt wurde und seine Aufgaben kennt, die Teilprozessziele und –strategien festgelegt wurden und der Untersuchungsbereich für den Teilprozess dokumentiert wurde. [S.153] Die Vorbereitungsphase für das Gesamtprojekt ist abgeschlossen, wenn die Vorbereitungsphasen aller Teilprojekte beendet sind, alle Pilotprojekte abgeschlossen sind und die Erfahrungen in das Projektkonzept eingearbeitet sind. [S.164]

Die zweite Phase nennt sich *Analyse und Grobdesign ohne IBSIS*. Dabei müssen die Projektmitarbeiter die IST-Prozesse der Anwender-Organisation analysieren. „Sie müssen soviel Information zusammentragen, damit sie ein Modell der Realität erstellen können. Anhand des Modells ist dann zu beurteilen, ob die gesetzten Prozessziele erreicht werden können. Ziel ist es, ´das komplexe, dynamische und intransparente Gebilde` Geschäftsprozess zu verstehen. Es dürfen daher nicht nur Daten, Funktionen und Organisationseinheiten beschrieben werden, sondern es sind auch die Regeln und Prinzipien zu analysieren, die zu der IST-Situation geführt haben. Das erarbeitete Strukturwissen soll die Projektbeteiligten in die Lage versetzen, Schwachstellen und Verbesserungspotentiale zu erkennen.“ [S.167] Das Ergebnis der Analyse entsteht vor allem in den Köpfen der Analytiker: Es „... entstehen in den Köpfen der Projektbeteiligten kognitive Karten des IST-Zustands des Unternehmens.“ Die kognitiven Karten oder Schemata „dienen als Bezugsrahmen für alle weiteren Handlungen.“ [BAR96, S.167]
Die Ergebnisse der Analyse sind: Prozessablauf, Ergebnisse und Kunden, Regeln und Prinzipien, Kenngrößen, Stärken und Schwächen.
„In der Phase Analyse und Grobdesign ohne IBSIS geht es nicht nur um das Neugestalten von Prozessen Durch begleitende Untersuchungen, wie zum Beispiel der Kunden-/ Lieferantenbeziehungsanalyse, werden auch andere Unternehmensbereiche auf Verbesserungspotentiale hin untersucht.“ [S.168]
Analysiert werden soll auch das „Verhalten des Sponsors und der Mitglieder des Steuerkreises“. Dieses soll, wenn es nicht der Vorbildfunktion gerecht wird, gegebenenfalls korrigiert werden.

Im zweiten Teil der Phase (*Grobdesign ohne IBSIS*) werden die neuen Prozesse entworfen. Dazu werden Prozessprototypen erstellt, die völlig unabhängig von der noch auszuwählenden Standardsoftware sind. „Dazu entwerfen die Mitarbeiter des Teilprojektteams nach dem Business Process Redesign einen SOLL-Prozess weitgehend losgelöst vom IST-Zustand.“ Die in der Analyse erhobenen „... Problembereiche der IST-Prozesse ...“ sind aber Quelle für Verbesserungsmöglichkeiten. „Durch dieses verzahnte Vorgehen ... soll schließlich die Effizienz und Effektivität eines Geschäftsprozesses gesteigert werden“. [S.171] Außerdem haben die Beteiligten den in der Analyse erarbeiteten Ist-Zustand des Unternehmens im

Hinterkopf und laufen nicht Gefahr neue Prozesse zu entwerfen, die mit dem Unternehmen gar nicht machbar wären. Die erstellten Soll-Prozesse werden zur Überprüfung in einem *Review* mit den Ist-Prozessen verglichen. Die Erkenntnisse aus dem Review werden dann wieder in das Soll-Modell eingearbeitet. [S.186]
Das erstellte Soll-Modell ist die Basis für die Auswahl eines konkreten IBSIS. Die Eigenschaften dieses IBSIS bzw. dessen Referenzmodell führen dann wieder zu Veränderungen des Soll-Modells.

Zusammenfassung

Das Vorgehensmodell von Barbitsch unterstützt die Einführung von Standardsoftware unabhängig von einem konkreten Produkt und konzentriert sich hierbei insbesondere auf die radikale Neukonstruktion von Geschäftsprozessen. Analyse soll ein gemeinsames Verständnis der Thematik und der Entwicklung einer gemeinsamen Sprache dienen. Weiterhin dient sie der Überprüfung der Vollständigkeit und Machbarkeit des Soll-Modells sowie der Erklärung und Bewertung von Veränderungsmaßnahmen. Der spätere Entwurf des Soll-Modells soll aber möglichst unabhängig vom Ist-Modell auf der „grünen Wiese“ erfolgen.
Barbitsch vermeidet die Verwendung der Begriffe Anforderung und Anforderungsanalyse.

Vorgaben	Anwender-Organisation, Untersuchungsfeld, Geschäftsprozesse, Basisszenarien
Methoden	**BPR, Kunden / Lieferantenbeziehungsanalyse, Prozessanalyse**

dokumentierte Ergebnisse	• **organisatorischer Rahmen und Grundkonzeption für Projekt festgelegt**
	• **Abgrenzung des Untersuchungsfeldes**
	• **Untersuchungsmethoden festgelegt**
	• **Geschäftsprozesse identifiziert, ausgewählt, dokumentiert und in Teilprozesse zerlegt**
	• **Prozessmodell**
	  o **Funktionen, Organisationseinheiten, Daten, Zeit, Kosten**
	  o **Regeln und Prinzipien**
	• **Formalziele (Leistungsziele, Terminziele, Kosten, Nutzungsziele)**
	• **Schwachstellen und Verbesserungspotentiale**

„mentale“ Ergebnisse	• **gemeinsames Verständnis** • **gemeinsame Sprache** • **Kognitive Karten**

Tab. 13: Analyse in der IBSIS-Einführung

2.3 Vorgehensmodelle im Spannungsfeld

Dieses Unterkapitel stellt das Spannungsfeld zwischen Standardsoftwareeinführungsprozessen und Softwareneuentwicklungsprozessen in den Vordergrund. Zuerst wird betrachtet, wie sich die Vorgehensmodelle selbst in das Spannungsfeld einordnen. Dann wird gezeigt, dass die Vorgehensmodelle auf der einen Seite des Spannungsfeldes eher die Konstruktion und Veränderung eines Software-Produktes verfolgen und auf der anderen Seite eher die Optimierung der Anwender-Organisation. Dies soll herausgestellt werden, in dem die Vorgehensmodelle unter diesen beiden Tendenzen (Softwarekonstruktion und Reorganisation), in jeweils einem Unterkapitel für die beiden Seiten des Spannungsfeldes beleuchtet werden. Anschließend werden die Tendenzen zusammen in den Blick genommen und die Unterschiede auf beiden Seiten des Spannungsfeldes im Zusammenhang aufgezeigt.

Die Vorgehensmodelle geben explizit an, welche Softwareprozesse sie unterstützen wollen (Abb. 13). Auf Seite der Softwareneuentwicklung stehen das Modell von Helmut Balzert und der Rational Unified Process: „... hochwertige Software ... so zu entwickeln, dass die Anforderungen der Endbenutzer erfüllt werden.“ [KRU99, S.xv]
Das IBSIS-Vorgehensmodell von Barbitsch und das IMG-Vorgehensmodell von SAP ordnen sich Softwareprozessen der Standardsoftwareeinführung zu: „Entstanden ist ein Rahmenkonzept für die Abwicklung eines Projektes zur Einführung eines IBSIS ...“ [BAR96, S.95].
STEPS dagegen will beide Softwareprozesse abdecken. Eine etwas stärkere Betonung liegt auf der Seite der Softwareentwicklung: STEPS will „... Software-Entwicklung als integrativen Teil einer übergreifenden Organisationsentwicklung ...“ betrachten. Andererseits wird auch Standardsoftware in den Blick genommen: „Gegenstand ist heute meist die Software-Unterstützung ... auf der Grundlage von angepasster Standardsoftware.“ [FKRW97, S.13]. In Abb. 13 ist STEPS deswegen etwas in Richtung Softwareneuentwicklung verrückt dargestellt.

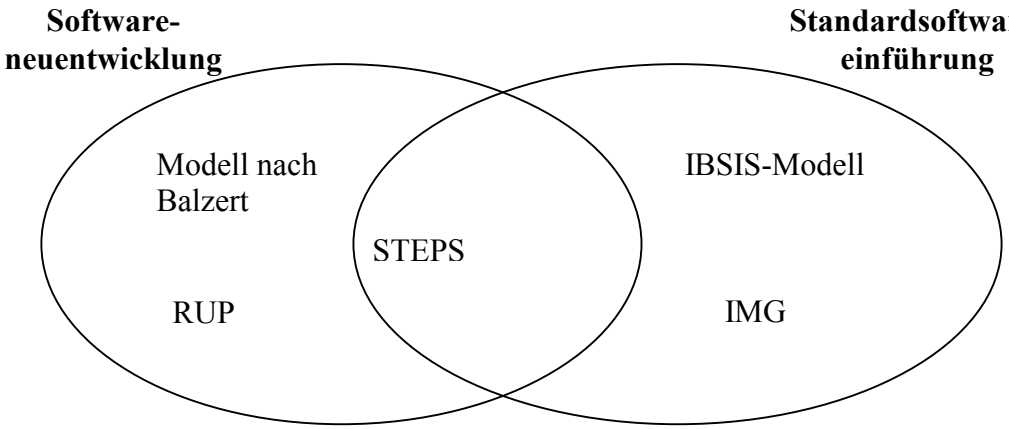


Abb. 13: Selbsteinordnung der Vorgehensmodelle

Die beiden in den folgenden Unterkapiteln festgestellten Tendenzen Softwarekonstruktion und Reorganisation dienen in Kapitel 4 zur Klassifizierung von Analyse-Methoden. Diese Anforderungen an Analyse-Methoden wurden herausgehoben, weil sie das betrachtete Spannungsfeld am stärksten polarisieren. Andere Anforderungen sind auf beiden Seiten des Spannungsfeldes zu finden oder zu abstrakt formuliert, um klare Unterschiede zu anderen Anforderungen aufzeigen zu können. Dies liegt daran, dass die Vorgehensmodelle generell eher konkrete Methoden, Modelle usw. nennen (BPR, Kunden/Lieferantenbeziehungs-analyse, Prozessanalyse, Interviews, Fragebögen, UML), anstatt anzugeben, welche Eigen-schaften von einer Methode etc. erwartet werden (s.a. Kap. 5.2). Da die Analyse-Methoden in diesem Kapitel als Blackbox betrachtet werden, besteht nicht die Möglichkeit, in die Selbstbeschreibung einer Methode zu schauen.
Die festgestellten Tendenzen der Vorgehensmodellen sind gleichzeitig die zentralen Merkmale der Softwareprozesse. Die Zuordnung geschieht über die Selbsteinordnung der Vorgehensmodelle. Für Standardsoftwareeinführungsprozesse ist also eher Reorganisation und für Softwareneuentwicklungsprozesse eher Softwarekonstruktion wichtig.

2.3.1 Softwarekonstruktion

Die verschiedenen Vorgehensmodelle stellen die Konstruktion und Veränderung von Software unterschiedlich stark in den Mittelpunkt. Dabei lässt sich erkennen, dass Softwareprozesse der Neuentwicklung Softwarekonstruktion in den Mittelpunkt stellen, während Standardsoftwareeinführungsprozesse sich damit nur am Rande beschäftigen.

Der RUP bezeichnet sich selbst als „architekturzentrierten Prozess“ [KRU98, S.79ff]. Die Analyse bereitet die Konstruktion einer Software-Architektur vor, in dem ein Analyse-Modell erstellt wird, das eine Abstraktion des zukünftigen Systems ist: „Der Sinn der Analyse ist es, Anforderungen an das System in eine Form zu übertragen, die dem Software-Designer weiterhilft – das bedeutet: in eine Anzahl von Klassen und Subsystemen.“ [KRU99, S.155] Die geforderte Modellierung von Geschäftsprozess-Usecases betrifft weniger Unternehmensabläufe als Schnittstellen zum Softwaresystem. Usecases sind „... die Basis für den gesamten Entwicklungsprozess ...“ [KRU99, S.100].

Das Modell von Helmut Balzert konzentriert sich auf „... das zu entwickelnde Software-Produkt ...“ und seine Basisanforderungen, die eine „... bewusste Konzentration auf die fundamentalen Eigenschaften des Produktes und ihre Beschreibung auf einem hinreichenden Abstraktionsniveau ...„ darstellen [BALZ96, S.57]. Zum Pflichtenheft gehört ein „...

semiformales bzw. formales Produktmodell ...“ [BALZ96, S.112] Auch ein Prototyp ist Teil der Analyse-Ergebnisse. Im zweiten Teil seines Lehrbuches betrachtet Helmut Balzert Unternehmensmodellierung im Zusammenhang mit Software-Entwicklung: [BALZ98, S.717] Hier erweitert Helmut Balzert die reine Produktmodellsicht auf ein Unternehmensmodell: „Es ergibt sich als Ergebnis ein Unternehmensmodell, in das Anwendungs-Software als Subsysteme eingebettet sind.“ [BALZ98, S.722] Helmut Balzert präsentiert Unternehmensmodellierung aber weitgehend getrennt von und wenig integriert mit seinem Vorgehensmodell. Andererseits gibt es Anhaltspunkte, dass mit Geschäftsprozessen eigentlich Anwendungsfälle gemeint sind (s.a. Kap. 3.2.1), die sich nicht auf akteursübergreifende Unternehmensabläufe, sondern auf die Schnittstellen zur Software konzentrieren: Zum einen spricht Balzert von der Identifikation der Objekttypen Interface, Control und Entity und sagt, da „... ein großer Teil des Unternehmensmodells durch Anwendungssoftware realisiert werden muss, müssen die Konzepte und Notationen kompatibel zu Konzepten und Notationen der Software-Entwicklung sein." [BALZ98]

STEPS, das ja sowohl Softwareneuentwicklung als auch Standardsoftwareeinführung unterstützen will, sieht die Konstruktion einer Softwarearchitektur als technisches Kernanliegen: „Technisches Kernanliegen der evolutionären Systementwicklung ist eine änderbare Softwarearchitektur, die die anwendungsfachlichen Gegebenheiten direkt widerspiegelt. Dies ist mit einer modularen, besser noch mit einer objektorientierten Konstruktionstechnik leistbar...“. [FKRW97, S.15] Wie die von STEPS geforderte Aufgabenorientierte Anforderungsermittlung die Konstruktion einer Software-Architektur unterstützen soll, wird nicht näher beschrieben.

Das IMG-Modell „... unterscheidet sich von anderen Vorgehensmodellen zur Informationssystementwicklung darin, dass eine ‚fertige’, lediglich an die Bedürfnisse des Unternehmens anzupassende Lösung, eingeführt werden soll. Der Aufwand für die Systemkonzeption, Systembeschreibung und Programmierung wird durch den Einsatz der Standardsoftware erheblich reduziert bzw. ist zum Teil überhaupt nicht notwendig.“ [MEI95, S.488] Ergebnis der Analyse, das die Software betrifft, ist vor allem der Funktionsabdeckungsgrad der Standardsoftware.

Auch das IBSIS-Modell kümmert sich kaum um die Vorbereitung der Konstruktion und Veränderung von Software. Es wird sogar davon ausgegangen, dass die Software so wenig veränderbar ist, dass die Anwender-Organisation dadurch zur Reorganisation förmlich gezwungen wird: „Auch wenn ein IBSIS durch Parameter und Module flexibel an verschiedenste Organisationsformen angepasst werden kann, so ist doch bekannt, dass ihm bestimmte betriebswirtschaftliche Modelle und damit auch organisatorische Regeln zugrunde liegen, von denen nicht abgewichen werden kann.“ [BAR96, S.46]

2.3.2 Reorganisation

Reorganisation der Anwender-Organisation wird von den betrachteten Vorgehensmodellen unterschiedlich betont. Dabei lässt sich feststellen, dass die Vorgehensmodelle der Standardsoftwareeinführung Reorganisation stark betonen, während Modelle der Softwareneuentwicklung ihn eher nicht betrachten.

Beim IBSIS-Modell steht die Optimierung des Unternehmens und insbesondere der Geschäftsprozesse bei der Einführung von Standardsoftware im Vordergrund. Insbesondere spielt deswegen Business Process Redesign eine zentrale Rolle. Hier werden „wesentliche

Unternehmensprozesse ... grundlegend hinterfragt und radikal umgestaltet ...“ [BAR96, S.1] Dabei können auch „... Prozesse BPR unterzogen werden, obwohl sie vom IBSIS gar nicht berührt werden.“ [BAR96, S.103] Die erwarteten Ergebnisse der Analyse sind deswegen vor allem Prozessmodelle und deren Schwachstellen und Verbesserungspotentiale.

Auch das IMG-Modell fordert BPR: „Die Einführung des Systems R/3 bietet die Chance, verkrustete Strukturen und Abläufe im Unternehmen im Sinne von Business Process Reengineering zu optimieren.“ [MEI95, S.488] Zumindest Meinhardt redet aber nicht von einem Reorganisationszwang.

Der RUP zielt nicht speziell auf Reorganisation ab. Die Erstellung von Usecase-Diagrammen in der Analyse dient nicht der Optimierung von Geschäftsprozessen, sondern Identifiziert die Schnittstellen zum Softwaresystem.

Für das Vorgehensmodell von Helmut Balzert spielt die Software „... eine entscheidende Rolle bei der Realisierung eines Unternehmensmodells." [BALZ98, S.691] Balzert sieht die Unternehmensmodellierung als ingenieurmäßige Tätigkeit. [BALZ98, S.718] Die Analyse des Unternehmens ist Ausgangspunkt für dessen Umgestaltung: "Ausgangspunkt für die Umgestaltung oder Neugestaltung eines Unternehmens ist eine dokumentierte Unternehmensvision. Da in der Regel die Basis für eine Umgestaltung ein existierendes Unternehmen ist, entwickelt sich eine Unternehmensvision in Wechselwirkung mit der Analyse des existierenden Unternehmens.“ [BALZ98, S.722] Konkret bedeutet dies für ihn aber nur, das Usecases ähnlich wie beim RUP erstellt werden. Wie ein Unternehmen optimiert wird und wie die Analyse dabei hilft, bleibt offen.

STEPS sieht einen Zusammenhang zwischen Softwareeinführung und Reorganisation: „Da von einem Zusammenspiel von Softwareeinführung und organisatorischer Veränderung ausgegangen wird, wird Software-Entwicklung als integrativer Teil einer übergreifenden Organisationsentwicklung gesehen.“ [FKRW97, S.14].

2.3.3 Veränderung von Software versus Veränderung von Organisation

Betrachtet man die beiden Tendenzen Reorganisation und Softwarekonstruktion im Zusammenhang, fällt auf, dass besonders dort sehr stark nach Optimierung bzw. Veränderung der Anwender-Organisation gerufen wird, wo die Software möglichst nicht verändert werden soll. Der „off-the-shelf“ Charakter von Standardsoftware wird sogar als Argument genommen, dass das Unternehmen sich verändern muss, weil die Software ja unveränderlich ist. Es entsteht ein durchaus positiv gemeinter „Reorganisationszwang“ [BAR96, S.46].
Insgesamt lässt sich beobachten, dass bei Standardsoftwareeinführung vor allem die Veränderung der Anwender-Organisation durch die Einführung von Standardsoftware herbeigeführt werden soll. Die Veränderung der Software durch Eigenarten der Organisation soll möglichst minimal sein. (Abb. 14)

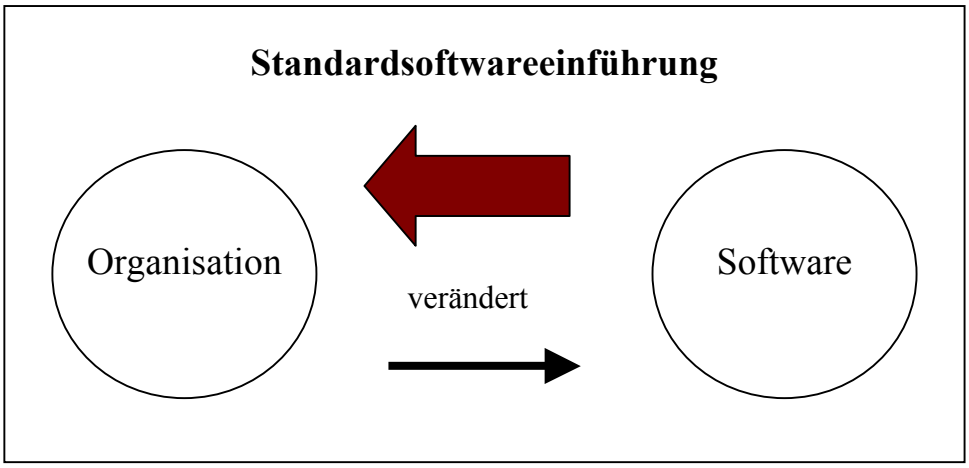


Abb. 14: Bei Standardsoftwareeinführung fließen vor allem Konzepte aus Software in die Organisation

Umgekehrt wollen sich Vorgehensmodelle zur Konstruktion von Software eher aus Reorganisation heraushalten. Insbesondere bei völliger Neuentwicklung dient die Analyse der Anwender-Organisation zu deren optimaler Unterstützung, aber nicht zu deren Veränderung durch Zwänge, die aus der Software entstehen. [KRU99, S.133] Erstellte Unternehmensmodelle zielen auf die Konstruktion von Software ab: „Da ein großer Teil des Unternehmensmodells durch Anwendungssoftware realisiert werden muss, müssen die Konzepte und Notationen kompatibel zu Konzepten und Notationen der Software-Entwicklung sein." [BALZ98, S.717]
Insgesamt lässt sich feststellen, dass aus der Anwender-Organisation vor allem Vorgaben für die Software kommen. Veränderungen der Anwender-Organisation durch die Software werden im Vergleich dazu weniger stark verfolgt. (Abb. 15)

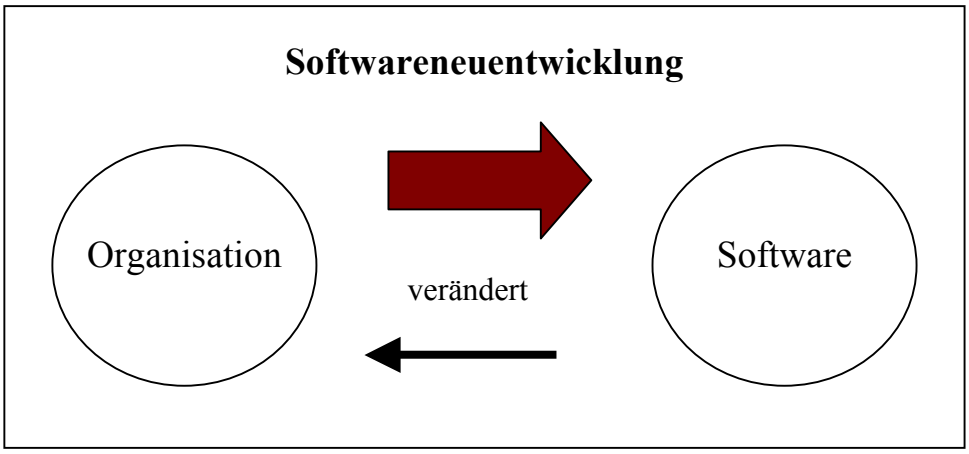


Abb. 15: Bei Softwareneuentwicklung fließen vor allem Konzepte aus der Organisation in die Software

3 Grundlagen und Ergebnisse von Analyse-Methoden

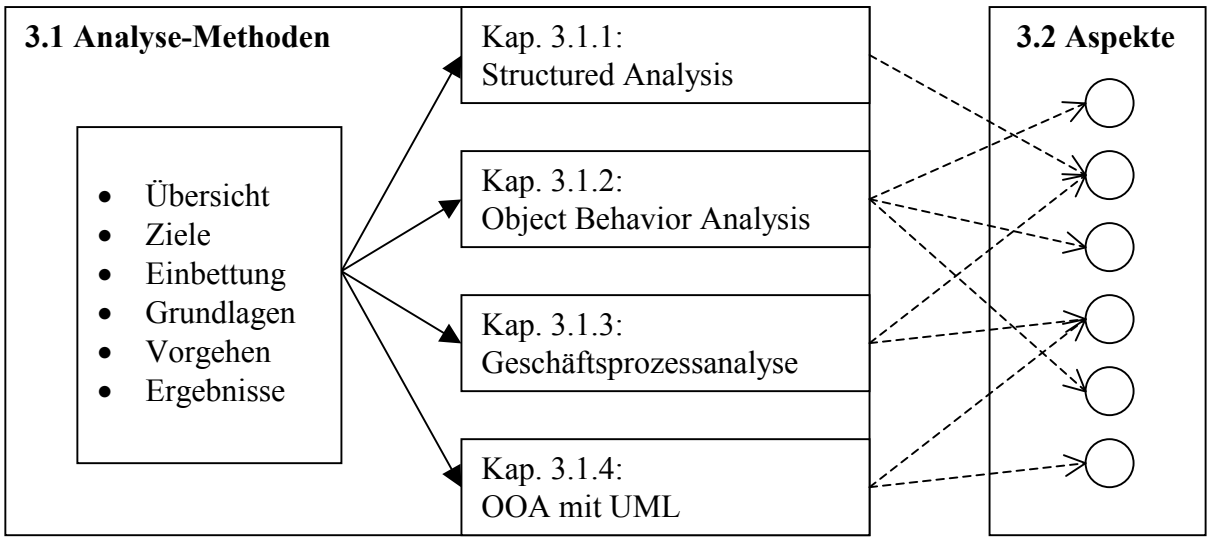


Abb. 16 Gliederung zu Kapitel 3

In diesem Kapitel wird in vier Analyse-Methoden hineingeschaut. Es wird die Vorgehensweise bei der Analyse betrachtet, getrennt von Vorgehensmodellen, denen eine Analyse-Methode vielleicht zugeordnet wird. Dabei werden die Grundlagen und Ergebnisse aus Sicht der Analyse-Methoden festgestellt. Am Ende des Kapitels werden verschiedene Aspekte herausgestellt und für die Analyse-Methoden betrachtet.

3.1 Grundlagen und Ergebnisse einzelner Analyse-Methoden

Folgende Analyse-Methoden werden hier betrachtet:

- Structured Analysis (SA)
- Object Behaviour Analysis (OBA)
- Geschäftsprozessanalyse (GPA)
- Use-Case-gesteuerte Analyse mit UML

Die explizit angegebenen Ziele trennen die Methoden vergleichsweise klar voneinander: SA, OBA und OOA nach Heide Balzert wollen die Entwicklung einer neuen Software unterstützen. SA bereitet dabei strukturiertes top-down Design vor, während OBA und OOA nach Heide Balzert bottom-up ein OO-Design vorbereiten.
GPA verfolgt Reorganisation, insbesondere die Optimierung von Prozessen, im Rahmen der Einführung von Standardsoftware.

Zu jeder Analyse-Methode wird im Folgenden erst eine kurze Übersicht gegeben und ihre allgemeinen Ziele beschrieben. Dann wird aus Sicht der Analyse-Methode deren Einbettung in einen Softwareprozess bzw. ein Vorgehensmodell dargestellt und die Grundlagen sowie Vorgehensweise und Ergebnisse der Methode beschrieben. Die wichtigsten verwendeten Modelle bzw. Diagramme der Methoden werden am Beispiel der Prüfungsverwaltung einer Universität veranschaulicht.

3.1.1 Structured Analysis (SA)

Übersicht

Structured Analysis (SA) wurde bereits 1978 veröffentlich und ist eine der ersten Analyse-Methoden, die sowohl umfassend dokumentiert wurde als auch in der Praxis weite Verbreitung fand. Der Ansatz geht auf Tom DeMarco zurück [DEM79], wurde aber auch von zahlreichen anderen Autoren aufgegriffen und variiert (vgl. z.B. [POM00]). Für Helmut Balzert war SA zumindest 1996 noch „der Industriestandard“ [BALZ96, S.398]. An dieser Stelle wird der Ansatz nach DeMarco von 1978/79 vorgestellt.

Allgemeine Ziele

Die Analyse führt nach DeMarco zur Spezifikation eines neuen Systems, welches später implementiert werden soll. [DEM79, S.4]. Besonders wichtig ist für ihn das Verhältnis des Analytikers zum Benutzer (user liaison). Der Analytiker sollte Lehrer, Übersetzer und Ratgeber des Benutzers sein. Er ist außerdem Mittelsmann zwischen dem Benutzer, der entscheidet, was getan werden muss, und dem Entwickler, der dies realisiert. [S.7]
Desweiteren verfolgt DeMarco folgende Ziele [DEM79, S.15-16]:

- Da Veränderungen natürlich sind, muss auch das Spezifikationsdokument leicht veränderbar sein.
- Große Probleme müssen effektiv aufgeteilt werden.
- Es sollten grafische Notationen verwendet werden.
- Es muss nach logischen und physischen Dingen unterschieden werden.
- Verantwortlichkeiten müssen geregelt sein.
- Es muss ein logisches Systemmodell erstellt werden, das dem Benutzer einen Eindruck des Systems vermittelt, bevor es implementiert wird.

Einbettung der Analyse

Für DeMarco ist die Analyse eine Phase im life cycle. [DEM79, S.4] Structured Analysis bereitet dabei insbesondere Structured Design vor bzw. zieht Tätigkeiten des Structured Design bereits in die Analyse-Phase vor (Abb. 17).

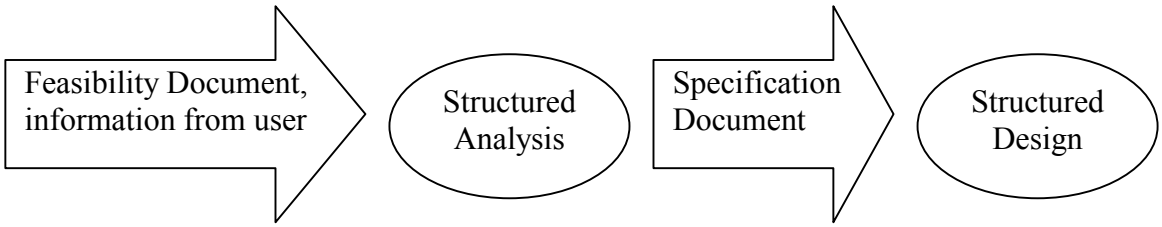


Abb. 17: Structured Analysis bereitet Structured Design vor

Grundlagen

Den Gegenstandsbereich (subject matter) handhabbar zu machen, um ihn auf einzelne Mitarbeiter eines Analyse-Teams aufzuteilen, ist nach DeMarco ein kompliziertes Partitionierungs-Problem [DEM79, S.13]. Vor der Analyse soll eine erste Vermutung über den Projektumfang gewagt und die betroffenen Benutzer identifiziert werden. Die Analyse-Phase basiert auf zwei Grundlagen: der Machbarkeitsstudie (feasibility document) und den Informationen vom Benutzer (direct information flow from the user). Die Machbarkeitsstudie klärt, ob es wirklich einen neuen Weg gibt, wie ein Unternehmen organisiert werden kann (new procedures, new devices, new business areas), und ob dieser im Verhältnis zu den

Kosten steht. Sie bestimmt dann die Parameter des Projekts. [DEM79, S.19] Die Analyse soll dabei durchgeführt werden, ohne dass ständig an die in der Machbarkeitsstudie angestrebten Veränderungen gedacht wird. [DEM79, S.27]
Der für die Erstellung der Datenflussdiagramme zu betrachtende Kontext soll groß genug gewählt werden, um alle für die Entwicklung relevanten Dinge einzuschließen, aber klein genug, um irrelevante Dinge auszuschließen. [DEM79, S.63]

Vorgehen & Ergebnisse

Das wichtigste Produkt der Analyse ist das Spezifikationsdokument, welches ein Modell des zu erstellenden Systems ist. [DEM79, S.7] Eine sinnvolle Partitionierung des Spezifikationsdokuments ist dabei besonders wichtig: „We have to stop writing Victorian novel specifications, enormous documents that can only be read from start to finish. Instead, we have to learn to develop dozens or even hundreds of 'mini-specifications.'" [DEM79, S.13]
Das Spezifikationsdokument besteht im Einzelnen aus:

- Data Flow Diagrams
- Data Dictionary
- Structured English
- Decision Tables
- Decision Trees

Durch Interviews, Walkthroughs und „enge Zusammenarbeit mit dem Benutzer" lernt und dokumentiert der Analytiker die „Art und Weise, wie die Dinge zur Zeit verrichtet werden". Es wird dabei nicht die Sicht eines Benutzers eingenommen, sondern die Sicht der Daten. [DEM79, S.27] Zentral ist deswegen die Erstellung von Datenflussdiagrammen (DFD).
Der Gegenstandsbereich (physical environment) wird zuerst in ein physisches DFD überführt, welches noch physische Charakteristika enthalten darf: department names, locations, organizations and service bureaus, people's names, files and data stores, procedures, devices, existing automated and electromechanical systems [DEM79, S.230+231]. Das physische DFD wird dann in ein logisches DFD umgewandelt, indem die physischen Charakteristika eliminiert werden. Auf dessen Basis wird eine neue logische Umgebung definiert, der physische Eigenschaften hinzugefügt werden, um verschiedene Entwürfe neuer physischer DFDs zu produzieren. Für die verschiedenen DFD-Varianten werden dann Zeit und Kosten quantifiziert und schließlich eine Option ausgewählt. Das ausgewählte neue physische DFD plus Dokumentation werden dann zum Teil des Spezifikationsdokuments.
In Datenflussdiagrammen fließen Datenflüsse (data flows) von Prozess zu Prozess (process).

> „A data flow is a pipeline through which packets of information of known composition flow." [DEM79, S.52]
>
> "A process is a transformation of incoming data flow(s) into outgoing data flow(s)." [DEM79, S.56]

In Abb. 18 ist ein logisches Datenflussdiagramm abgebildet, in dem das Zulassungsverfahren zu Prüfungen abgebildet ist. Ein Studierender reicht einen Zulassungsantrag beim Prüfungsausschuss ein, der dort auf Übereinstimmung mit der Prüfungsordnung geprüft und genehmigt oder abgelehnt wird. Den genehmigten Zulassungsantrag reicht der Studierende dann beim Prüfungsamt ein, wo kontrolliert wird, ob er die im Zulassungsantrag angegebenen Veranstaltungen besucht hat. Auch hier wird der Antrag entweder abgelehnt

oder angenommen. Beim logischen DFD interessiert nur der Fluss der Daten. Reihenfolge und Abhängigkeiten zwischen den Datenflüssen interessieren nicht. Auch die Organisationseinheiten, in denen die Prozesse ablaufen, werden nicht mitmodelliert.

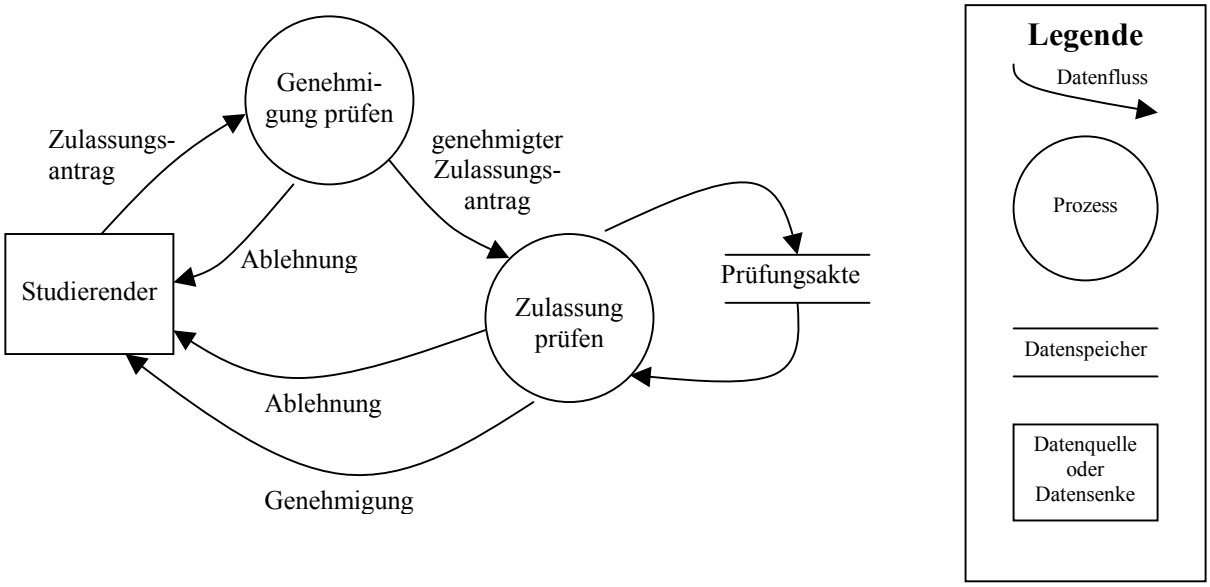


Abb. 18: logisches Datenflussdiagramm Zulassungsantrag

Datenflussdiagramme sollen durch ihre Sicht auf Daten den Gesamtzusammenhang eines Unternehmens in den Blick nehmen: „The advantage of this approach is that the data sees the big picture, while the various people and machines and organizations that work on the data see only a portion of what happens.“ [DEM79, S.49] Dabei werden insbesondere auch Unternehmensteile mit DFDs modelliert, die nicht automatisiert werden sollen: „...consider the context of analysis to be whatever we propose to change plus enough surrounding procedure to buffer the changed area from activity on the outside.” [DEM79, S.60] Der Kontrollfluss wird in DFDs nicht modelliert. [S.54] DFDs werden in verschiedenen Abstraktionsebenen benutzt und als Diagrammhierarchie angelegt, bei der hinter Prozessen der höheren Abstraktionsebene ein DFD der niedrigeren steht.
Ein Eintrag im Data-Dictionary könnte folgendermaßen aussehen:

Studenten-Daten = Name + Adresse + Matrikelnummer + Fachsemester

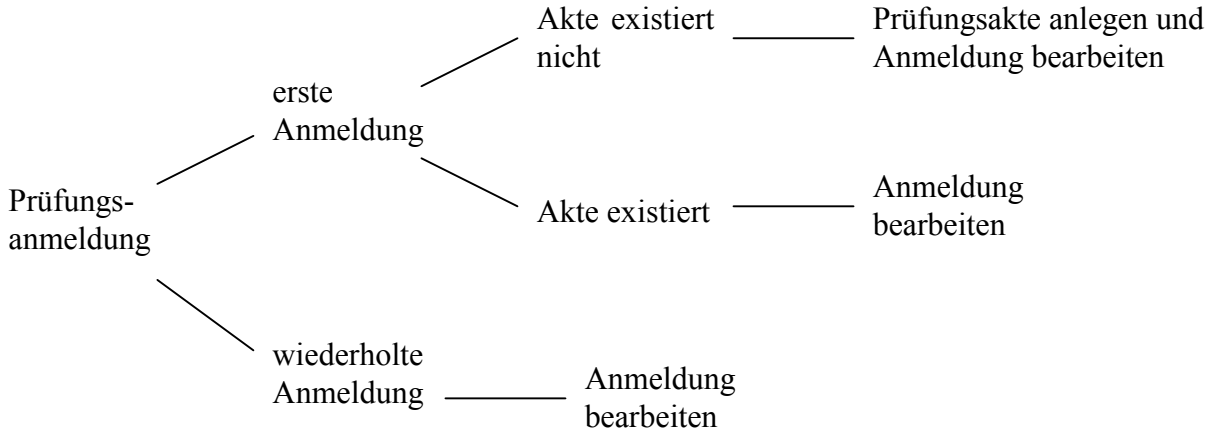
Structured English ist die Vorstufe zum strukturierten Programmablauf. Ein Beispiel für Structured English wäre:

```
If erste Prüfungsanmeldung
        If Prüfungsakte existiert
                Prüfungsanmeldung bearbeiten
        Else (Prüfungsakte existiert nicht)
                Prüfungsakte anlegen und Prüfungsanmeldung bearbeiten
Else (zweite oder spätere Prüfungsanmeldung)
        Prüfungsakte anlegen und Prüfungsanmeldung bearbeiten
```

Dieses Beispiel kann auch als Decision Table dargestellt werden:

	RULES			
CONDITIONS	1	2	3	4
erste Prüfungsanmeldung	Y	N	Y	N
Prüfungsakte existiert	Y	Y	N	N
ACTIONS				
Prüfungsanmeldung bearbeiten	Y	Y	Y	Y
Prüfungsakte anlegen	N	N	Y	Y

Als Decision Tree dargestellt sieht dies folgendermaßen aus:



Zusammenfassung

Die Analyse-Methode Structured Analysis (SA) bereitet die strukturierte Programmierung von Software durch die Erstellung eines logischen strukturierten Modells vor. Die Erstellung des logischen Modells wird top-down durchgeführt, indem physische Charakteristika eliminiert werden. Zentral sind Datenflussdiagramme, die explizit nicht die Sicht von Benutzern einnehmen, sondern die Sicht der Daten in den Vordergrund stellen.
SA sieht sich als eine Phase in einem Phasenmodell (life-cycle-model).

Grundlagen	• Vermutung über Projektumfang • Machbarkeitsdokument • betroffene Benutzer • Informationen von den Benutzern • department names, locations, organizations and service bureaus, people's names, files and data stores, procedure, devices, existing automated and electromechanical systems
Tätigkeiten	• Interviews • Walkthroughs • enge Zusammenarbeit mit dem Benutzer
Modelle etc.	Data Flow Diagrams (DFD), Data Dictionary, Structured English, Decision Tables, Decision Trees
Ergebnisse	Specification Document
Ziele	• Spezifikation eines Systems, welches implementiert wird • leicht veränderbares Spezifikationsdokument • effektive Aufteilung großer Probleme • grafische Notationen • Unterscheidung in logische und physische Dinge • geregelte Verantwortlichkeiten • logisches Systemmodell, das dem Benutzer einen Eindruck des Systems vermittelt, wird vor der Implementierung erstellt

Tab. 14: Structured Analysis – Grundlagen und Ergebnisse

3.1.2 Object Behavior Analysis (OBA)

Übersicht

Object Behavior Analysis (OBA) ist eine Methode zur objekt-orientierten Analyse (OOA). Die Autoren von OBA, Rubin und Goldberg, definieren OOA wie folgt: "Object-oriented analysis endeavors to model a situation in terms of a collection of interacting entities, each of which provides a well-defined set of behaviors and attributes" [RUG92, S.48]

Allgemeine Ziele

Ein Ziel von OBA ist, dass Analyse in einer vorhersagbaren und kontrollierbaren Weise durchgeführt wird [RUG92, S.48]. Desweiteren schreiben Rubin und Goldberg: „... our goal ... is that we have a clear understanding of the behaviors exhibited by the system, the objects that exhibit these behaviors, the relationships among the objects, and how the objects interact with one another … In addition, any implementation code must be traceable back to the results of the analysis." [S.48]
"Our goal is to be able to answer the question: Which roles and responsibilities are needed in order to accomplish the required tasks?" [S.49]

Einbettung der Analyse

„OBA is part of a larger process model incorporating the specific engineering opportunities introduced by object-oriented technology" [RUB92, S.49]
"We assume that analysis is carried out in the context of a project process model which includes a system of periodic reviews." [S.55]

Grundlagen

Ausgangspunkt für OBA ist eine Problemsituation. [RUB92, S.49] Durch Konzentration auf Verhaltensweisen (Behaviors) eines Systems kann festgestellt werden, was im System geschieht [S.48]. Die Behaviors werden verschiedenen Teilen des Systems zugeordnet.

Vorgehen & Ergebnisse

OBA besteht aus fünf iterativen Schritten:

0. Bestimmen des Kontextes der Analyse
1. Problemverstehen durch Konzentration auf Behaviors
2. Objekte definieren, die Behaviors ausdrücken
3. Objekte klassifizieren und ihre Beziehungen identifizieren
4. Modellieren von System-Dynamiken

Im 0. Schritt werden Ziele und bestehende Dokumente sowie relevante Akteure identifiziert, die zur Analyse beitragen können. Außerdem werden die zu analysierenden Kernbereiche der Aktivitäten festgestellt und ein erster Analyseplan wird vorbereitet, der die Bereiche der Aktivitäten partitioniert, priorisiert und erste Zeit und Ressourcenabschätzungen macht.

Im 1. Schritt wird bestimmt, was das System für wen leisten soll. Rubin und Goldberg gehen hier von der Erstellung von Szenarios (use scenarios) aus [RUG92, S.51], die sie mit der Methode der Use-Case-Erstellung nach Jacobson vergleichen [JAC92]. Dazu empfehlen sie strukturierte Interviews mit Benutzern und Domänenexperten. Die Szenarien werden dann in tabellarischer Form strukturiert (Abb. 19), die auch für die Benutzer verständlich sein soll. Jede Zeile der Notation ist ein Vertrag (contract) zwischen einem Initiator und einem Participant (Teilnehmer). Initiatoren und Participants, die eine signifikante Rolle im System spielen, werden als Objekte identifiziert.
Bspw. will sich ein Studierender zu einer Prüfung anmelden. Dazu lässt er sich von der elektronischen Prüfungsanmeldung freie Termine ausgeben. Die Prüfungsanmeldung bietet also den Service an, freie Termine anzuzeigen.
Desweiteren werden Glossare für Parties (Initiatoren, Participants), Services und Attribute erstellt. Schritt 1 dient außerdem dazu, ein gemeinsames Vokabular zwischen Analytikern und Interviewten zu etablieren.

Im 2. Schritt werden die Participants ausgewählt, die zu Analyse-Objekten werden sollen. Jeder Participant ist ein potentielles Objekt, da er einen Service anbietet. Mögliche Objekte werden auf Object Modeling Cards (erweiterte CRC-Karten) festgehalten. CRC-Karten enthalten den Objekt-Namen, Vorfahren, zusätzliche Informationen und Behaviors, Attribute, Services und eine Art Historie (card trace). [RUG92, S.55ff]

WWW-Prüfungsanmeldung – Schritt 1			
Script Name: Modifikation.1.Beispiel Autor: D. Adams Version: 4.2 Vorbedingung: WWW-Seite für Anmeldung wird angezeigt Nachbedingung: Prüfungsanmeldung abgeschlossen oder abgebrochen Trace: Kern Aktivität – Modifikation			
Initiator	**Action**	**Participant**	**Service**
Studierender	Auswahl Haupt-/Grundstudium	Radiobutton	select a button
Studierender	Eingabe NAME	name-field	set content to text
Studierender	Auswahl Prüfung	Prüfungsliste	select row
Studierender	click Okay-Button	Okay-Button	activate button
Studierender	Auswahl eines Termins	Terminliste	select row
Studierender	click Okay-Button	Okay-Button	activate button
Studierender	Auswahl eines Prüfers	Prüferliste	select row
Studierender	click Okay-Button	Okay-Button	activate button

Abb. 19: Script Notation nach OBA

Im 3. Schritt werden die Beziehungen zwischen den Objekten festgestellt. Nun werden die Objekte reorganisiert (abstraction, specialization, factorization), so dass z.B. Oberklassen gefunden werden oder ein Objekt in mehrere aufgespalten wird. [S.58f]

Im 4. Schritt werden dynamische Aspekte modelliert. Dazu gehören Zustände der Objekte und ein Zustandsdiagramm (empfohlen: Harel Statechart) für jedes Objekt, das beschreibt, durch welche Ereignisse es von einem zum nächsten Zustand wechselt. Der Kontrollfluss muss durch eine Notation modelliert werden, die beliebige Reihenfolgen der Aktivitäten zulässt (concurrently, repetitively, selectively, optionally, sequentially). Rubin und Goldberg verweisen auf Petri-Netze, Action Diagrams oder Statecharts, ohne aber näher auf deren Einsatz einzugehen.

Ergebnisse von OBA sind [RUG92, S.48-49]:

- Beschreibungen der erwarteten Systembenutzung
- Glossare von Party Names (Initiators, Participants), Participants' services, Attribute, Zustandsdefinitionen
- Objektmodelle mit hierarchischen und vertraglichen (contractual) Beziehungen
- System-dynamische Modelle: Object life cycles, Sequencing of operations

"The analysis result should be understandable to the end user, lend itself to further design and implementation, and be traceable to system goals and objectives." [RUG92, S.49]

Zusammenfassung

Die objekt-orientierte Analyse-Methode OBA will, durch Konzentration auf Verhaltensweisen (Behaviors) von Objekten, Situationen in vorhersagbarer und kontrollierbarer Weise modellieren. Die Methode führt von Szenarios über sequentielle Abläufe in Tabellen zu Objektmodellen, auf die objektorientiertes Design aufsetzen soll.
OBA sieht sich als Teil eines objektorientierten iterativen Prozessmodells.

Grundlagen	Problemsituation, bestehende Dokumente, relevante Akteure, Kernbereiche der Aktivitäten
Tätigkeiten	• Bestimmen des Kontextes der Analyse • Erstellung von Szenarios • strukturierte Interviews von Benutzern und Domänenexperten • Konzentration auf Verhaltensweisen (Behaviors) von Objekten • Objekte definieren, klassifizieren und ihre Beziehungen identifizieren • Modellieren von System-Dynamiken (Verweis auf Literatur)
Modelle etc.	Object Modeling Cards, Szenarios, Modelle für dynamische Aspekte (Petri-Netze, Action Diagrams, Statecharts)
Ergebnisse	• Analyseplan • gemeinsames Vokabular zwischen Analytikern und Interviewten • Beschreibungen der erwarteten Systembenutzung • Glossare von Party Names (Initiators, Participants), Participants' services, Attributen, Zustandsdefinitionen • Objektmodelle mit hierarchischen und vertraglichen Beziehungen • System-dynamische Modelle: Object life cycles, Sequencing of operations
Ziele	• vorhersagbare und kontrollierbare Weise der Durchführung • Verständnis der Verhaltensweisen des Systems, der Objekte und deren Beziehungen und Interaktionen • Implementations-Code muss auf die Ergebnisse der Analyse zurückführbar sein • Beantwortung der Frage: Welche Rollen und Verantwortlichkeiten werden gebraucht, um die benötigten Aufgaben zu erledigen? • Analyse-Ergebnisse sollen vom Endbenutzer verstanden werden können, einen Beitrag zu Design und Implementation beitragen und auf die Ziele (goals and objectives) zurückführbar sein.

Tab. 15: Object Behavior Analysis – Grundlagen und Ergebnisse

3.1.3 Geschäftsprozessanalyse (GPA)

Übersicht

Staud bietet eine Methode zur Geschäftsprozessanalyse an, die insbesondere zur Unternehmensmodellierung verwendet werden soll. Die Modellierung wird als Abbildung von „betrieblichen Weltausschnitten" in Software betrachtet. [STA99, S.18] Im Vordergrund steht die Verwendung von Ereignisgesteuerten Prozessketten (EPK).

Allgemeine Ziele

Zur Unternehmensmodellierung werden die betrieblichen Abläufe mit dem Konzept der Geschäftsprozesse erfasst und formal beschrieben. Dabei geht es letztendlich um Optimierung der Unternehmensabläufe. [STA99, S.7,9] Insbesondere einzelaufgabenorientierte Arbeitsplätze sollen überwunden und prozessorientiert organisiert werden. [S.9]

> „Ein Geschäftsprozess besteht aus einer zusammenhängenden abgeschlossenen Folge von Tätigkeiten, die zur Erfüllung einer betrieblichen Aufgabe notwendig sind. Die Tätigkeiten werden von Aufgabenträgern in organisatorischen Einheiten unter Nutzung der benötigten Produktionsfaktoren geleistet. Unterstützt wird die Abwicklung der Geschäftsprozesse durch das Computergestützte Informationssystem (CIS) des Unternehmens.“ [STA99, S.6]

Einbettung der Analyse

Geschäftsprozesse sind von zentraler Bedeutung für betriebswirtschaftliche Standardsoftware, „... da diese prozessorientiert ist in dem Sinne, dass sie integriert die gesamten Geschäftsprozesse eines Unternehmens unterstützt und nicht nur einzelne Funktionen ...“ [STA99, S.21] „Grundlage betriebswirtschaftlicher Standardsoftware ist ... eine möglichst umfassende Analyse der in den Unternehmen konkret vorliegenden Geschäftsprozesse.“ [S.26]
„Die Betrachtung von Geschäftsprozessen ist verbunden mit einer umfassenden integrierten Analyse des gesamten Umfelds, in dem sie abgewickelt werden: der benutzten Informationsobjekte, der Informationsflüsse, der Einzeltätigkeiten (später Funktionen genannt), der Organisationsstrukturen, usw.“ [STA99, S. 6]
Unternehmensmodellierung ist die Voraussetzung für eine umfassende DV-Durchdringung und die Einführung betriebswirtschaftlicher Standardsoftware. [STA99, S.2] Dabei geht es letztendlich um Optimierung der Unternehmensabläufe. [STA99, S.7,9]
Staud trennt die Einführung von Standardsoftware in die Zeit vor dem Kauf und die Zeit nach dem Kauf. [STA99, S.36ff] Vor dem Kauf können folgende Aktivitäten stattfinden:

- Abgrenzung und Festlegung der Ziele
- Ist-Analyse
- Optimierungsziele festlegen
- Inaugenscheinnahme der Standardsoftware
- Softwareauswahl

Nach dem Kauf folgen die Aktivitäten:

- Vergleich der realen Geschäftsprozesse mit denen der Standardsoftware unter Mitberücksichtigung der Optimierungsziele
- Konkrete Einführung

Staud verweist insbesondere auf das Vorgehensmodell von SAP [STA99, S.38].

Grundlagen

„Mit dem Konzept des Geschäftsprozesses ... stehen längere zusammenhängende Folgen von Tätigkeiten, die zur Erledigung einer größeren Aufgabe nötig sind, im Mittelpunkt.“ [STA99, S.5]
Ausgangspunkt ist eine betriebswirtschaftliche Problemstellung, die „ ... z.B. aus den Strukturen und Abläufen eines Unternehmens besteht ...“. Die Problemstellung wird in Fachkonzepte der Organisation, Daten, Steuerung und Funktionen abgebildet. [STA99, S.19]

Vorgehen & Ergebnisse

Geschäftsprozessmodelle sind das zentrale Ergebnis der Analyse. Dabei ist im Detaillierungsgrad „... alles möglich, von einer hoch angesiedelten Meta-Ebene bis zu einer

tayloristischen Ausdifferenzierung der einzelnen Tätigkeiten.“ [STA99, S.10] „Die Entscheidung, wo die Grenzen einzelner Geschäftsprozesse gezogen werden, ist die des Analysten und hängt nur von seiner Kompetenz und der Zielsetzung der Analyse ab.“ [S.10] Es wird eine „... veränderte Sichtweise bzgl. der Strukturen und Abläufe in den Unternehmen eingenommen ..., wenn die Analyse auf dem Geschäftsprozesskonzept beruht.“ [STA99, S.2] Staud hebt die Geschäftsprozesskomponenten nach Scheer hervor [S.13], die aus folgenden Elementen besteht: Vorgänge, Ereignisse, Zustände, Bearbeiter, Organisationseinheiten, Ressourcen der Informationstechnologie. Diese werden graphisch in ereignisgesteuerten Prozessketten abgebildet (EPKs: Abb. 20).

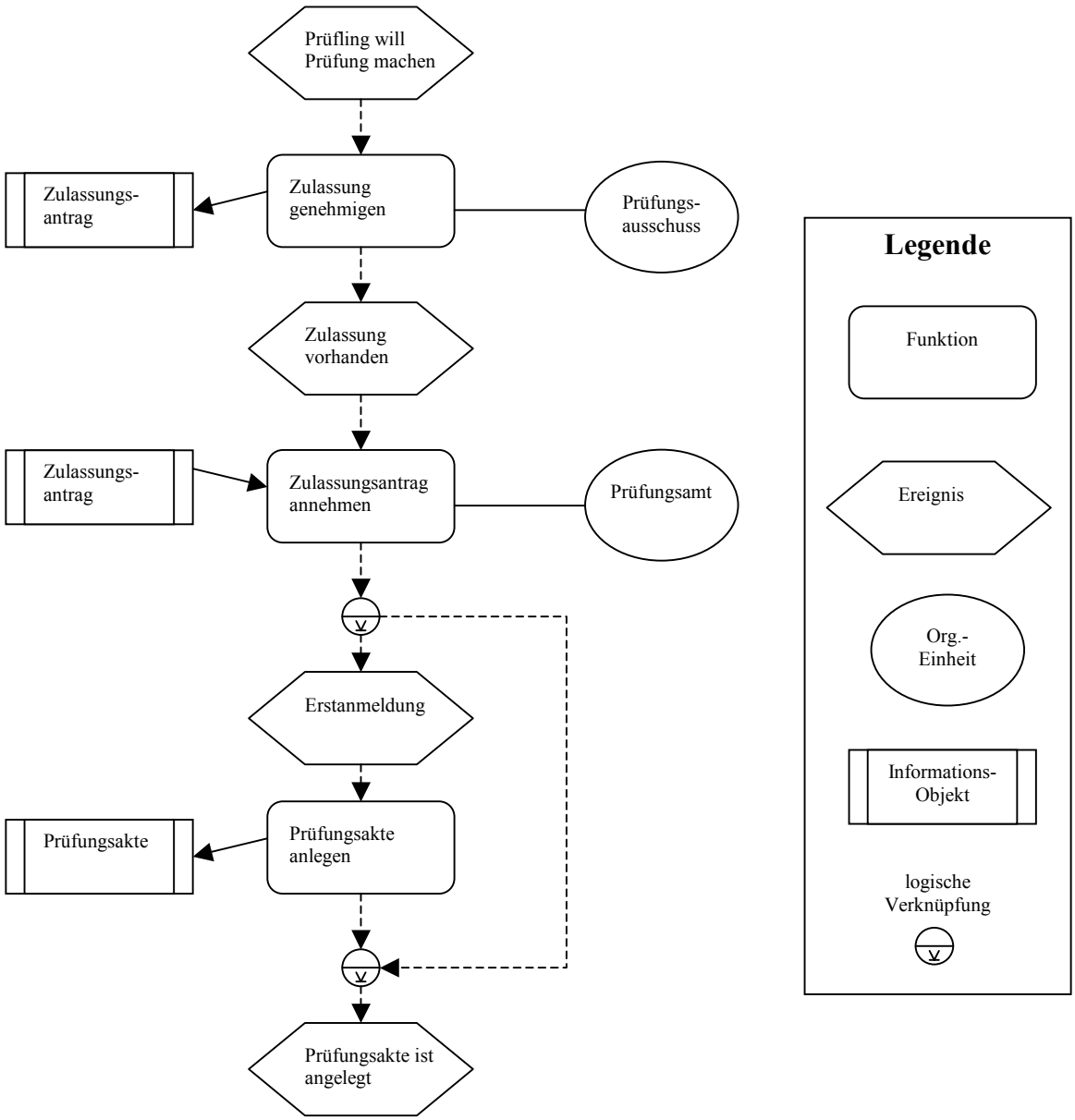


Abb. 20: Ausschnitt aus Ereignis-Prozess-Kette für Prüfung

Ein Geschäftsprozess ist für Staud „... kein Programm, sondern eine von Menschen getragene Abfolge von Tätigkeiten.“ [STA99, S.88] Die EPKs legen den zeitlichen Ablauf der Funktionen fest. Sie geben an, in welcher Reihenfolge Funktionen ausgeführt werden.

Ereignisse legen fest, wann eine Funktion ausgeführt wird. Mit EPKs können Dinge wie Warten [S.93] oder bestimmte Termine (Monatsende, Mahntermin, Frist [S.97]) modelliert werden. Der Kontrollfluss bewegt sich abhängig vom Eintritt der Ereignisse zeitlich fort oder bleibt stehen. Zeitverbrauch („diese Funktion muss innerhalb eines Tages erledigt sein“) wird dagegen „... im Regelfall nicht erfasst.“ [S.97/98]
Für Staud ist außerdem „die Betonung der Praxisnähe ... wichtig, da dabei andere Ereignisgesteuerte Prozessketten entstehen, als wenn entlang einer Software modelliert wird.“ [S.103]
Dabei muss für ihn eine „intensive Diskussion zwischen den Prozessverantwortlichen bzw. denen, die am Geschäftsprozess teilhaben und den Modellierern“ stattfinden, was oft im Rahmen von mehrtägigen Workshops geschieht.
„Das eigentliche Ziel der Beschäftigung mit Geschäftsprozessen ist ihre optimierte Neugestaltung. Die Optimierungsrichtung muss sich an den Zielen des Unternehmens orientieren.“ [S.173] Das sind vor allem die Formalziele:

- kurze Durchlaufzeiten
- minimale Kosten
- größtmögliche Qualität

Optimieren lassen sich besonders Medienbrüche und Organisationsbrüche.
Bei Medienbrüchen wird dieselbe Information nacheinander auf verschiedene Informationsträger gebracht. So werden z.B. Informationen sowohl in einer Datenbank erfasst als auch in einer Textverarbeitung. Oder Daten werden auf mehreren Datenträgern gespeichert. Oder Daten müssen manuell von verschiedenen Programmen bzw. Masken abgetippt werden.
Medienbrüche können bestimmt werden, indem die Informationsobjekte in den EPKs sorgfältig und detailliert mitmodelliert werden. Dabei werden dann auch gleichzeitig die Informationsflüsse mitmodelliert.
Organisationsbrüche können zum Teil durch das Erfassen der Organisationseinheiten in den EPKs festgestellt werden. „Damit ist gemeint, dass ein Geschäftsprozess unnötig oft die Organisationseinheit bzw. die bearbeitende Person wechselt.“ [STA99, S.175]
Auch hier kommt es auf detaillierte Modellierung an. Da der Wechsel der bearbeitenden Person oder Abteilung oft aus Kontrollzwecken (Vier-Augen-Prinzip etc.) erfolgt, ergibt sich „...ein Widerspruch zwischen Optimierung und Kontrolle.“ Dies ist z.B. bei genehmigungsbedürftigen Beschaffungsvorgängen der Fall.
Wenn die Prozesse verworren sind oder „wild wuchern“, obwohl nicht auf einer zu niedrigen Ebene modelliert wurde, kann das bedeuten, dass die Verantwortlichen den Überblick verloren haben und es keinen Prozessverantwortlichen gibt. Ein solcher sollte dann eingerichtet werden und sich den Überblick verschaffen, um den Wildwuchs zu beseitigen. [STA99, S.175]
Die Koordination von Geschäftsprozessen über Unternehmensgrenzen hinweg birgt für die Zukunft ein großes Optimierungspotential. Bei der Optimierung von Prozessen sollte aber darauf geachtet werden, dass nicht in zu hohem Maße zu Ungunsten von Geschäftspartnern optimiert wird. [STA99, S.176]

Zusammenfassung

Geschäftsprozessanalyse (GPA) will einzelplatzübergreifende Unternehmensprozesse modellieren. Ziel ist die Überwindung einzelaufgabenorientierter Arbeitsplätze und die Optimierung der Geschäftsprozesse. Dabei wird insbesondere auch von der Abwicklung der

Prozesse durch Software ausgegangen. Zentrales Modellierungsmittel für die Erstellung der Geschäftsprozessmodelle sind Ereignisgesteuerte Prozessketten.
GPA sieht sich insbesondere als Voraussetzung für die Einführung von Standardsoftware.

Grundlagen	betriebswirtschaftliche Problemstellung (besteht z.B. aus Strukturen und Abläufen eines Unternehmens)
Tätigkeiten	Betriebliche Abläufe erfassen und formal beschreiben.
Modelle etc.	Geschäftsprozessmodelle: Ereignisgesteuerte Prozessketten (EPK)
Ergebnisse	• Geschäftsprozessmodelle • veränderte Sichtweise bzgl. der Strukturen und Abläufe im Unternehmen
Ziele	• Optimierung der Unternehmensabläufe • Überwindung einzelaufgabenorientierter Arbeitsplätze • prozessorientierte Organisation • Formalziele: - kurze Durchlaufzeiten - minimale Kosten - größtmögliche Qualität • Prozessoptimierung**:** - Medienbrüche - Organisationsbrüche

Tab. 16: Geschäftsprozessanalyse – Grundlagen und Ergebnisse

3.1.4 Objektorientierte Analyse (OOA) nach Heide Balzert

Übersicht

Heide Balzert bietet eine objektorientierte Analyse-Methode an, die auf einer Erweiterung der Unified Modeling Language (UML) basiert [BAL99]. UML hat sich mittlerweile weltweit als Modellierungssprache in der Softwareentwicklung etabliert. Sie wird auch im RUP-Vorgehensmodell verwendet und in der neuen Auflage des Vorgehensmodells von Helmut Balzert.

Allgemeine Ziele

„Ziel der Analyse ist es, die Wünsche und Anforderungen eines Auftraggebers an ein neues Softwaresystem zu ermitteln und zu beschreiben. Es muss ein Modell des Fachkonzepts erstellt werden, das konsistent, vollständig, eindeutig und realisierbar ist.“ [BAL99, S.8] Dabei werden Aspekte der Implementierung bewusst ausgeklammert. Die Analyse wird in einem „kontinuierlichen Prozess“ durchgeführt, in dem Information gesammelt, gefiltert und dokumentiert werden. Der Analytiker muss sich dem Auftraggeber verständlich machen.
„Ziel der objektorientierten Analyse ist es, das zu realisierende Problem zu verstehen und in einem OOA-Modell zu beschreiben. ... Es darf keinerlei Optimierungen für das verwendete Computersystem oder die benutzte Basissoftware enthalten.“ [S.8+9] „Das OOA-Modell beschreibt die essentielle Struktur und Semantik des Problems, aber noch keine technische Lösung.“ [BAL99, S.15]

Einbettung der Analyse

Objektorientierte Analyse ist für Heide Balzert eine Phase der objektorientierten Softwareentwicklung.

Die Analyse-Methode soll einen evolutionären Entwicklungsprozess unterstützen: „Das bedeutet, dass zunächst eine objektorientierte Analyse für den Produktkern erstellt wird, der anschließend zu entwerfen und zu implementieren ist. Dieser Kern wird in weiteren Zyklen erweitert, bis ein auslieferbares System entsteht.“ [BALZ99, S.122]

Grundlagen

Im Speziellen wird bei der Objektorientierten Analyse (OOA) von Objekten in der „realen Welt“ ausgegangen. „Aus einem realen Objekt wird durch Modellbildung und geeignete Abstraktion ein Objekt unseres objektorientierten Modells.“ [BAL99, S.8]
Der während der Analyse erstellte Prototyp wird dem Auftraggeber vorgelegt und dient somit der Rückkopplung und weiteren Verbesserung des OOA-Modells.
Die Daten des Systems können aus Formularen, Dateibeschreibungen, Listen, Benutzerhandbüchern und Bildschirmmasken eines alten Systems (soweit vorhanden) und anderen Dokumenten entnommen werden oder durch Interviews festgestellt werden. [BAL99, S.121]

Vorgehen & Ergebnisse

Zuerst wird ein Pflichtenheft erstellt, das dann als Ausgangsbasis für eine systematische Modellbildung dient. „Das Pflichtenheft ist eine textuelle Beschreibung dessen, was das zu realisierende System leisten soll.“ [BAL99, S.9]
Heide Balzert unterscheidet Basiskonzepte (Objekt, Klasse, Attribut, Operation), statische Konzepte (Assoziation, Vererbung, Paket) und dynamische Konzepte (Geschäftsprozess, Botschaft, Szenario, Zustandsautomat).

Das OOA-Modell (Analysemodell) wird auf Basis des Pflichtenhefts erstellt und „...bildet die fachliche Lösung des zu realisierenden Systems“ (Fachkonzept). Es besteht aus statischem und dynamischem Modell. Das statische Modell beschreibt insbesondere Klassen des Systems und deren Beziehungen. Das dynamische Modell zeigt Funktionsabläufe. Dazu gehören Geschäftsprozesse, Szenarios und Zustandsautomaten. Bei großen Projekten werden Pakete gebildet, die jeweils nur einen Teil des Systems betrachten.

Erster Schritt der OOA ist die Identifikation der relevanten Geschäftsprozesse, in Interviews mit Benutzerrepräsentanten und Experten.

> „Ein Geschäftsprozess (use case) besteht aus mehreren zusammenhängenden Aufgaben, die von einem Akteur durchgeführt werden, um ein Ziel zu erreichen bzw. ein gewünschtes Ergebnis zu erstellen.“ [BAL99, S.63]

„Geschäftsprozesse sind ein reines Analyse-Konzept.“ [BAL99, S.131] Die Prozesse werden auf einer hohen Abstraktionsebene beschrieben und sollten auch vom Auftraggeber verstanden werden können. Es interessiert die Kommunikation der Akteure mit dem System. Interne Struktur und Algorithmen interessieren dagegen nicht. Zur besseren Übersicht werden Geschäftsprozessdiagramme erstellt (Abb. 21). „Wenn ausgedrückt werden soll, dass bestimmte Verarbeitungsschritte aus fachlicher Sicht parallel ausgeführt werden können, dann ist das Aktivitätsdiagramm anzuwenden.“ [BAL99, S.127] Heide Balzert gibt konkrete Checklisten (Fragenkataloge, Fehlerquellen etc.) zur Identifizierung von Akteuren und Geschäftsprozessen.

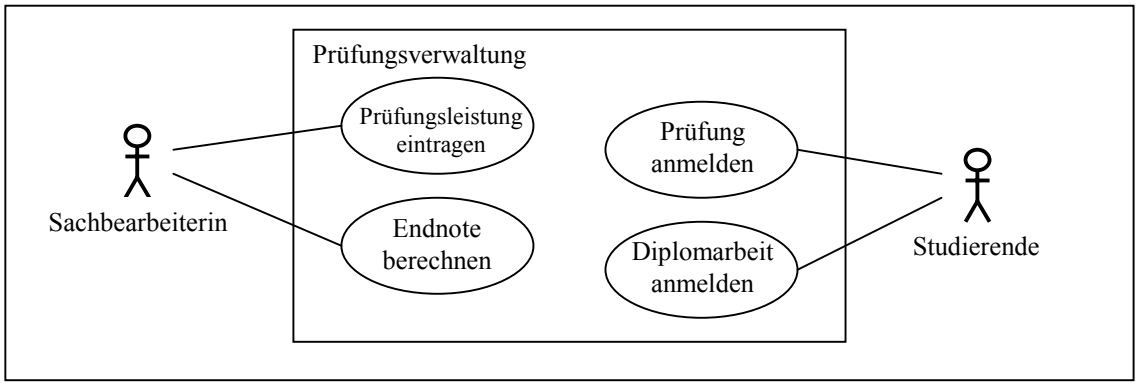


Abb. 21: Ausschnitt aus Geschäftsprozessdiagramm

Dann wird das statische Modell aus den Geschäftsprozessen top-down abgeleitet. Dabei soll nach Substantiven und Verben gesucht werden. „Der Erfolg dieser Methode wird entscheidend durch die Sicherheit des Systemanalytikers im Erkennen der potentiellen Klassen bestimmt." [BAL99, S.143]
Soweit vorhanden wird auf Beschreibungen des alten Softwaresystems zurückgegriffen, die dann mittels Dokumentanalyse bearbeitet werden: dabei werden Attribute „... mittels bottom-up-Vorgehen zu Klassen zusammengefasst ...„ und ein Klassenname nach der Gesamtheit der Attribute gewählt [BAL99, S.142]. Es entstehen Klassendiagramm, Kurzbeschreibung der Klassen, Objektdiagramm und Attributspezifikation. [BAL99, S.123]

Darauf wird das dynamische Modell erstellt, das mit dem statischen Modell rückgekoppelt wird. Es entstehen Sequenzdiagramm, Kollaborationsdiagramm, Zustandsdiagramm, Klassendiagramm und fachliche Beschreibung. [BAL99, S.124] Die Geschäftsprozesse werden hier mit Szenarios verfeinert, die durch Interaktionsdiagramme dokumentiert werden. Szenarios helfen beim Identifizieren der Operationen der Klassen und bei der Definition der Botschaften, die durch das System fließen und dienen zur Validierung der Vollständigkeit und Korrektheit des statischen Modells. [BAL99, S.170] Für einige Klassen kann es sinnvoll sein, ihren Lebenszyklus durch einen Zustandsautomaten explizit zu machen.

Der Analytiker wechselt zwischen der Erstellung eines dynamischen Modells und eines statischen Modells hin und her. [BAL99, S.120] Statisches und dynamisches Modell sollen im Gleichgewicht stehen. Aber: „Die Konzentration auf das statische Modell vor dem dynamischen sorgt für eine größere Stabilität des Modells und schafft durch die Bildung von Klassen eine wesentliche Abstraktionsebene. Steht zu Beginn der Modellierung nur das dynamische Modell im Vordergrund, dann besteht meines Erachtens das Problem, dass der Analytiker in der großen Menge von – oft geänderten – Funktionen den Überblick verliert." [BAL99, S.121] Die Modelle werden schließlich aber parallel weiterentwickelt. Dabei muss stets auf die Konsistenz der beiden Modelle geachtet werden.

Das Modell des Fachkonzepts muss alle Informationen enthalten, um daraus einen Prototyp der möglichst vollständigen Benutzungsoberfläche abzuleiten. „Der Prototyp der Benutzungsoberfläche ist ein ablauffähiges Programm, das alle Attribute des OOA-Modells auf die Oberfläche abbildet." [BAL99, S.10] Der Prototyp kann nach festen Transformationsregeln aus dem OOA-Modell abgeleitet werden. [BAL99, S.9]
„Das OOA-Modell wird im Team von Systemanalytikern, Fachexperten und zukünftigen Benutzern oder den Benutzerrepräsentanten erstellt." [BAL99, S.11]

Zusammenfassung

Die objektorientierte Analyse-Methode von Heide Balzert will die Anforderungen an ein neues Softwaresystem ermitteln und in einem OOA-Modell beschreiben. Zur Modellierung statischer und dynamischer Aspekte konzentriert sich die Methode auf die Verwendung von UML. Die Methode geht insbesondere von Usecases aus und führt zu einem Klassenmodell. Ein Prototyp der Benutzeroberfläche dient zur Rückkopplung des OOA-Modells beim Auftraggeber.
Die Analyse-Methode sieht sich in einen evolutionären, objektorientierten Softwareentwicklungsprozess eingebettet.

Grundlagen	• Objekte der realen Welt • Beschreibungen des altes Systems (Formulare, Dateibeschreibungen, Listen, Benutzerhandbücher und Bildschirmmasken eines alten Systems) • Andere Dokumente
Tätigkeiten	Interviews, Erstellung von Modellen
Modelle etc.	OOA-Modell: • Statisches Modell o Klassendiagramm o Kurzbeschreibung der Klassen o Objektdiagramm o Attributspezifikation • Dynamisches Modell o Geschäftsprozesse (Geschäftsprozessdiagramme, Aktivitätsdiagramme) o Szenarios (Interaktionsdiagramme: Sequenzdiagramm, Kollaborationsdiagramm) o Zustandsautomaten (Zustandsdiagramm) • fachliche Beschreibung
Ergebnisse	• Pflichtenheft • OOA-Modell
Ziele	• Wünsche und Anforderungen des Auftraggebers ermitteln • konsistentes, vollständiges, eindeutiges und realisierbares Modell des Fachkonzepts erstellen • Verstehen und Beschreiben des Problems (Struktur und Semantik) • Stabiles, konsistentes OOA-Modell

Tab. 17: OOA mit UML nach Heide Balzert – Grundlagen und Ergebnisse

3.2 Unterschiede der Analyse-Methoden

Hier wird auf verschiedene Aspekte der betrachteten Analyse-Methoden eingegangen. Dazu werden alle Methoden für verschiedene Aspekte betrachtet. Dazu wird erst eine knappe Definition des Aspekts gegeben und dann für jede Methode einzeln betrachtet, wie sie diesen Aspekt abdeckt. Am Ende steht eine Übersichtstabelle über die Abdeckung der Aspekte für alle Methoden.

3.2.1 Anwendungsfälle

Am Ende von Standardsoftwareeinführungsprozessen und auch Softwareneuentwicklungsprozessen stehen Benutzer einer Software gegenüber, die sie in irgendeiner Form benutzen. Diese Schnittstelle zwischen Benutzern und Software wird aber nur bei einigen Analyse-Methoden in den Blick genommen. Als Überschrift könnte hier der Begriff Usecase stehen, der auch mit Anwendungsfall übersetzt werden kann. Andere Verwendungen dieses Begriffs setzen ihn aber mit Geschäftsprozessen gleich. Geschäftsprozesse werden wiederum mit Akteursübergreifenden Abläufen gleichgesetzt, die hier aber nicht gemeint sein sollen (zu dieser Problematik s.a. [BAL99, S.62]).

SA will einen breiten Gegenstandsbereich betrachten: „...consider the context of analysis to be whatever we propose to change plus enough surrounding procedure to buffer the changed area from activity on the outside.” [DEM79, S.60] Dabei werden bewusst auch Bereiche modelliert, die später nicht automatisiert werden sollen. [S.60] Es wird nicht die Sicht der Benutzer eingenommen, sondern die Sicht der Daten. Die Schnittstellen zwischen manuellen und automatisierten Prozessen werden durch Datenflüsse in den DFDs modelliert [S.60]. Eigentlich müssten die Schnittstellen aber in den Prozessen liegen, wenn Daten z.B. in das System eingegeben oder ausgedruckt werden. Die Betrachtung von Anwendungsfällen widerspricht auch eher der Philosophie von SA, dass nicht die Benutzersicht, sondern die Datensicht eingenommen wird.

Auch bei GPA steht die Schnittstelle zur Software nicht im Vordergrund, sondern „... die Unternehmensrealität in all ihren formellen Aspekten ...„ [STA99, Vorwort]. Es werden dabei auch Handlungen modelliert, die die Software nur zum Teil berühren. Der Wunsch von GPA ist aber, dass Abläufe in der Anwender-Organisation am Besten in ganzer Länge durch Software unterstützt werden. Anwendungsfälle werden nicht betrachtet.

OBA konzentriert sich zur Analyse auf die zu erstellende Software und deren Verhaltensweisen. [RUG92, S.49ff] Dabei werden in der Script Notation explizit Handlungen (actions) beschrieben, in denen ein Benutzer das System benutzt. Anwendungsfalle liegen somit den Zeilen der Script Notation vor.

Bei OOA nach Heide Balzert sind die Grenzen der Software im Usecase-Diagramm durch die Ränder des System-Rechtecks festgelegt. Usecases betreffen für Heide Balzert die „... ergebnisorientierten Arbeitsabläufe bei Benutzung der zu realisierenden Software ...“ [BAL99, S.63]. Aufgaben, die die Akteure ohne das System erledigen, sind nicht von Interesse und werden nicht modelliert. Andererseits ist aber „... zu diesem frühen Zeitpunkt ... noch nicht abzusehen, ob jeder use case ausschließlich durch Software realisiert wird oder auch organisatorische Schritte enthält, in denen der Bediener Entscheidungen treffen oder bestimmte Aktivitäten durchführen muss.“ [BAL99, S.63] Heide Balzerts Usecase-Begriff stimmt mit dem hier benutzten Begriff Anwendungsfall überein.

3.2.2 Softwareabläufe

Die Modellierung von Abläufen in der Software wird mit Hilfe von Kontrollstrukturen (Sequenz, Auswahl und Wiederholung) durchgeführt. Der Ablauf wird dann als Kontrollfluss bezeichnet. Abläufe, die später in Software umgesetzt werden sollen, zeichnen sich durch einen hohen Detailgrad aus.

SA will mit seinen Datenflussdiagrammen den Kontrollfluss nicht direkt in den Vordergrund stellen: „A data flow is not a representation of flow of control, the stream of consciousness of the computer or person that processes the data.“ [DEM79, S.54] Die nicht mehr zerlegbaren Prozesse der Datenflussdiagramme werden aber mittels Structured English genauer spezifiziert. Structured English bietet alle Kontrollstrukturen an und dient so zur Modellierung von Abläufen in der Software. [DEM79, S.179ff].

Zentrales Modellierungsmittel bei GPA sind EPKs. Diese bieten im Prinzip Sequenz, Auswahl und Wiederholung an und können so alle Arten von Kontrollflüssen modellieren. Der Fokus liegt hier aber nicht auf Abläufen in der Software, sondern auf den übergreifenden Geschäftsprozessen eines Unternehmens. Obwohl eine Modellierung von Softwareabläufen im Prinzip möglich ist (Es ist „... alles möglich, von einer hoch angesiedelten Meta-Ebene bis zu einer tayloristischen Ausdifferenzierung der einzelnen Tätigkeiten.“ [STA99, S.10]), ist die Modellierung von EPKs wie Staud sie betreibt zu abstrakt bzw. unscharf [S.10] (s.a. Akteursübergreifende Abläufe 3.2.4).

OBA arbeitet zur Vereinfachung unter der Annahme, dass alle Operationen sequentiell nacheinander ablaufen: „It has been our experience that over-constraining the order in which activities take place within a system is one of the principal causes of change requests in big systems.” [RUG92, S.61]. Rubin und Goldberg verweisen deswegen zur Darstellung von Kontrollflüssen auf andere Modelle, die sie selbst nicht näher behandeln. Der Schwerpunkt von OBA liegt nicht auf der Modellierung der Abläufe, sondern auf der Identifikation der Objekte.

OOA nach Heide Balzert bietet zur Modellierung von Kontrollflüssen das Sequenzdiagramm und das Activity-Diagramm. Das Sequenzdiagramm bietet alle Kontrollstrukturen an [BAL99, S.73]. Dabei stehen die Botschaften, die sich Objekte untereinander schicken, im Vordergrund. Das Activity-Diagramm wird von Heide Balzert zwar als Sonderfall des Zustandsdiagramms in die Beschreibung ihrer Konzepte aufgenommen [S.78ff], auf dessen Verwendung wird aber bei der Beschreibung der Methode nicht näher eingegangen.

3.2.3 Klassenmodell

Die Identifizierung von Klassen ist Voraussetzung für die objektorientierte Software-konstruktion.

Zur Zeit als SA entstand, gab es noch keine Objektorientierte Softwareentwicklung. SA benutzt die Begriffe Objekt und Klassen auch nicht. Stattdessen konzentriert es sich auf Daten.

GPA fordert bei der Modellierung von Geschäftsprozessen auch die Identifikation von Informationsobjekten. Informationsobjekte sind Daten, die ein „... Unternehmen selbst und seine informationelle Umwelt ganz oder in Ausschnitten modellieren.“ [STA99, S.50]

(Kundendaten, alte Angebote, Marktpreise etc.). Die Informationsobjekte genügen der Objektorientierten Sicht von Klassen nicht.

Bei OOA nach Heide Balzert werden zwei einfache Methoden zur Identifizierung von Klassen vorgeschlagen: Dokumentanalyse und Durchsuchen der Geschäftsprozessbeschreibungen. Weitere Hilfestellungen werden nicht gegeben: „Der Erfolg ... wird entscheidend durch die Sicherheit des Systemanalytikers im Erkennen der potentiellen Klassen bestimmt.“ [BAL99, S.143]

Für OBA ist die Identifizierung von Klassen dagegen zentrales Anliegen. Besonders für große Systeme kritisieren die Autoren Methoden, die die Suche nach Substantiven, Verben und Attributen fordern oder die Annahme machen, dass formale und korrekte Anforderungsspezifikationen existieren. [RUG92, S.48]

3.2.4 Akteursübergreifende Abläufe

Bei akteursübergreifende Abläufen (gemeint sind hier menschliche Akteure oder Organisationseinheiten) sind im Gegensatz zu Softwareabläufen (Kap. 3.2.2) mehrere Akteure involviert. Sie modellieren auch Abläufe, die in Teilen unabhängig von der Benutzung eines Informationssystems sind. Akteursübergreifende Abläufe sind das zentrale Mittel um die Ablauforganisation eines Unternehmens darzustellen. Dabei können Kontrollstrukturen verwendet werden.

Die Datenflussdiagramme bei SA modellieren akteursübergreifende Abläufe. Den Prozessen, zwischen denen die Datenflüsse fließen, können bei Erstellung des ersten physischen DFD bestimmte Personen, Organisationseinheiten und Orte zugeordnet werden. Diese sollen aber beim Übergang zum logischen DFD eliminiert werden: „Physical attributes are allowed in the initial subphase of analysis and then systematically eliminated later.“ [DEM79, S.231]. DFDs bieten auch keine Kontrollstrukturen.

GPA berücksichtigt bei der Modellierung von Geschäftsprozessen mehrere „Aufgabenträger“ in verschiedenen „organisatorischen Einheiten“ [STA99, S.6]. Die Überwindung der einzelplatzorientierten Sicht ist hier sogar zentrales Anliegen. Die Prozesssicht dient insbesondere zur Prozessoptimierung. Zur Modellierung werden EPKs verwendet, die auch alle Kontrollstrukturen zur Verfügung stellen. Die Modellierung der Abläufe ist, wegen der hohen Abstraktion der Modellierung, unscharf: „Was für das Business Process Reengineering im allgemeinen gilt ... gilt auch für die Geschäftsprozessanalyse im besonderen: die Methoden sind nicht exakt, auch nicht die Identifikation und Abgrenzung von Geschäftsprozessen.“ [STA99, S.10]

Die Script Notation von OBA bietet im Prinzip die Möglichkeit Interaktionen zwischen mehreren Akteuren darzustellen, wenn man sowohl Participants als auch Initiatoren als Akteure begreift und in einer Tabelle auch mehrere Akteure vorkommen dürfen. Dies wird z.B. von Krabbel, Wetzel und Ratuski für die Analyse von übergreifenden Aufgaben vorgeschlagen [KWR96]. Rubin und Goldberg verwenden die Script Notation aber anders: Zentral für sie ist die Interaktion zwischen Benutzer und System, um die Verhaltensweisen von Objekten festzustellen. Die Kontrollstrukturen Wiederholung und Auswahl sind in der Script Notation nicht vorgesehen.

OOA nach Heide Balzert modelliert akteursübergreifende Abläufe nicht. Akteure stehen immer dem System gegenüber, das sie in irgendeiner Form benutzen (Anwendungsfall). Aus dem Usecase-Diagramm kann nur ersehen werden, ob verschiedene Akteure denselben Anwendungsfall verwenden. Interaktionen, die nicht das neu zu entwickelnde Softwaresystem betreffen, sind nicht interessant. In den Interaktionsdiagrammen ist immer nur ein Akteur involviert. Nur das Sequenzdiagramm bietet alle Kontrollstrukturen.

3.2.5 Organisationsbrüche

Unter Organisationsbrüchen soll hier ein Wechsel in der Aufbauorganisation eines Unternehmens von einer Organisationseinheit zu einer anderen oder ein Ortswechsel gemeint sein, der bei Erledigung einer übergreifenden Aufgabe entsteht. Organisationsbrüche hängen deswegen eng zusammen mit der Modellierung akteursübergreifender Abläufe.

SA erlaubt in den physischen Datenflussdiagrammen Personen, Organisationseinheiten und Orte aufzunehmen. Dadurch können Organisationsbrüche bei der Verarbeitung von Daten in verschiedenen Prozessen aufgedeckt werden. Für DeMarco steht die Aufdeckung von Organisationsbrüchen aber nicht im Vordergrund, sondern die Darstellung des Bezugs zu den Datenflüssen in der Anwender-Organisation.

GPA expliziert die Aufdeckung von Organisationsbrüchen. Durch die Modellierung der Organisationseinheiten in Geschäftsprozessen wird der Wechsel zwischen Organisationseinheiten offensichtlich.

OBA und OOA nach Heide Balzert bieten keine Mittel zur Aufdeckung von Organisationsbrüchen an. Die mit UML mögliche Darstellung von Organisationsbereichen mittels Swimlanes in Aktivitätsdiagrammen wird bei Heide Balzert nicht benutzt. [UML99, S.155]

3.2.6 Übersicht

Eine Übersicht über die betrachteten Aspekte für alle Analyse-Methoden ist in Tab. 18 abgebildet.

Methode / Aspekt	**SA**	**GPA**	**OBA**	**OOA (Heide Balzert)**
Identifizierung von Anwendungs-fällen	nein (verschwinden in DFDs)	nein	ja (Script Notation)	ja (Usecases)
Modellierung von Software-abläufen	möglich (Structured English)	möglich, wird aber nicht dafür verwendet	nur sequentiell (Verweis auf andere Modelle)	möglich, Fokus liegt aber auf Objekt-Kommunikation (Sequenzdiagramm)
Konstruktion eines Klassen-modells	nein	nein	ja	ja (aber nur wenig konkrete Anleitung zur Identifizierung von Klassen)
Modellierung akteursüber-greifender Abläufe	möglich (physische DFDs), wird aber nicht dafür verwendet	ja, aber mit inhärenter Unschärfe	nein	nein
Identifizierung von Organisa-tionsbrüchen	möglich (physische DFDs), wird aber nicht dafür verwendet	ja	nein	nein

Tab. 18: Aspekte der Analyse-Methoden im Überblick

3.3 Erweiterungen der Analyse-Methoden

An dieser Stelle sollen drei Erweiterungen der Analyse-Methoden beschrieben werden. Alle Erweiterungen verwischen die Grenze zwischen Softwarekonstruktion und Reorganisation.
Software-Spezifikation mit ereignisgesteuerten Prozessketten kann als Erweiterung zu GPA gesehen werden. Objektorientierte Analysetechniken für übergreifende Aufgaben kann als Erweiterung zu OBA gesehen werden und Business Modeling mit UML als Erweiterung zu OOA nach Heide Balzert.

3.3.1 Software-Spezifikation mit ereignisgesteuerten Prozessketten

Constantin Szallies schlägt die Verwendung von EPKs zur Spezifizierung von Software vor [SZA98]. Er hebt dazu u.a. die „Kommunikation mit der Fachseite“ und die „Erstellung des Softwaredesigns“ hervor.
Die bei Erstellung einer Spezifikation wichtige Kommunikation mit der Fachseite kann durch EPKs unterstützt werden, weil diese oft auf der Fachseite bereits bekannt ist. Deswegen ist sie dort leichter zu verstehen. [S.40]
Bei Erstellung eines Softwaredesigns hält Szallies eine starke Formalisierung der Spezifikation für besser als einen nicht formalisierten Text. EPKs sieht er als Kompromiss zwischen diesen beiden Extremen. Die fehlende formale Grundlage von EPKs wurde zwar versucht durch deren Abbildung auf Petri-Netze auszugleichen, dies hat sich in der Praxis aber relativiert. [S.40]
Insgesamt werden nach Szallies aber neben EPKs noch weitere Modelle zur Spezifikation benötigt, wie bspw. Datenmodell, Zustandsautomaten oder Benutzeroberflächen-Prototypen. [S.41]

3.3.2 Objektorientierte Analysetechniken für übergreifende Aufgaben

Mit der von Krabbel, Wetzel und Ratuski für die Analyse von übergreifenden Aufgaben vorgeschlagene Erweiterung von OBA [KWR96] können auch akteursübergreifende Abläufe betrachtet werden (Tab. 19).

Initiator	Action	Participant	Service
Studierender	Antrag auf Zulassung einreichen	Sachbearbeiterin	Antragsannahme
Sachbearbeiterin	Zulassung überprüfen	Prüfungsordnung	Zulassungsvoraussetzungen
Sachbearbeiterin	Prüfungsakte anlegen (bei Erstanmeldung)	Prüfungsakte	
Sachbearbeiterin	Identität des Studierenden überprüfen (bei Erstanmeldung)	Ausweis	Name, Foto
Sachbearbeiterin	Zulassungsvoraussetzungen kontrollieren	Prüfungsakte	Vollständigkeit überprüfen
Sachbearbeiterin	Zulassungsvoraussetzungen kontrollieren	Studienplan	Genehmigung überprüfen
Sachbearbeiterin	Zulassungsvoraussetzungen kontrollieren	Fristen	Frist prüfen
Prüfungsausschuss-vorsitzender	Unterschreiben der Zulassung	Zulassungsgenehmigung	Unterschreiben
Sachbearbeiterin	Versendung der Zulassungsgenehmigung	Post	Versenden

Tab. 19: Erweiterung der Script Notation von OBA nach [KWR96]

3.3.3 Business Modeling mit UML

Heide Balzert verwendet in ihrer Analyse-Methode nur Bruchteile von UML. Insbesondere bietet UML auch ein Profile zum Business Modeling. Exemplarisch soll hier die Modellierung von akteursübergreifenden Abläufen und Organisationsbereichen in Aktivitätsdiagrammen betrachtet werden.
Aktivitäts-Diagramme in UML können auch zur Modellierung akteursübergreifender Abläufe benutzt werden. Durch die Einführung von Swimlanes können außerdem Organisationseinheiten abgebildet werden (Abb. 22). Die Aktivitäten des Aktivitätsdiagramms können werden dann jeweils einer Organisationseinheit zugeordnet. Dadurch können z.B. Organisationsbrüche aufgedeckt werden.

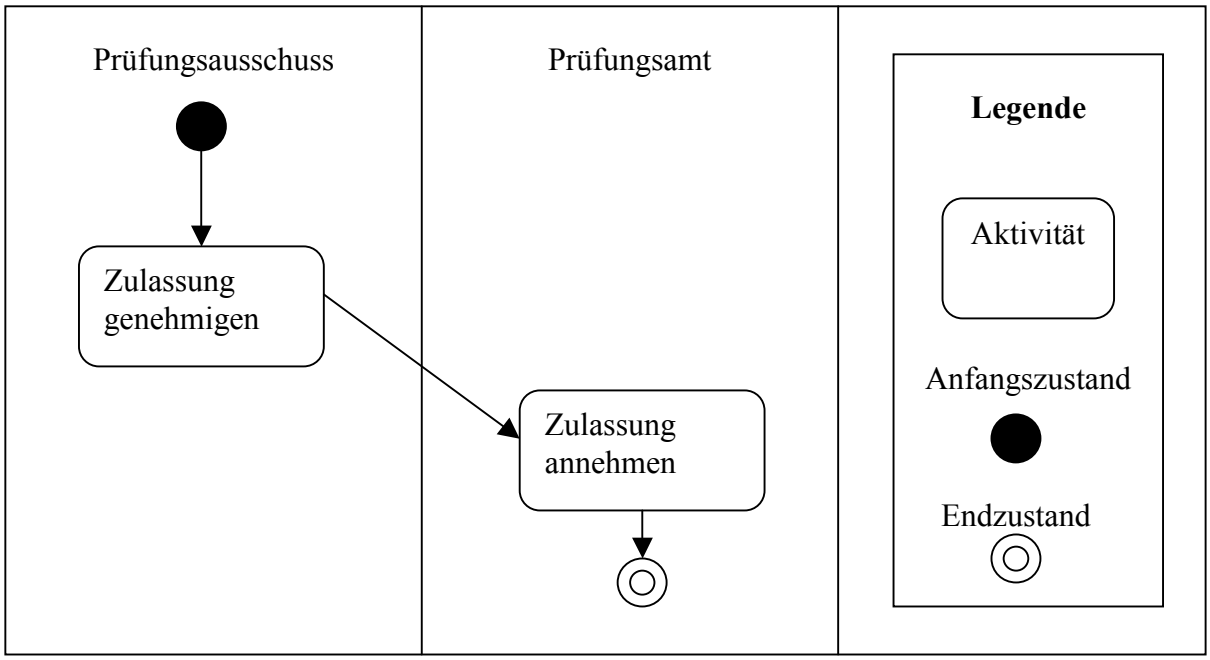
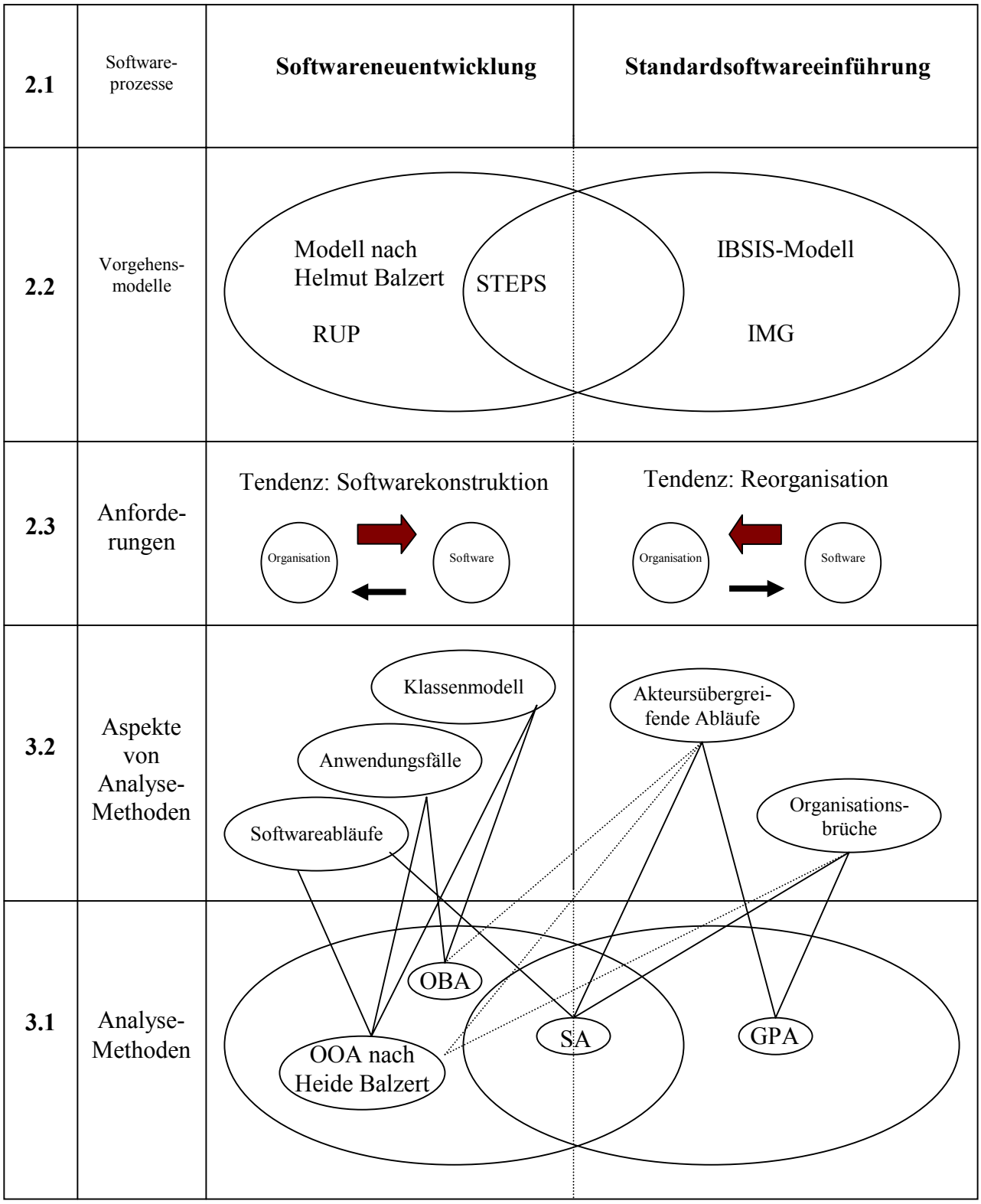


Abb. 22: Aktivitätsdiagramm mit je einer Swimlane für Prüfungsausschuss und Prüfungsamt

4 Klassifizierung der Analyse-Methoden

In diesem Kapitel werden die beiden Sichten auf Analyse aus den Kapiteln 2 und 3 zusammengeführt. In Kapitel 2 ergaben sich für die betrachteten Vorgehensmodelle zwei Tendenzen: Vorgehensmodelle der Standardsoftwareeinführung fordern von Analyse-Methoden, dass diese eine Reorganisation ermöglichen, während Vorgehensmodelle der Softwareneuentwicklung Analyse-Ergebnisse erwarten, die die Konstruktion von Software ermöglichen. In Kapitel 3 ergaben sich für die betrachteten Analyse-Methoden eine Reihe von Aspekten, in denen die Methoden unterschiedliche Schwerpunkte setzen. In diesem Kapitel wird gezeigt, dass die für Analyse-Methoden festgestellten Aspekte zum Großteil auch Differenzierungen der Tendenzen der Softwareprozesse sind. Über die Aspekte lassen sich deswegen die Analyse-Methoden den Softwareprozessen zuordnen (Tab. 22).

Im Unterkapitel 4.1 wird betrachtet, welche Aspekte der Analyse-Methoden sich einer der beiden Tendenzen von Vorgehensmodellen zuordnen lassen. Dadurch werden die Analyse-Methoden in das Spannungsfeld der betrachteten Softwareprozesse eingeordnet und abhängig von Softwareprozessen klassifiziert. Dabei wird auch betrachtet, wie sich die in Kapitel 3.3 vorgestellten Erweiterungen der Analyse-Methoden auf die Klassifizierung auswirken.
Im dann folgenden Unterkapitel 4.2 wird ein auf der Basis der Klassifizierung erstellter Methodenkatalog präsentiert, der dem Praktiker die Auswahl einer Analyse-Methode in Abhängigkeit vom jeweils angestrebten Softwareprozess ermöglicht.



Tab. 20: Zuordnung von Analyse-Methoden zu Softwareprozessen

4.1 Analyse-Methoden im Spannungsfeld

Im 3. Kapitel wurden mehrere Aspekte festgestellt, in denen sich die Analyse-Methoden unterscheiden:

- Anwendungsfälle
- Abläufe in der Software
- Klassenmodell
- akteursübergreifende Abläufe
- Organisationsbrüche

Alle Aspekte entsprechen den Forderungen der Vorgehensmodelle eines der beiden Softwareprozesse Standardsoftwareeinführung und Softwareneuentwicklung. Dies soll im Folgenden gezeigt werden. Die Aspekte entsprechen einerseits den Anforderungen der Vorgehensmodelle und konkretisieren die festgestellten Tendenzen Softwarekonstruktion und Reorganisation. Andererseits sind die Aspekte Eigenschaften, die Analyse-Methoden haben können oder nicht. Dadurch können die Analyse-Methoden, die eher Aspekte haben, die sich für Reorganisation eignen, Standardsoftwareprozessen zugeordnet werden. Analyse-Methoden, die eher Aspekte erfüllen, die Softwarekonstruktion vorbereiten, lassen sich den Softwareneuentwicklungsprozessen zuordnen.
Im Folgenden wird für jeden Aspekt zuerst betrachtet, wie er durch die Analyse-Methoden variiert wird. Dann wird gezeigt, ob er eher zu Softwareprozessen der Standardsoftware-einführung oder Softwareneuentwicklung gehört.

Anwendungsfälle

Der Gegenstandsbereich, den die Analyse-Methoden betrachten, reicht von „... ergebnis-orientierten Arbeitsabläufe[n] bei Benutzung der zu realisierenden Software ...“ [BAL99, S.63] über den Bereich der Veränderung und seiner „prozeduralen“ Umgebung (“...whatever we propose to change plus enough surrounding procedure to buffer the changed area from activity on the outside.” [DEM79, S.60]) bis zur „... Unternehmensrealität in all ihren formellen Aspekten ...“ [STA99, Vorwort].
GPA und SA betrachten Anwendungsfälle, also die Schnittstellen zwischen Benutzer und Software nicht explizit, während diese bei OOA nach Heide Balzert mit Usecases explizit angesprochen werden. Vor allem bei GPA ist der vermeintlich größere Gegenstandsbereich, der die gesamte „Unternehmensrealität“ betrachten will, also gar nicht größer, sondern abstrakter bzw. nimmt andere Dinge in den Blick.
Die Betrachtung der Vorgehensmodelle unter dem Aspekt der Anwendungsfälle erhellt ein begriffliches Problem: Alle Vorgehensmodelle bis auf STEPS sprechen von Geschäfts-prozessen (bei STEPS taucht der Begriff Arbeitsprozesse auf). Die Softwareneuent-wicklungsmodelle meinen dabei aber Anwendungsfälle, während die Standardsoftware-einführungsmodelle akteursübergreifende Abläufe meinen. Geschäftsprozess ist nicht gleich Geschäftsprozess (s.a. Tab. 21). Die Identifizierung von Anwendungsfällen entspricht also den Anforderungen von Softwareneuentwicklungsprozessen, nicht aber denen von Standard-softwareeinführungsprozessen. Anwendungsfälle unterstützen die Tendenz Software-konstruktion insofern, als dass aus ihnen direkt ein Klassenmodell abgeleitet werden soll.

IBSIS [BAR96]: Geschäftsprozessanalyse	Balzert [BALZ98]: Unternehmensmodellierung	RUP [KRU98]: Usecase-Analyse
• Dokumentation der Analyseergebnisse der Geschäftsprozesse • Entwickeln eines Sollmodells der Geschäftsprozesse und • Abgleich des Sollmodells der Geschäftsprozesse mit dem Modell des IBSIS • wiederverwendbare Modellierung für leichte Veränderbarkeit • Abbildung von Funktionen, Organisationseinheiten, Daten, Zeit und Kosten • leichte Beherrschbarkeit (Schulungszeit von ein bis zwei Stunden) • funktionsübergreifende Betrachtung der Aktivitäten. Aktivitäten sollen nicht aus einem Prozess gelöst und einzeln optimiert werden.	• Unternehmensmodellierung für Software-Ingenieur relevant • Software entscheidend bei der Realisierung eines Unternehmensmodells • Beschreibung und Optimierung der Geschäftsprozesse • Unterscheidung in Teilprozesse, die durch Software oder durch Mitarbeiter erledigt werden • Objekttypen: Interface, Control, Entity • Übergang vom Unternehmensmodell zu OOA-Modellen	• Struktur und Dynamik einer Organisation verstehen • sicherstellen, dass Kunden, Endanwender und Entwickler das gleiche Bild von einer Organisation haben • Systemanforderungen ableiten, die benötigt werden, um eine Organisation zu unterstützen • Organisationseinheiten grob skizzieren und abgrenzen • die existierenden Geschäftsprozesse und Akteure feststellen. • Spezifikation um Workflows, Worker, Entitäten, Verantwortlichkeiten, Aktivitäten und Attribute ergänzen

Tab. 21: Geschäftsprozesse bei verschiedenen Vorgehensmodellen

Softwareabläufe

Die Modellierung von Abläufen in der Software wird von den Methoden in verschiedenen Graden ermöglicht. Während OBA sich auf sequentielle Abläufe beschränkt und auf andere Methoden verweist, bieten die anderen Methoden zumindest eine Form der Modellierung von Kontrollflüssen an, die alle Kontrollstrukturen (Sequenz, Wiederholung, Bedingung) zur Verfügung stellen. Diese reichen von textueller Darstellung (Structured English bei SA), über EPKs (GPA) zu Sequenzdiagrammen, in denen die Kommunikation zwischen Objekten im Vordergrund steht (OOA nach Heide Balzert). Bei GPA liegt der Fokus der Abläufe aber nicht auf der Umsetzung in Software, sondern betrachtet die Ablauforganisation eines Unternehmens. Dies liegt auch an der Unschärfe der Methode.
Softwareabläufe sind Teil einer dynamischen Software-Architektur. Vor allem zu Zeiten von Structured Design waren sie zur Spezifikation von Software zentral. In Zeiten von Objektorientierung wird die zentrale Orientierung an Abläufen aber als sekundär oder sogar nachteilig betrachtet, da dabei im Prinzip beliebige Ausführungsreihenfolgen statisch festgelegt werden. Das Festschreiben von Abläufen steht so einer veränderlichen Softwarearchitektur entgegen. Dennoch werden Abläufe auch in objektorientierten Programmen gebraucht, wo Aufrufreihenfolgen von Objekten oder auch Algorithmen innerhalb eines Objektes dargestellt werden müssen. Softwareabläufe sind deswegen ein Aspekt der Softwarekonstruktion und entsprechen den Anforderungen von Softwareneuentwicklungsprozessen.

Klassenmodell

Klassenmodelle werden von SA und GPA nicht unterstützt. Die bei GPA zu identifizierenden Informationsobjekte sind keine Objekte im Sinne der Objektorientierung.
Bei OBA und OOA von Heide Balzert ist die Erstellung eines Klassenmodells dagegen zentral. Während OBA ihr Hauptaugenmerk auf die Identifizierung von Objekten und Klassen legt, liefert OOA von Heide Balzert hierzu wenig methodische Hilfestellung.
Die Identifizierung von Klassen und Objekten und die Erstellung einer Software-Architektur in Form eines Klassenmodells ist für die objektorientierte Softwarekonstruktion zentral. Es gibt zwar auch Ansätze, die versuchen ein objektorientiertes Unternehmensmodell aus Business Objects zu erstellen (z.B. [KOH96], [FES96]); da sich ein Klassenmodell in Bezug auf Abläufe aber nicht festlegt, ist es zumindest für eine prozessorientierte Reorganisation des Unternehmens nicht geeignet. Die Erstellung eines Klassenmodells scheint also eher für Softwarekonstruktion als für Reorganisation geeignet.
Die Erstellung eines Klassenmodells wird hier deswegen indirekt über die Tendenzen der Softwareprozesse den Softwareneuentwicklungsprozessen zugeordnet.

Akteursübergreifende Abläufe

Nicht alle Analyse-Methoden unterstützen die Modellierung von akteursübergreifenden Abläufen. Für GPA stehen diese mit der Prozesssicht zentral im Vordergrund und sind die Basis für spätere Prozessoptimierung. SA erlaubt zwar die Modellierung von „physischen Charakteristika“ (Personen, Organisationen, Orten etc.) in den ersten Entwürfen der Datenflussdiagramme, will diese aber später möglichst vermeiden [DEM79, S.231]. OOA von Heide Balzert und OBA betrachten keine akteursübergreifenden Abläufe.
Akteursübergreifende Abläufe dienen der Darstellung der Ablauforganisation eines Unternehmens und sind zentral bei deren Reorganisation (Prozessoptimierung). Für Softwarekonstruktion sind sie dagegen zu unscharf. Der Aspekt der akteursübergreifenden Abläufe wird hier deswegen indirekt über die Tendenzen den Standardsoftwareeinführungsprozessen zugeordnet.

Organisationsbrüche

Die Aufdeckung von Organisationsbrüchen wird von OBA und OOA nach Heide Balzert nicht verfolgt. Bei GPA wird sie explizit verfolgt. Mit SA wäre die Aufdeckung anhand des ersten physischen DFDs zwar möglich, wird in der Methode aber nicht explizit verfolgt.
Die Aufdeckung von Organisationsbrüchen kann potentiell der Reorganisation eines Unternehmens dienen. Es kann z.B. versucht werden, häufige Wechsel zwischen Organisationseinheiten der Aufbauorganisation zu reduzieren, oder aber auch eine Auslagerung von Aktivitäten betrachtet werden (z.B. Outsourcing). Für die Softwarekonstruktion ist das Aufdecken von Organisationsbrüchen dagegen sekundär. Die Identifizierung von Organisationsbrüchen kann also indirekt über die Tendenzen von Softwareprozessen den Standardsoftwareeinführungsprozessen zugeordnet werden.

Übersicht

Die obige Zuordnung der Aspekte zu einem der beiden Softwareprozesse ist als Übersicht in Tab. 22 dargestellt. Über die Aspekte lassen sich nun auch die Analyse-Methoden den Vorgehensmodellen bzw. Softwareprozessen zuordnen (Abb. 23). Es ergibt sich eine relativ klare Zuordnung von OBA und OOA nach Heide Balzert auf Seite der Softwareneuentwicklungsprozesse, GPA auf Seite der Standardsoftwareeinführung und SA dazwischen. Die Erweiterungen der Analyse-Methoden (Kap. 3.3) zeigen aber, dass die Analyse-

Methoden des einen Softwareprozesses auch auf Aspekte des anderen erweiterbar sind (gestrichelte Zuordnung der Aspekte in Abb. 23).

Softwareneuentwicklung	Standardsoftwareeinführung
• Identifizierung von Anwendungsfällen • Modellierung von Softwareabläufen • Konstruktion eines Klassenmodells	• Modellierung akteursübergreifender Abläufe • Identifizierung von Organisationsbrüchen

Tab. 22: Zuordnung der Aspekte von Analyse-Methoden zu Softwareprozessen

4.2 Methodenkatalog

Diese Arbeit soll für den Praktiker folgende Frage beantworten:

> „Welche der vielen existierenden Analyse-Methoden soll ich im Verlauf meiner Softwareentwicklung bzw. Standardsoftwareeinführung benutzen?“

Der hier präsentierte Methodenkatalog stellt dem Praktiker Orientierungswissen zur Verfügung, das ihm bei der Beantwortung dieser Frage hilft. Der Katalog nimmt also nicht nur eine Art von Softwareprozess in den Blick, sondern betrachtet und klassifiziert Analyse-Methoden übergreifend für Standardsoftwareeinführung und Softwareneuentwicklung. Ein solcher Methodenkatalog bereichert sowohl Softwaretechnik als auch Wirtschaftsinformatik (gefordert wird dies z.B. von Rolf [ROL98] und Hofmann [HOF00, S.232 u. 236]).

Der Methodenkatalog steht unter der Prämisse, dass Standardsoftwareeinführungsprozesse eher Analyse-Methoden brauchen, die Reorganisation vorbereiten, während Softwareneuentwicklungsprozesse Analyse-Ergebnisse zur Konstruktion von Software benötigen. (s.a. Kapitel 2.3)

Eine Übersicht, welche Analyse-Methode für welchen Softwareprozess geeignet ist, zeigt Abb. 23. Geschäftsprozessanalyse (GPA) deckt nur Aspekte der Reorganisation ab, ist also für Standardsoftwareeinführung geeignet. Object Behavior Analysis (OBA) und Objektorientierte Analyse (OOA) nach Heide Balzert decken nur Aspekte der Softwarekonstruktion ab und eignen sich deswegen für Softwareneuentwicklungsprozesse. Structured Analysis (SA) unterstützt Aspekte beider Softwareprozesse, eignet sich für objektorientierte Softwareentwicklungsprozesse aber nicht, da sie kein Klassenmodell konstruiert.

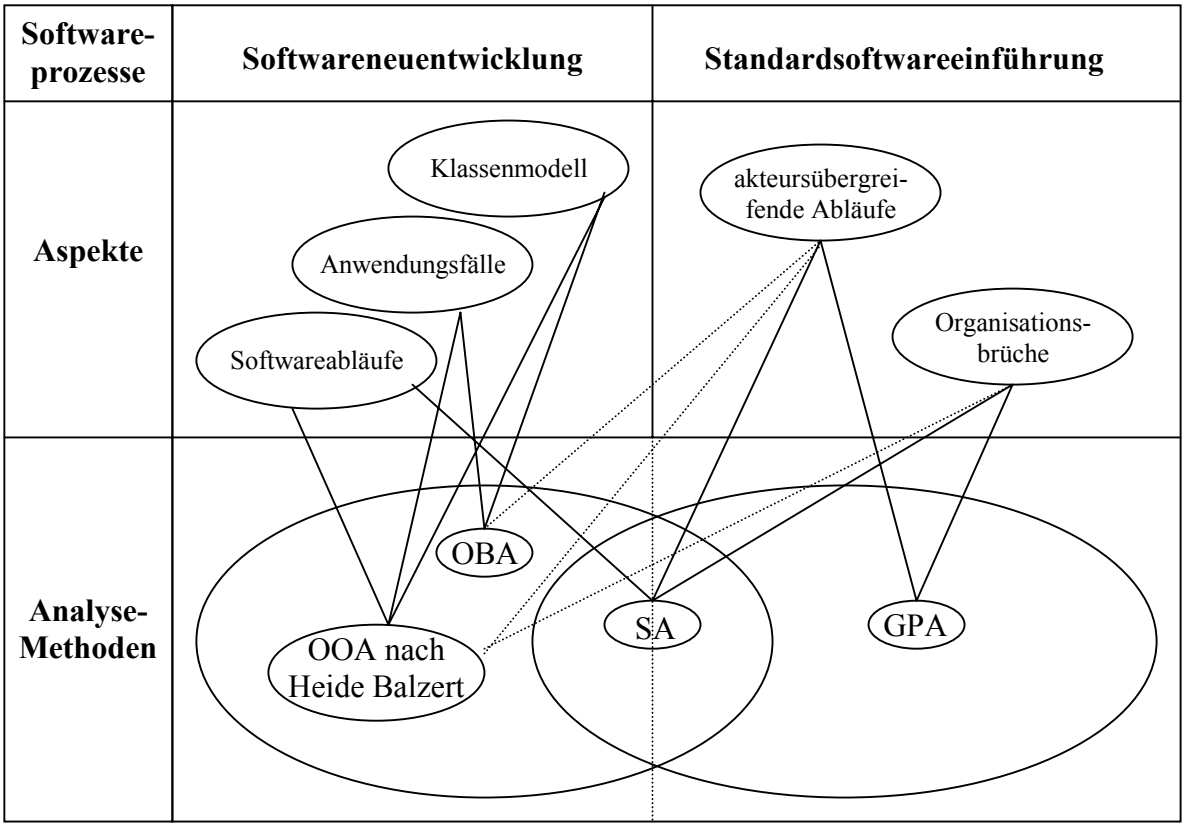


Abb. 23: Aspekte der Analyse-Methoden (Erweiterungen der Methoden: gestrichelte Linien)

Zu den Analyse-Methoden gibt es außerdem Erweiterungen (Kapitel 3.3), die in Abb. 23 durch die gestrichelt zugeordneten Aspekte dargestellt werden. Sie verschieben das Bild: Die beiden objektorientierten Analyse-Methoden erweitern ihren Horizont auf Aspekte, die Standardsoftwareeinführungsprozesse fordern. Mit der Erweiterung von OBA wird die Modellierung akteursübergreifender Abläufe möglich, mit der Erweiterung von OOA nach Heide Balzert werden akteursübergreifende Abläufe modellierbar und Organisationsbrüche offengelegt.

Wenn ein in der Praxis angestrebter Softwareprozess von den hier festgestellten Tendenzen von Softwareprozessen abweicht, also z.B. eine Softwareneuentwicklung durchgeführt werden soll, bei der gleichzeitig Reorganisation besonders berücksichtigt wird, muss die starre Zuordnung von Analyse-Methoden zu Softwareprozessen verlassen werden. Bei einer so erweiterten Fragestellung ermöglichen sowohl die festgestellten Tendenzen Reorganisation und Softwarekonstruktion als auch die festgestellten Aspekte eine differenzierte Auswahl (s.a. Abb. 23). Sollen beispielsweise akteursübergreifende Abläufe bei Softwareneuentwicklung zusätzlich in den Blick genommen werden, könnte OBA und dessen Erweiterung zur Modellierung akteursübergreifender Abläufe verwendet werden. Ebenso möglich, aber aufwändiger, wäre natürlich auch die Verwendung von OBA oder OOA nach Heide Balzert und zusätzlich die Durchführung von GPA.

Im Folgenden wird auf die hier betrachteten Analyse-Methoden und ihre Erweiterungen im Sinne eines prozessübergreifenden Methodenkatalogs eingegangen. Es wird jeweils beschrieben, für welche Softwareprozesse sich die Methode eignet, welche Vor- und

Nachteile sie bei Verwendung für Standardsoftwareeinführung und Softwareneuentwicklung bringt und was bei einem Wechsel des Softwareprozesses beachtet werden sollte.

Structured Analysis (SA)

SA unterstützt vor allem ältere Softwareentwicklungsprozesse, die strukturiertes Design betreiben (s.a. Kap. 3.1.1). Für objekt-orientierte Software-Neuentwicklung fehlt der Methode der Fokus auf Objekte und Klassen. Bei Neuentwicklung ist SA vor allem interessant, wenn alte Softwaresysteme mit Hilfe von SA erstellt wurden und auf die dabei erstellten Dokumente zugegriffen werden kann (Datenflussdiagramme, Data-Dictionaries etc.). Die Data-Dictionaries können hier zum Finden von ersten Klassenkandidaten helfen. Um alle Beziehungen zwischen Klassen aufzudecken, bedarf es aber einer zusätzlichen Methode. Auch die in Textform mit Structured English beschriebenen Softwareabläufe können zumindest für kleinere Algorithmen wiederverwendet werden. Die Entscheidungstabellen und –bäume von SA (Decision Trees, Decision Tables) können bei komplexen Entscheidungsproblemen auch in modernen Softwareentwicklungsprozessen eine sinnvolle Hilfestellung sein.
Unterstützung können alte Datenflussdiagramme auch bei Reorganisation im Zuge von Standardsoftwareeinführung bringen: Die Datenflussdiagramme modellieren akteursübergreifende Datenflüsse, die z.B. bei der Erstellung von EPKs verwendet werden können. In den ersten physischen DFDs werden dabei zum Teil auch Organisationseinheiten mitmodelliert, die in Geschäftsprozesse übernommen werden können. Inwieweit die Erstellung neuer DFDs in modernen Standardsoftwareeinführungsprozessen Sinn macht, bleibt zu prüfen.

Object Behavior Analysis (OBA)

Die objektorientierte Analyse-Methode OBA unterstützt insbesondere Softwareneuentwicklungsprozesse bei der Identifizierung von Klassen und Objekten (s.a. Kap. 3.1.2). OBA arbeitet viel mit Tabellen und benutzt kaum grafische Darstellungen. Anwendungsfälle werden nicht explizit hervorgehoben wie in den Usecase-Diagrammen bei OOA nach Heide Balzert, sondern sind in der Script Notation enthalten. Die Methode will nicht alle Probleme der Analyse lösen, sondern verweist z.B. zur Modellierung von Kontrollflüssen und Zuständen auf andere Methoden. OBA führt schließlich zu einem logischen Klassenmodell, das für OOD verwendet werden kann.
Die Methode liefert kaum Analyse-Ergebnisse, auf die eine Reorganisation bei Standardsoftwareeinführungsprozessen aufbauen kann. Die Erweiterung der Script Notation (Kap. 3.3.2) lässt aber eine Modellierung akteursübergreifender Abläufe zu, auf der z.B. eine Geschäftsprozessmodellierung mit EPKs aufsetzen kann.
Soll eine mit OBA durchgeführte Analyse für eine Standardsoftwareeinführung verwendet werden, ist eine Erweiterung der Script Notation auf akteursübergreifende Abläufe nachträglich vermutlich sehr aufwändig. Wahrscheinlich ist es besser, neue Modelle zu erstellen (z.B. EPKs) und die Script Notation zur Überprüfung der Vollständigkeit (bei EPKs: Funktionen und Informationsobjekte) hinzuzuziehen.

Geschäftsprozessanalyse (GPA)

Die Analyse-Ergebnisse von GPA lassen sich insbesondere zur Reorganisation bei Standardsoftwareeinführungsprozessen verwenden (s.a. Kap. 3.1.3). Durch Prozessmodellierung mit EPKs werden akteursübergreifende Abläufe mit zugehörigen Organisationseinheiten und Informationsobjekten modelliert.

Für Softwareneuentwicklungsprozesse liefert die Methode dagegen wenig Analyse-Ergebnisse. Insbesondere wird kein Klassenmodell erstellt und für Softwareabläufe sind die EPKs im Normalfall zu abstrakt und unscharf. Soll auf der Basis von EPKs dennoch eine Spezifikation von Software erfolgen, kann die in Kap. 3.3.1 betrachtete Methode hier Anhaltspunkte liefern.
Wird nach oder während der GPA die Entscheidung getroffen, doch Softwareneuentwicklung zu betreiben, können die Analyse-Ergebnisse zum Teil weiterverwendet werden. Die Informationsobjekte können erste Kandidaten für Klassen und Objekte sein. Auch hier kann die Erweiterung in Kap. 3.3.1 Anhaltspunkte liefern. Generell müssen die unscharfen Abläufe in EPKs verfeinert werden. Dazu kann eine Umwandlung in Aktivitäts-Diagramme in Betracht gezogen werden, die bei Verwendung von Swimlanes auch die Organisationseinheiten enthalten können. Für einen schnelleren Übergang zu OOA nach Heide Balzert kann auch versucht werden, gleich Sequenzdiagramme zu erstellen. Eine Umwandlung in die Script Notation von OBA kann besonders beim Auffinden von Klassen und Objekten helfen.

Objektorientierte Analyse (OOA) nach Heide Balzert

OOA nach Heide Balzert deckt die Analyse für moderne objektorientierte Softwareneuentwicklungsprozesse in großer Breite ab (s.a. Kap. 3.1.4). Ausgehend von der Identifikation von Anwendungsfällen wird ein logisches Klassenmodell erstellt, das Grundlage für objektorientiertes Design ist. Die Methode verwendet zahlreiche Diagramme der UML. Nachteil ist, dass die Methode zur Identifizierung von Klassen und Objekten wenig Hilfestellung gibt und vor allem auf die Erfahrung von Analytikern baut. Sollten Probleme beim Identifizieren von Klassen und Objekten auftreten, kann die Methode z.B. durch OBA ergänzt werden.
Die Methode liefert kaum Analyse-Ergebnisse, auf die eine Reorganisation bei Standardsoftwareeinführungsprozessen aufbauen kann. Identifizierte Anwendungsfälle und Sequenzdiagramme sind nicht akteursübergreifend und betreffen nur Schnittstellen zum Softwaresystem. Organisationseinheiten werden nicht betrachtet. Zur Modellierung von akteursübergreifenden Abläufen mit Organisationseinheiten kann die Erweiterung des Aktivitäts-Diagramms durch Swimlanes benutzt werden (Kap. 3.3.3). Hier entsteht aber Mehraufwand, da Heide Balzert Aktivitäts-Diagramme in der Analyse nur am Rande benutzt.

Ergibt sich nach oder während der Analyse, dass die Softwareneuentwicklung zugunsten einer Standardsoftwareeinführung gestoppt wird, können verschiedene Analyse-Ergebnisse weiterverwendet werden. Die Anwendungsfälle können als Grundlage für eine akteursübergreifende Prozessmodellierung herangezogen werden. Wird diese mit GPA durchgeführt, ist ein Anwendungsfall eine potentielle Funktion einer Ereignisprozesskette. Für die Identifizierung von Informationsobjekten der EPK können die mit OOA identifizierten Klassen und Objekte herangezogen werden. Wurden Aktivitäts-Diagramme mit Swimlanes erstellt, können die Swimlanes als Organisationseinheiten bei Prozessmodellierung übernommen werden. Ein hoher Detailgrad von Aktivitätsdiagrammen muss zugunsten einer abstrakten Sicht auf Unternehmensprozesse reduziert werden.

5 Schlusswort

5.1 Zusammenfassung

Gegenstand der Arbeit war die Klassifizierung von Analyse-Methoden in Abhängigkeit von Softwareprozessen. Dabei stand das Spannungsfeld zwischen den beiden Softwareprozessen Standardsoftwareeinführung und Softwareneuentwicklung im Vordergrund. Zur Klassifizierung wurden Sichtweisen der Softwaretechnik, aus der vor allem Vorgehensmodelle zur Neuentwicklung kommen, und der Wirtschaftsinformatik, aus der die Vorgehensmodelle zur Einführung von Standardsoftware kommen, integriert.
Methodisch wurden dazu Anforderungen, die Vorgehensmodelle der beiden Softwareprozesse an Analyse-Methoden stellen, mit den Grundlagen und Ergebnissen von konkreten Analyse-Methoden verglichen.
Bei Betrachtung der Vorgehensmodelle wurde festgestellt, dass Vorgehensmodelle der Standardsoftwareeinführung Analyse tendenziell als Basis zur Reorganisation der Anwender-Organisation sehen und weniger die Veränderung und Konstruktion von Software verfolgen. Der durch die Unabänderlichkeit von off-the-shelf Standardsoftware entstehende Reorganisationszwang wird dabei als positiver Beschleuniger von organisatorischen Veränderungen gesehen. Für Vorgehensmodelle der Softwareneuentwicklung steht dagegen die Konstruktion von Software im Mittelpunkt. Die Software soll optimal an die bestehende Anwender-Organisation angepasst werden. Reorganisation auf Grund technischer Zwänge ist nicht erwünscht. Nach der Vorstellung der betrachteten Vorgehensmodelle ist für Standardsoftwareeinführungsprozesse also Reorganisation zentral und für Softwareneuentwicklungsprozesse Softwarekonstruktion.
Bei der Zuordnung von Analyse-Methoden zu Softwareprozessen standen fünf Aspekte im Vordergrund, die zum einen Unterscheidungsmerkmale für die Methoden sind und zum anderen einer der beiden Tendenzen Reorganisation und Softwarekonstruktion entsprechen und somit den Softwareprozessen zugeordnet werden können:

1. Softwareneuentwicklung → Softwarekonstruktion
    a. Klassenmodell
    b. Anwendungsfälle
    c. Softwareabläufe
2. Standardsoftwareeinführung → Reorganisation
    a. akteursübergreifende Abläufe
    b. Organisationsbrüche

Es zeigt sich, dass zwei der hier betrachteten Analyse-Methoden den Softwareneuentwicklungsprozessen zuzurechnen sind (OBA und OOA nach Heide Balzert), eine den Standardsoftwareeinführungsprozessen (GPA) und eine zwischen beiden Prozessen liegt (SA). Die Zuordnung wird durch verschiedene Erweiterungen der Methoden relativiert. So können in Modellen von Objektorientierten Analyse-Methoden auch Aspekte, die sich eher für Reorganisation eignen, mitmodelliert werden.
Die Ergebnisse der Klassifizierung wurden schließlich in einem Methodenkatalog präsentiert, der dem Praktiker Anhaltspunkte gibt, welche Analyse-Methode für welchen Softwareprozess geeignet ist und welche Analyse-Ergebnisse er bei Wechsel des Softwareprozesses weiterverwenden kann bzw. neu erarbeiten muss.

5.2 Kritik der Methode

Die in dieser Arbeit gewählte Methode betrachtet Analyse auf zwei Arten: zum einen von außen als Blackbox-Modul in einem Vorgehensmodell; zum anderen wird sie von innen betrachtet und festgestellt, was sie nach außen liefert bzw. von dort erwartet. Auf den ersten Blick scheint das Heraustrennen von Analyse aus den Vorgehensmodellen leicht möglich, da einzelne Aktivitäten (bzw. Phasen, Workflows etc.) explizit als Analyse benannt werden und zumindest in grafischen Darstellungen als Module zu existieren scheinen. Benennungen wie „Aufgabenorientierte Anforderungsermittlung“ (STEPS) oder „Analyse und Grobdesign ohne SSW“ (IBSIS-Modell) suggerieren klar abgegrenzte Aktivitäten. Die meisten Autoren betonen aber auch, dass Analyse mit anderen Aktivitäten stark verschränkt ist.
Die Grenzen zwischen Analyse und anderen Aktivitäten lassen sich besonders auf einer hohen Abstraktionsebene schwer feststellen. Die Vorgehensmodelle nennen zwar Anforderungen, stellen aber oft nicht klar, ob diese an Analyse, Design oder sonstige Aktivitäten eines Softwareprozesses gerichtet sind.
Auch die hier entwickelte Definition für Analyse hilft beim Heraustrennen aus Vorgehensmodellen kaum weiter, da der Unterschied zwischen äußeren Anforderungen an Analyse und inneren Beschreibungen einer Analyse-Methode oft nur im unterschiedlichen Abstraktionsgrad der Beschreibung liegt: Betrachtet man z.B. die Aktivitäten Unternehmensmodellierung, Geschäftsprozessmodellierung und EPK-Modellierung (genannt in absteigender Abstraktion), ist die Frage, wo die Grenze zwischen Anforderungen an die Analyse und ihrem Ergebnis zu setzen ist. Zum einen kann die Modellierung von EPKs als Anforderung an Analyse gesehen werden, die eine Analyse-Methode erfüllen kann oder nicht. Wird aber Prozessmodellierung als Anforderung genommen, erfüllen neben Methoden, die EPKs modellieren, auch Methoden, die Petri-Netze benutzen, die Anforderungen. Sieht man gar Unternehmensmodellierung als äußere Anforderung, wird diese auch von Methoden, die die Aufbauorganisation modellieren oder eine objektorientierte Unternehmensmodellierung vornehmen, erfüllt. Die Wahl der Grenze ist in gewissem Maße willkürlich.
Ein weiteres Beispiel ist die für die hier betrachteten Vorgehensmodelle unterschiedliche Bedeutung des Begriffs Geschäftsprozess. Bis auf STEPS fordern alle Vorgehensmodelle die Feststellung von Geschäftsprozessen in der Anwenderorganisation. Hier entsteht der Eindruck, dass der RUP, das Vorgehensmodell von Helmut Balzert, das IMG-Modell und das IBSIS-Modell in diesem Punkt dieselben Anforderungen an Analyse haben. Geht man aber etwas mehr ins Detail, fällt auf, dass Balzert und RUP an Anwendungsfälle denken, während das IBSIS-Modell und das IMG-Modell an die Modellierung der Ablauforganisation und ganz konkret an die Modellierung mit EPKs denken. Dass Anwendungsfälle und Geschäftsprozesse unter dem Begriff Usecase zum Teil synonym gebraucht werden, hat bei einigen Autoren dazu geführt, auch die sehr verschiedenen Modelle zu deren Modellierung gleichzusetzen (insbesondere EPKs und Usecase-Diagramme, siehe z.B. [VER00]).
Wählt man die Anforderungen zu abstrakt, erfüllen fast alle Analyse-Methoden die Anforderungen aller Vorgehensmodelle. Geht man zu sehr ins Detail, bekommt man als Anforderungen konkrete Methoden. In dieser Arbeit wurde deswegen versucht, einen Mittelweg zu gehen: Die Zuordnung von Analyse-Methoden zu Vorgehensmodellen bzw. Softwareprozessen geschieht über die Methoden-Aspekte, die gleichzeitig als Ausprägungen der Tendenzen der Vorgehensmodelle identifiziert werden.
Das Problem würde sich nicht ergeben, wenn die Vorgehensmodelle selbst klar zwischen den Zielen, die durch Analyse erreicht werden sollen, und der Nennung von konkreten Analyse-Methoden unterscheiden würden. Dieser Modularisierung stehen aber zur Zeit noch

Vorgehensmodelle aus einem Guss gegenüber: Die Modelle halten sich begrifflich in bestimmten „Universen“ auf. Dies zeigt aber auch, dass Analyse auf den ganzen Prozess ausstrahlt. Die Dinge, die in der Analyse identifiziert werden, seien es Usecases, Geschäftsprozesse oder Objekte, prägen den ganzen Prozess. Eine bessere Modularisierung wird aber auf Grund der immer komplexeren Vorgehensmodelle notwendig.

5.3 Ausblick

Die Zukunft der Analyse wird auf vielen Ebenen weitergehen: Die Offenlegung von Vor- und Nachteilen der verschiedenen Methoden und ihrer Komponenten muss auf breiterer Ebene als bisher geschehen.
Es genügt nicht mehr zu sagen, dass eine Methode wie die Objektorientierte Analyse nun im Gegensatz zu früheren Methoden besser mit veränderlichen Anforderungen umgehen kann, sondern es müssen auch die Auswirkungen auf die Benutzer und die Anwender-Organisation offengelegt werden. Helmut Balzert versucht in seinem Vorgehensmodell Auswahlkriterien für verschiedene Methoden zur Verfügung zu stellen [BALZ96]. Seine Kriterien sind aber rein technischer Natur. Viele Fragen in Bezug auf Analyse-Methoden sind noch nicht beantwortet: Gibt es z.B. nachteilige Auswirkungen von Objektorientierten Entwicklungsmethoden auf die Anwender-Organisation? Andererseits hat Mertens für die Wirtschaftsinformatik schon 1996 gefragt, ob die starke Konzentration auf Prozesse nicht Nachteile bringt: „Process focus considered harmful?“ [MER96]. Dass man allein auf der Basis von Prozessen keine gute Software entwickeln kann, scheint durch den Untergang der strukturierten Programmierung und den Siegeszug der Objektorientierung bestätigt zu werden. Dennoch sind Prozesse (oder auch Workflows, Serviceflows etc.) auch in der Softwareentwicklung wichtig. Reicht die Modellierung von Anwendungsfällen, die nicht akteursübergreifend sind, zur Softwareentwicklung aus, wenn Softwaresysteme immer stärker vernetzt sind?
Im hier vorgestellten Methodenkatalog wurde bereits angedeutet, dass für einen angestrebten Softwareprozess eine Kombination aus einzelnen Aktivitäten, Modellen und Diagrammen verschiedener Analyse-Methoden sinnvoll sein kann. Auch hierzu müssen aber die Zielsetzung sowie Vor- und Nachteile der einzelnen Komponenten klar sein und im Vergleich zueinander dargestellt werden. (s.a. [HOF00])

Eine bessere Katalogisierung der verschiedenen Möglichkeiten der UML und ihrer Anwendung in Analyse-Methoden in Bezug auf unterschiedliche Softwareprozesse muss vorgenommen werden. Softwaretechniker sollten sich insbesondere mehr um den Bereich business modeling von UML kümmern. In der Wirtschaftsinformatik ist die Akzeptanz von UML zur Zeit noch gering. Wirtschaftsinformatiker sollten untersuchen, inwieweit sie UML für ihre Belange verwenden können. So könnte in Zukunft möglicherweise auch bei Standardsoftwareeinführungsprozessen mit UML modelliert werden.

Nicht betrachtet wurden in dieser Arbeit Analysemuster und Referenzmodelle. Bei der Analyse in Standardsoftwareeinführungsprozessen werden meistens Referenzmodelle verwendet (s.a. [SCH96]). Bei Softwareneuentwicklungsprozessen werden verstärkt Analysemuster verwendet (z.B. [BAL99], [FOW99], [SES00]). Die Referenzmodelle für Standardsoftwareeinführung sind dabei im Normalfall Prozessmodelle, während Analysemuster Objektmodelle und Klassenmodelle sind. Beide sollen die Durchführung der Analyse beschleunigen. Analog zur Fragestellung dieser Arbeit wäre interessant zu betrachten, wie Referenzmodelle und Analysemuster in Abhängigkeit von verschiedenen Softwareprozessen

verwendet werden sollten. Auch hier muss wieder eine Verbindung der Forschungsrichtungen Wirtschaftsinformatik und Softwaretechnik vollzogen werden.

Der in dieser Arbeit synthetisierte Methodenkatalog konzentriert sich auf nur fünf Aspekte von Analyse-Methoden. Viele andere Aspekte, wie z.B. Akteurskonstellationen, implizite Perspektiven usw. (siehe z.B. [ROL98], [JAY94]) bleiben weitgehend außerhalb der Betrachtung. Dies liegt zum Teil an der notwendigen Begrenzung des sehr umfangreichen Gegenstandsbereiches, zum Teil aber auch an den betrachteten Analyse-Methoden. Für die Zukunft sollte der Methodenkatalog deswegen sowohl um weitere Methoden als auch um weitere Aspekte ergänzt werden.

Hinter dem Begriff Reorganisation steht in dieser Arbeit vor allem prozessorientierte Reorganisation. Die Konzentration auf Geschäftsprozesse hat ihre Ursache in den Vorgehensmodellen zur Einführung von Standardsoftware bzw. in der Wirtschaftsinformatik. Die Vielfalt der Sichtweisen auf Organisation könnte in Zukunft zu ganz anderen Analyse-Methoden führen (s.a. [MER96], [KIE95], [ROL98]). Einige Ansätze, die Objektorientierte Analyse zur Unternehmensmodellierung verwenden, sind neben der Prozesssicht noch in der Minderheit (z.B. [KOH96], [FES96]).

Literaturliste

[BAL99] Balzert, Heide 1999: Lehrbuch der Objektmodellierung: Analyse und Entwurf. Heidelberg, Berlin: Spektrum

[BALZ96] Balzert, Helmut 1996: Lehrbuch der Software-Technik. Software-Entwicklung, Heidelberg/ Berlin/ Oxford: Spektrum

[BALZ98] Balzert, Helmut 1998: Lehrbuch der Software-Technik. Software-Management Software-Qualitätssicherung Unternehmensmodellierung, Heidelberg/ Berlin: Spektrum

[BAR96] Barbitsch, Ernst 1996: Einführung integrierter Standardsoftware: Handbuch für eine leistungsfähige Unternehmensorganisation, München/ Wien: Hanser

[BJO97] Bjørner, Dines 1997: Domains as a Prerequisite for Requirements and Software Domain Perspectives & Facets, Requirements Aspects and Software Views, in: Broy, Manfred/ Rumpe Bernhard (Hrsg.): Lecture Notes in Computer Science, Nr.1526, Requirements Targeting Software and Systems Engineering, S.1-41, Bernried: Springer

[BLU97] Blume, Andreas 1997 AFOS: Projektkompass SAP. Arbeitsorientierte Planungshilfen für die erfolgreiche Einführung von SAP-Software, Braunschweig/ Wiesbaden: Vieweg

[BOE88] Boehm, B. W. 1988: A spiral model for software development and enhancement, IEEE Computer, 21(5), Mai, S.61-72

[BRH95] Brenner, Walter/ Hamm, V. 1995: Prinzipien des Business Reengineering, in: Brenner und Keller 1995: Business Reengineering mit Standardsoftware, S.17- 43, Frankfurt

[BRÖ95] Bröckers, Alfred 1995: Process-Based Software Risk Assessment, in: Schäfer, Wilhelm (Hrsg.): Lecture Notes in Computer Science 913, Software Process Technology, S.9-29, Berlin/ Heidelberg: Springer

[CHE81] Checkland, P. 1981: Systems Thinking, Systems Practice. Wiley, Chichester

[CHR94] Chroust, Gerhard 1994: Navigation in Process Models, in: Lecture Notes in Computer Science 772, S.119-122, Berlin/ Heidelberg: Springer

[CY90] Coad & Yourdon 1991: Objekt-Orientierte Analyse

[DEM79] DeMarco, Tom 1979: Structured Analysis and Systems Specification. Englewood Cliffs, NJ: Prentice Hall

[DKW99] Derniame, Jean-Claude/ Kaba, Ali Badara/ Waryboy, Brian/ Montangero, Carlo (Hrsg.) 1999: The Software Process: Modelling and Technology, in: Lecture Notes in Computer Science 1500, S.1-14, Berlin, Heidelberg: Springer

[DUD93] Engesser, Hermann (Hrsg.)/ Claus, Volker/ Schwill, Andreas 1993: Duden Informatik. Ein Sachlexikon für Studium und Praxis

[DÜH00] Düwel, Stephan / Hesse, Wolfgang 2000: Bridging the gap between Use Case Analysis and Class Structure Design by Formal Concept Analysis, in: Ebert, Jürgen/ Frank, Ulrich (Hrsg.): Modelle und Modellierungssprachen in Informatik und Wirtschaftsinformatik. Beiträge des Workshops “Modellierung 2000” St. Goar, 5.-7. April 2000, Koblenz: Fölbach

[DWF84] Deutsches Wörterbuch Fremdwörterlexikon 1984, Vaduz/ Liechtenstein: Zweiburgen

[FES96] Ferstl, Otto K./ Sinz, Elmar J. 1996: Geschäftsprozessmodellierung im Rahmen des Semantischen Objektmodells, in: Vossen, Gottfried/ Becker, Jörg: Geschäftsprozessmodellierung und Workflow-Management, S. 47-62, Bonn/ Albany: Internat. Thomson Publ.

[FKRW97] Floyd, C./ Krabbel, A./ Ratuski, S./ Wetzel, I. 1997: Zur Evolution der evolutionären Systementwicklung: Erfahrungen aus einem Krankenhausprojekt, in: Informatik-Spektrum 20, S.13-20: Springer

[FLK98] Floyd, Christiane/ Klischewski, Ralf 1998: Informatik - eine Standortbestimmung. Informatik - Mensch - Gesellschaft 1. Prüfungsunterlagen, Hamburg: Universität Hamburg, Fachbereich Informatik

[FLO84] Floyd, Christiane 1984: A Systematic Look at Prototyping, in: Budde, R./ Kuhlenkamp, K./ Mathiassen, L. /Züllighoven, H. (Hrsg.): Approaches to Prototyping, S.1-18, Berlin: Springer

[FLZ97] Floyd, Christiane/ Züllighoven, Heinz 1997: Softwaretechnik, in: Rechenberger/ Pomberger: Informatik Handbuch, S.641-667, München: Hanser

[FOW99] Fowler, Martin 1999: Analysemuster. Wiederverwendbare Objektmodelle, Bonn: Addison Wesley Longman

[FRS89] Floyd, C./ Reisin, F.-M./ Schmidt, G. 1989: STEPS to Software Development with Users, in: Ghezzi, C./ McDermid, J.A. (Hrsg.): ESEC `89, Lecture Notes in Computer Science, Nr. 387, S.48-64: Springer

[GCT99] Gable, G. G./ Corbitt, Gail/ Tanlamai, U./ u.a. 1999: ERP-software – characteristics and consequences, in: Pries-Heje, Jan/ Ciborra, Claudio/ u.a.: 7th european conference on information systems proceedings, ECIS ´99, Volume I, S.1038-1043, Copenhagen: Copenhagen Business School

[GMD92] German Ministry of Defence 1992: V-Model. Software lifecycle process model, General Reprint No. 250: Bundesminister des Innern, Koordinierungs- und Beratungsstelle der Bundesregierung für Informationstechnik in der Bundesverwaltung

[GOD95] Godart, Claude/ Dietrich, D. 1995: Stepwise specification of interactive processes in COO, in: Schäfer, Wilhelm (Hrsg.): Lecture Notes in Computer Science 913, Software Process Technology, S.220-239, Berlin/ Heidelberg: Springer

[GRU94] Gruhn, Volker 1994: Software Process Management and Business Process (Re-) Engineering, in: Warboys, B. C. (Hrsg.): Lecture Notes in Computer Science. Software Process Technology, Berlin, Heidelberg, New York: Springer

[GRY96] Gryczan, Guido 1996: Prozeßmuster zur Unterstützung kooperativer Tätigkeit, Wiesbaden: Dt. Univ.-Verl.

[HAN96] Hansen, Hans Robert 1996: Wirtschaftsinformatik I, 7.Aufl., Stuttgart: Lucius und Lucius

[HEB90] Heinrich, J. / Burgholzer, P. 1990: Systemplanung II, 4.Aufl., München/ Wien

[HEB98] Hess, Thoma/ Brecht, Leo 1996: State of the Art des Business Process Redesign. Darstellung und Vergleich bestehender Methoden (2. Auflage), Wiesbaden

[HIR83] Hirschheim, Rudy 1983: Assessing Participative System Design. Some Conclusions From an Exploratory Study. Information & Management, 6 (1), S.317-327.

[HKL95] Hirschheim, Rudy/ Klein, Heinz K. / Lyytinen, Kalle 1995: Information Systems Development and Data Modeling. Conceptual and Philosophical Foundations, Cambridge: Cambridge University Press

[HLG99] Holland, C. P./ Light, Ben/ Gibson, Nicola 1999: A critical success factors model for enterprise resource planning implementation, in: Pries-Heje, Jan/ Ciborra, Claudio/ u.a.: 7th european conference on information systems proceedings, ECIS ´99, Volume I, S.273-287, Copenhagen: Copenhagen Business School

[HMF92] Hesse, W./ Merbeth, G./ Frölich, R. 1992: Handbuch der Informatik. Software-Entwicklung. Vorgehensmodelle, Projektführung, Produktverwaltung, München, Wien: Oldenbourg

[HOF00] Hofmann, Huber F. 2000: Requirements engineering. a situated discovery process, Wiesbaden: Gabler

[HRS99] Hagemeyer, Jens/ Rolles, Roland/ Scheer, August-Wilhelm 1999: Der schnelle Weg zum Sollkonzept: Modellgestützte Standardsoftwareeinführung mit dem ARIS Process Generator, Heft 152, März, Saarbrücken: Universität des Saarlandes, Institut für Wirtschafsinformatik

[HSR98] Hung, Kitty/ Simons, Tony/ Rose, Tony 1998: http://www.dcs.shef.ac.uk/~kitty /OOIS98_Submission.htm, 11.1.2001, 19:16 Uhr

[JAC92] Jacobson, Ivar 1992: Object-Oriented Software Engineering. A Use Case Driven Approach, Wokingham/ Reading / Bonn: Addison-Wesley

[JAY94] Jayaratna, Jimal 1994: Understanding and Evaluating Methodologies. NIMSAD. A Systemic Framework, Berkshire, England: McGraw-Hill

[JBR98] Jacobson, Ivar/ Booch, Grady/ Rumbaugh, James 1998: The Unified Software Development Process. The complete guide to the Unified Process from the original designers, Reading, Bonn, Sydney: Addison-Wesley

[KEI89] Keil-Slawik, Reinhard 1989: Systemgestaltung mit Aufgabennetzen, Berlin, in: Maaß, S./ Oberquelle, H.: Software-Ergonomie ´89. Aufgabenorientierte Systemgestaltung und Funktionalität, S.123-133, Stuttgart: Teubner

[KEN92] Kendall, P. 1992: Introduction to Systems Analysis and Design. A Structured Approach, USA: W.C. Brown Publishers

[KIE95] Kieser, Alfred (Hrsg.) 1995: Organisationstheorien, 2. Aufl., Stuttgart/Berlin/Köln: Kohlhammer

[KIR96] Kirchmer, Mathias 1996: Geschäftsprozessorientierte Einführung von Standard-software, Wiesbaden: Gabler

[KLI99] Klischewski, Ralf 1999/2000: Softwareentwicklung und –einführung in Organisationen. Skript zur Vorlesung WS 1999/2000: Arbeitsbereich Softwaretechnik im Fachbereich Informatik an der Universität Hamburg

[KLU89] Kluge, Friedrich (Hrsg.)/ Elmar Seebold 1989: Etymologisches Wörterbuch der deutschen Sprache, 22.Aufl., Berlin, New York: Walter de Gruyter

[KLW99] Klischewski, Ralf/ Wetzel, Ingrid 1999: Cooperation Modeling for Relating Systems Development and Organizational Change, Hamburg: Arbeitsbereich Softwaretechnik, Fachbereich Informatik, Universität Hamburg

[KOH96] Kohl, Claudia 1996: Objektorientierte Analysekonzepte in der Unter-nehmensmodellierung, in: Vossen, Gottfried/ Becker, Jörg: Geschäftsprozess-modellierung und Workflow-Management, S. 63-80, Bonn/ Albany: Internat. Thomson Publ.

[KRU99] Kruchten, Philippe 1999: Der Rational Unified Process. Eine Einführung, München: Addison-Wesley-Longman

[KWR96] Krabbel, A./ Wetzel, I./ Ratuski, S. 1996: Objektorientierte Analysetechniken für übergreifende Aufgaben. Softwaretechnik'96, Beiträge der GI-Fachtagung, Koblenz 12.-13.September 1996, S.65-72

[LAN73] Langefors, Börje 1973: Theoretical analysis of information systems, Philadelphia: Auerbach publishers

[MER96] Mertens, Peter 1996: Process focus considered harmful?, in: Wirtschaftsinformatik 38, Wiesbaden, S.446-447

[MEI95] Meinhardt Stefan 1995: Geschäftsprozeßorientierte Einführung von Standard-Software am Beispiel des SAP-Systems „R/3", in: Wirtschaftsinformatik 37, Wiesbaden, S.487-499

[MEY97] Meyer, Bertrand 1997: Object-oriented software construction, 2.Ausg, New Jersey: Prentice Hall

[MUM83] Mumford, E. 1983: Designing Human Systems – The ETHICS Method. Manchester: Manchester Business School

[NOS99] Noack, Jörg/ Schienmann, Bruno 1999: Objektorientierte Vorgehensmodelle im Vergleich, in: Informatik-Spektrum 22, S.166-180: Springer

[ÖST96] Österle, Hubert 1996: Das Geschäftsmodell im Informationszeitalter, in: Wirtschaftsinformatik 38, Wiesbaden, S.447-449

[PFL98] Pfleeger, Shari Lawrence 1998: Software Engineering. theory and practice, London/ Sydney/ Toronto: Prentice Hall

[POB93] Pomberger, Gustav/ Blaschek, Günther 1993: Software Engineering. Prototyping und objektorientierte Software-Entwicklung, 2.Aufl, München: Hanser

[POM00] Pommerehn, Julia 2000: Sinnlichkeit der Wahrnehmung und der Darstellung in Rainer Maria Rilkes „Dinggedichten“, Hamburg: Universität Hamburg

[POR85] Porter, M.E. 1985: Competitive Advantage. Creating and Sustaining Superior Performance, New York: The Free Press

[RAT97] Ratuski, Sabine 1997: Vorgehensweise zur Auswahl von Standardsoftware am Beispiel von Krankenhausinformationssystemen. Diplomarbeit, Universität Hamburg: Fachbereich Informatik

[RBA91] Robinson, M./ Bannon, L. 1991: Questioning representations, in: Proceedings of ECSCW ´91 (Amsterdam, The Netherlands, Sep. 1991), S.219-233

[ROB97] Robindosn, M. 1997: As real as it gets. Taming models and reconstructing procedures, in: Bowker, G. (Hrsg.): Social science, technical systems, and cooperative work. beyond the great divide, Mahwah NJ, S.257-274: Lawrence Erlbaum Associates

[ROL98] Rolf, Arno 1998: Grundlagen der Organisations- und Wirtschaftsinformatik, Berlin/Heidelberg/New York: Springer

[ROS79] Ross, D.T./ Schoman, K.E. 1979: Structured Analysis for Requirements Definition, in: Yourdon, Edward (Hrsg.): Classics in Software Engineering, New York: Yourdon Press

[ROV95] Rombach, H. Dieter/ Verlage, Martin 1995: Directions in Software Process Research, in: Zelkowitz, Marvin (Hrsg.): Advances in Computers, Volume 41, S.2-65

[RUG92] Rubin, Kenneth S./ Goldberg, Adele 1992: Object Behavior Analysis, in: CACM, 9/1992, Vol.35, No.9, S.48-62

[SAN85] Sandberg, A. 1985: Socio-technical Design. Trade Union Strategies and Action Research, in: Mumford, E./ Hirscheim, R./ Fitzgerald, G./ Wood-Harper, A. T. (Hrsg.) Research Methods in Information Systems. S. 79-92, North-Holland, Amsterdam

[SCH96] Scheruhn, Hans-Jürgen 1996: Maßgeschneidert. Referenzmodelle – Hilfe bei der Einführung von Standardsoftware, in: iX Multiuser Multitasking Magazin, 1/96, S.112-119: Heise

[SCH98] Schmitz, Reiner 1998: Einführung von PPS-Systemen, in: Luczak, Holger (Hrsg.)/ Eversheim, Walter (Hrsg.)/ Schotten, Martin: Produktionsplanung und -steuerung: Grundlagen, Gestaltung und Konzepte, 1. Aufl., Berlin/ Heidelberg/ New York: Springer

[SCHE98] Scheer, August-Wilhelm 1998: ARIS – vom Geschäftsprozess zum Anwendungssystem (3.Aufl), Berlin

[SCHW96] Schewe, Bettina 1996: Kooperative Softwareentwicklung. Ein objektorientierter Ansatz, Wiesbaden: Dt.Univ.-Verl.

[SES00] Šešera, L’ubor 2000: Analysis Patterns, in: Hlaváč, Václav (Hrsg.): Lecture Notes in Computer Science. SOFSEM 2000: Theory and Practice of Informatics, Nr. 1963, S.129-151

[STA95] The Standish Group 1995: Sample Research Paper, www.standishgroup.com, auch zitiert in: OBJEKTspektrum 6/2000: Erfolgsfaktoren eines Softwarearchitekten, S.36,37

[STA99] Staud, Josef 1999: Geschäftsprozeßanalyse mit Ereignisgesteuerten Prozeßketten. Grundlagen des Business Reengineering für SAP R/3 und andere Betriebswirtschaftliche Standardsoftware, Berlin/ Heidelberg/ New York: Springer

[STH99] Stahlknecht, Peter/ Hasenkamp, Ulrich 1999: Einführung in die Wirtschaftsinformatik, 9. vollst. überarb. Aufl., Berlin/Heidelberg/New York: Springer

[SZA98] Szallies, Constantin 1998: Software-Spezifikation mit ereignisgesteuerten Prozeßketten (EPK), in: GI Softwaretechniktrends, Band 18 Heft 2, Mai 98, S.38-43

[TEW98] Tewis, Holger 1998: Auswahl von betriebswirtschaftlicher Standardsoftware. Studienarbeit, Fachbereich Informatik, Universität Hamburg

[TIE96] Tiemeyer, Ernst 1996: OrgTools. AfürO – Softwareführer für die Organisationsarbeit, Stuttgart: Schäffer-Poeschel

[TOC95] Totland, Terje/ Conradi, Reidar 1995: A Survey and Comparison of Some Research Areas Relevant to Software Process Modeling, in: Schäfer, Wilhelm (Hrsg.): Lecture Notes in Computer Science 913, Software Process Technology, S.65-69, Berlin/ Heidelberg: Springer

[UML99] OMG, June 1999: OMG Unified Modeling Language Specification, Version 1.3, www.omg.org

[VER00] Versteegen, Gerhard 2000: Projektmanagement mit dem rational unified process, Berlin/ Heidelberg/ New York: Springer

[VER94] Verlage, Martin 1994: Multi-View Modeling of Software Processes, in: Lecture Notes in Computer Science 772, S.119-122, Berlin/ Heidelberg: Springer

[WAS96] Wasserman, Anthony I. 1996: Toward a discipline of software engineering, IEEE Software, 13(6), Nov., S.23-31

[WIL90] Wildemann, H. 1990: Einführungsstrategien für die computerintegrierte Produktion (CIM), München

Erklärung

Hiermit versichere ich, dass die vorliegende Diplomarbeit von mir selbstständig erstellt wurde und ich außer den angegebenen Quellen keine fremden Hilfsmittel verwendet habe.

Hamburg, 21. Februar 2001

_____

Holger Tewis