Komponentenorientierte Vorgehensmodelle im Vergleich

Peter Fettke, Iulian Intorsureanu, Peter Loos

Technische Universität Chemnitz, Fakultät für Wirtschaftswissenschaften, Information Systems & Management (Professur Wirtschaftsinformatik II), D-09107 Chemnitz, Germany, Tel.: +49/371/531-4375, Fax: -4376, E-Mail: peter.fettke@isym.tu-chemnitz.de, iintorsu@inforec.ase.ro, loos@isym.tu-chemnitz.de, WWW: http://www.isym.tu-chemnitz.de/

Zusammenfassung. Werden Softwaresysteme auf Basis eines komponentenorientierten Architekturparadigmas entwickelt, stellt sich die Frage, welches Vorgehensmodell zur Projektabwicklung herangezogen werden kann. In der Literatur werden unterschiedliche Vorgehensmodelle zur komponentenorientierten Softwareentwicklung vorgeschlagen. Aus diesen werden in der vorliegenden Untersuchung vier Modelle ausgewählt: Catalysis, Perspective, Rational Unified Process 2002 und V-Modell ’97. Die ausgewählten Vorgehensmodelle werden auf Basis eines allgemeinen Rahmens beschrieben und verglichen. Dabei werden die Aspekte Terminologie, Klassifizierung, Komponentenbegriff, Abdeckung des Lebenszyklus einer Komponente, Abdeckung der Tätigkeitsbereiche, Prozessarchitektur, Prozesssteuerung, Rollenabdeckung und Adaption untersucht. Komponentenorientierte Vorgehensmodelle sind sowohl Weiterentwicklungen bekannter konventioneller Vorgehensmodelle als auch ausschließlich auf die komponentenorientierte Entwicklung ausgerichtet. Obwohl die ausgewählten Vorgehensmodelle speziell auf eine komponentenorientierte Entwicklung ausgerichtet sind, zeigt sich, dass wesentliche Lebenszyklen einer Komponente nur rudimentär behandelt werden. Eine Ausnahme bildet hier Catalysis.

Schlüsselworte: Catalysis, Perspective, Rational Unified Process 2002, V-Modell ’97, Methodenvergleich, Komponente, Component Engineering, Prozessmodell

1 Ausgangssituation und Problemstellung

Die Idee, gesamte Softwaresysteme aus mehr oder weniger vorgefertigten Bausteinen zu entwickeln, ist bereits mehrere Jahrzehnte alt und wurde in der Vergangenheit immer wieder aufgegriffen.[Same1997; Grif1998; Szyp1999] Dabei zeigte sich, dass verschiedene Autoren unterschiedliche Auffassungen an den Begriff der Komponente legten. Neben den Aspekten der Granularität einer Komponente wurden weitere Eigenschaften wie Wiederverwendbarkeit, Abgeschlossenheit, Vermarktbarkeit, Kombinierbarkeit usw. in der Literatur diskutiert.[Turo2001, S. 15-19] In jüngster Zeit zeigte sich, dass gewisse Übereinkünfte in der Literatur erzielt werden konnten, was unter einer Komponente zu verstehen ist. Dies ermöglichte u. a. im Arbeitskreis 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik eine einheitliche Definition des Begriffes Komponente zu formulieren:

„Eine Komponente besteht aus verschiedenartigen (Software-)Artefakten. Sie ist wiederverwendbar, abgeschlossen und vermarktbar, stellt Dienste über wohldefinierte Schnittstellen zur Verfügung, verbirgt ihre Realisierung und kann in Kombination mit anderen Komponenten

eingesetzt werden, die zur Zeit der Entwicklung nicht unbedingt vorhersehbar ist. [im Original z. T. kursiv gesetzt, die Autoren]“[Acke+2002, S. 1]

In Theorie und Praxis sind seit Beginn des Software Engineering unterschiedliche Vorgehensmodelle bekannt.[Brem1998] Ein Vorgehensmodell kann verstanden werden als „ein Muster zur Beschreibung eines Entwicklungsprozesses [im Original z. T. kursiv gesetzt, die Autoren]“[FBM+1998, S. 20] für ein Softwaresystem. Die zunehmende Verwendung von Komponenten-Strategien zur Entwicklung von Softwaresystemen wirft die Frage auf, wie Vorgehensmodelle zur Entwicklung derselben ausgestaltet sind bzw. sein sollten. Die Durchsicht der Literatur zeigt, dass verschiedene Vorgehensmodelle zur Entwicklung von Komponenten vorgeschlagen werden.

Ziel des vorliegenden Beitrages ist es, einen vergleichenden Überblick über ausgewählte Vorgehensmodelle zur komponentenorientierten Entwicklung von Softwaresystemen zu geben. Dabei sollen insbesondere diejenigen Aspekte berücksichtigt werden, die im Hinblick auf die komponentenorientierte Entwicklung eine besondere Rolle einnehmen. Es sollen die vorgestellten Vorgehensmodelle nicht im Detail behandelt werden. Dazu sei auf die Originalquellen verwiesen. Der Vergleich der Vorgehensmodelle ist systematisch, da er auf Basis eines einheitlichen Vergleichsrahmens durchgeführt wird. (Ein nicht-systematischer Vergleich, sondern allgemeiner Überblick über komponentenorientierte Vorgehensmodellen findet sich bei [Turo2001, S. 110-121]).

Die Arbeit ist wie folgt aufgebaut: Kapitel 2 erläutert kurz die Auswahlentscheidung der betrachteten Vorgehensmodelle und gibt dann einen kurzen Überblick über diese. Im Kapitel 3 wird der Vergleichsrahmen für die Untersuchung der Vorgehensmodelle beschrieben. Kapitel 4 stellt die Vorgehensmodelle vergleichend gegenüber. Im abschließenden Kapitel 5 wird ein Resümee der Untersuchung gezogen und ein Ausblick auf weitere Fragestellungen gegeben.

2 Überblick über die betrachteten Vorgehensmodelle

2.1 Auswahl der Vorgehensmodelle

Grundlage der Auswahl der untersuchten Vorgehensmodelle war eine Literaturrecherche. Um in das Untersuchungsfeld aufgenommen zu werden, wurden folgende Kriterien angelegt:

- Das Vorgehensmodell sollte explizit eine komponentenorientierte Vorgehensweise unterstützen. Dabei sollte der in Kapitel 1 eingeführte Komponentenbegriff als Grundlage dienen. Dieser musste mit dem jeweiligen Komponentenbegriff der Autoren der betrachteten Vorgehensmodelle zumindest in grundsätzlichen Merkmalen übereinstimmen. Diese Einschränkung führt dazu, dass innerhalb dieser Untersuchung keine Aussagen getroffen werden können, wie nicht-komponentenorientierte Vorgehensmodelle eine komponentenorientierte Entwicklung unterstützen. Durch dieses Kriterium werden bspw. Vorgehensmodelle wie Extreme Programming [Beck2000; Beck1999; BeFo2000], Crystal [o.V.2001] oder Personal Software Process [Hump1995] ausgeschlossen, deren Protagonisten zwar zum Teil feststellen, dass auf Basis dieser Modelle eine komponentenorientierte Vorgehensweise möglich ist, allerdings keine speziellen Konzepte und Methoden dafür einführen.
- Das Vorgehensmodell sollte von vornherein nicht ausschließlich ausgewählte Aktivitäten in der Entwicklung unterstützen, sondern einen breiten Rahmen von Entwick-

lungsphasen abdecken. Ferner sollten die beschriebenen Aktivitäten in einer hinreichenden Tiefe dargelegt werden, um deren unmittelbare Anwendung zu erlauben. Diese Einschränkung führt dazu, dass ausschließlich Modelle betrachtet werden, die einen hohen Reifegrad erreicht haben, um realistischerweise als Grundlage zur Entwicklung in der industriellen Praxis herangezogen werden zu können, ohne zunächst eine tiefgreifende Ausarbeitung und Detaillierung notwendig werden zu lassen. Aufgrund dieser Einschränkung sind folgende Arbeiten nicht mit in die Untersuchung einbezogen worden: [STR2000; Burg+2000; McIn1999] beschreiben erste grundlegende Ideen für komponentenorientierte Vorgehensmodelle, [Schr2001] behandelt primär die Einführung von komponentenorientierten Vorgehensmodellen.

- Die Ausführungen sollten den Charakter eines Vorgehensmodells haben. Aufgrund dieses Kriteriums werden die Arbeiten von [Same1997; Szyp1999; HeCo2001; Grif1998; Turo2001] nicht mit in die Untersuchung einbezogen. Diese Arbeiten geben zwar einen umfassenden Überblick über Aufgaben, Techniken und Konzepte bei der komponentenorientierten Softwareentwicklung. Letztlich liefern diese Autoren allerdings keine zusammenhängende Beschreibung eines Entwicklungsprozesses für komponentenorientierte Softwaresysteme.

Bei der Literaturrecherche gefundene Vorgehensmodelle, die den definierten Kriterien genügten, sind in Bild 1 überblicksartig zusammengestellt und werden in den folgenden Abschnitten 2.2 bis 2.5 zunächst knapp erläutert.

Bezeichnung	Autor(en)	Erscheinungsjahr	Quelle(n)
Catalysis	D'Souza; Wills	1998	[DSWi1998], www.catalysis.org
Perspective	Allen; Frost	1998	[AlFr1998]
Rational Unified Process 2000	Rational Software	2001	[Rati2001]
V-Modell '97	Öffentliche Hand	1997	[o.V.1997a]

Bild 1: Betrachtete Vorgehensmodelle

2.2 Catalysis

Catalysis wurde von Desmond D'Souza und Allan Wills entwickelt. Das Vorgehensmodell ist im Jahre 1998 in [DSWi1998] beschrieben, verschiedene Informationen sind ebenso unter www.catalysis.org zu finden. Das Modell stützt sich weitgehend auf die Unified Modeling Language (UML). Catalysis beschreibt nicht einen Entwicklungsprozess, der für alle Entwicklungsprojekte gültig ist, sondern besteht vielmehr aus einer Reihe von sogenannten Prozess-Mustern (Process Patterns). Ein Prozess-Muster kann jeweils in besondern Kontexten bzw. unter bestimmten Randbedingungen angewendet werden. Darüber hinaus zeichnet sich Catalysis durch folgende Eigenschaften aus:

- Das Modell unterstützt ausschließlich eine komponentenorientierte Entwicklung. Um dies zu erreichen wird ein sogenannter Produktfamilien-Ansatz gewählt. Die Autoren

verstehen unter einer Produktfamilie mehrere Softwaresysteme, die auf einer Menge gleicher Komponenten beruhen.

- Das Modell basiert auf einem stark iterativen und inkrementellen Vorgehen, das zu sehr kurzen Entwicklungszeiten führt.
- Auch wenn der Ansatz keine durchgängige formale Spezifikation von Softwaresystemen erfordert, wird dennoch Wert auf eine hohe Qualität der Spezifikation eines Systems gelegt.

2.3 Perspective

Das Vorgehensmodell Perspective wurde ursprünglich 1994-1995 von der Firma Select (jetzt: Aonix) entwickelt. Im Jahre 1998 haben die Autoren Allen und Frost in einer Publikation ein überarbeitetes Modell vorgelegt.[AlFr1998] Gleichzeitig bietet die Firma Aonix ein zusammenhängendes Softwarepaket zur Unterstützung des Vorgehensmodells an. Perspective ist vollständig auf die Entwicklungsprozesse einer komponentenorientierten Entwicklung ausgerichtet. Das Vorgehensmodell verwendet zur Modellierung von Softwaresystemen die UML. Einen besonderen Fokus bekommt die Arbeit durch die explizite Betrachtung und Integration der Geschäftsprozessmodellierung in das Vorgehensmodell.

Das Vorgehensmodell versteht sich als eine konsolidierte Zusammenfassung einer Menge von Erfahrungen und Techniken, die sich beim Einsatz in der Praxis bewährt haben. Ein charakteristisches Merkmal des Prozesses ist die Trennung des Vorgehens in einen Prozess der Entwicklung unternehmensspezifischer Softwaresysteme einerseits, sowie der Entwicklung von Komponenten andererseits. Koordiniert werden beide Prozesse durch ein Repositorium für Komponenten, in dem diese archiviert und recherchiert werden können. Hierbei wird zwischen Benutzer-, Geschäfts- sowie Daten-Diensten unterschieden. Sowohl der Solution als auch der Component Process werden iterativ und inkrementell durchlaufen. Die Autoren weisen darauf hin, dass das Vorgehensmodell die parallele Entwicklung verschiedener Systeme bzw. Systemteile fördert.

2.4 Rational Unified Process 2000

Der Rational Unified Process (RUP) wird von der Firma Rational Software als Hypertext angeboten.[Rati2001] Dieser Untersuchung lag die Version 2002.05.00 aus dem Jahre 2001 zu Grunde. Der RUP kann als eine konkrete und detaillierte Variante des Unified Software Development Process (USDP) verstanden werden, der von [JBR1998] entwickelt wurde. Der USDP wiederum ist auf das Vorgehensmodell Objectory zurückzuführen, dass in einer frühen Version in [JCJÖ1992] beschrieben wurde. Der RUP basiert auf einer objektorientierten Modellierung mit der UML und erlaubt ein inkrementelles und iteratives Vorgehen. Neben der Hypertext-Dokumentation werden von Rational Software eine Reihe von Werkzeugen für verschiedene Aufgaben im Entwicklungsprozess angeboten. Die Integration der Beschreibung ist allerdings nicht soweit fortgeschritten, dass der RUP nur mit den angebotenen kommerziellen Werkzeugen verwendet werden kann. Vielmehr kann das Vorgehensmodell auch mit anderen Werkzeugen genutzt werden.

Der RUP ist weniger als ein konkretes Vorgehensmodell zu verstehen, sondern vielmehr als ein Prozessrahmenwerk, dass für verschiedene Einsatzgebiete angepasst werden kann. Hierbei werden von Rational unterschiedliche Möglichkeiten, sogenannte Roadmaps, vorgestellt. Un-

ter anderem existiert eine Adaption für die komponentenorientierte Vorgehensweise, auf die sich die folgende Beschreibung bezieht. Das Vorgehensmodell hat zwei Dimensionen: Zum einen wird hinsichtlich der Zeit unterschieden, in welcher Entwicklungsphase sich ein Projekt befindet. Orthogonal zu der zeitlichen Untergliederung wird hinsichtlich verschiedener Arbeitsbereiche unterschieden, die in jedem Iterationsschritt auszuführen sind.

2.5 V-Modell ’97

„Das V-Modell [Vorgehensmodell, die Autoren] ist ein anerkannter Entwicklungsstandard für IT-Systeme, der einheitlich und verbindlich festlegt, was zu tun ist, wie die Aufgaben durchzuführen sind und womit dies zu geschehen hat. Es umfasst

- das Vorgehensmodell,
- die Methodenzuordnung und
- die funktionalen Werkzeuganforderungen.

Damit ist klar umrissen, in welchen Schritten und mit welchen Methoden die Entwicklungsarbeiten auszuführen sind.“[o.V.1997b, S. 2.] Ursprünglich entwickelt wurde das V-Modell im Auftrag des Bundesministeriums für Verteidigung in Zusammenarbeit mit dem Bundesamt für Wehrtechnik und Beschaffung in Koblenz von der Industrieanlagen-Betriebsgesellschaft mbH in Ottobrunn bei München. Der aktuelle Standard ist das V-Modell ’97 (kurz: V-Modell), in dem eine Reihe von Ergänzungen und Verbesserungen vorgenommen worden sind (bspw. Straffung der Prozessdokumentation sowie Einführung von modernen Entwicklungsszenarien). Das V-Modell wird inzwischen nicht nur von Behörden verwendet, sondern hat auch in der Industrie an Verbreitung gewonnen.

Das V-Modell kennt verschiedene Szenarien, für die der Ablauf der definierten Aktivitäten in den verschiedenen Submodellen exemplarisch dargestellt wird. Von den sechs beschriebenen Szenarien wie bspw. „Inkrementelle Entwicklung“, „Grand Design“, „Objektorientierte Entwicklung“ soll im Folgenden das Szenario „Einsatz von Fertigprodukten“ herausgegriffen werden. Dieses Szenario basiert auf der Idee, den „Entwicklungsaufwand durch Nutzung bereits verfügbarer Bausteine (SW und HW) zu reduzieren“[o.V.1997a, Teil 3: Handbuchsammlung - Szenarien]

3 Vergleichsrahmen

3.1 Überblick

Die Untersuchung der Vorgehensmodelle basiert auf einem einheitlichen Vergleichsrahmen. Der Vergleichsrahmen wurde auf Grundlage von [NoSc1999] entwickelt, die einen Vergleich objektorientierter Vorgehensmodelle durchgeführt haben. Der hier verwendete Vergleichsrahmen wurde im Hinblick auf komponentenorientierte Vorgehensmodelle angepasst. Der Vergleichsrahmen besteht aus folgenden Aspekten:

- Terminologie
- Klassifizierung
- Komponentenbegriff

- Abdeckung des Lebenszyklus einer Komponente
- Abdeckung der Tätigkeitsbereiche
- Prozessarchitektur
- Prozesssteuerung
- Rollenabdeckung
- Adaption.

Der Vergleichsrahmen wird in den folgenden Abschnitten 3.2 bis 3.10 vorgestellt.

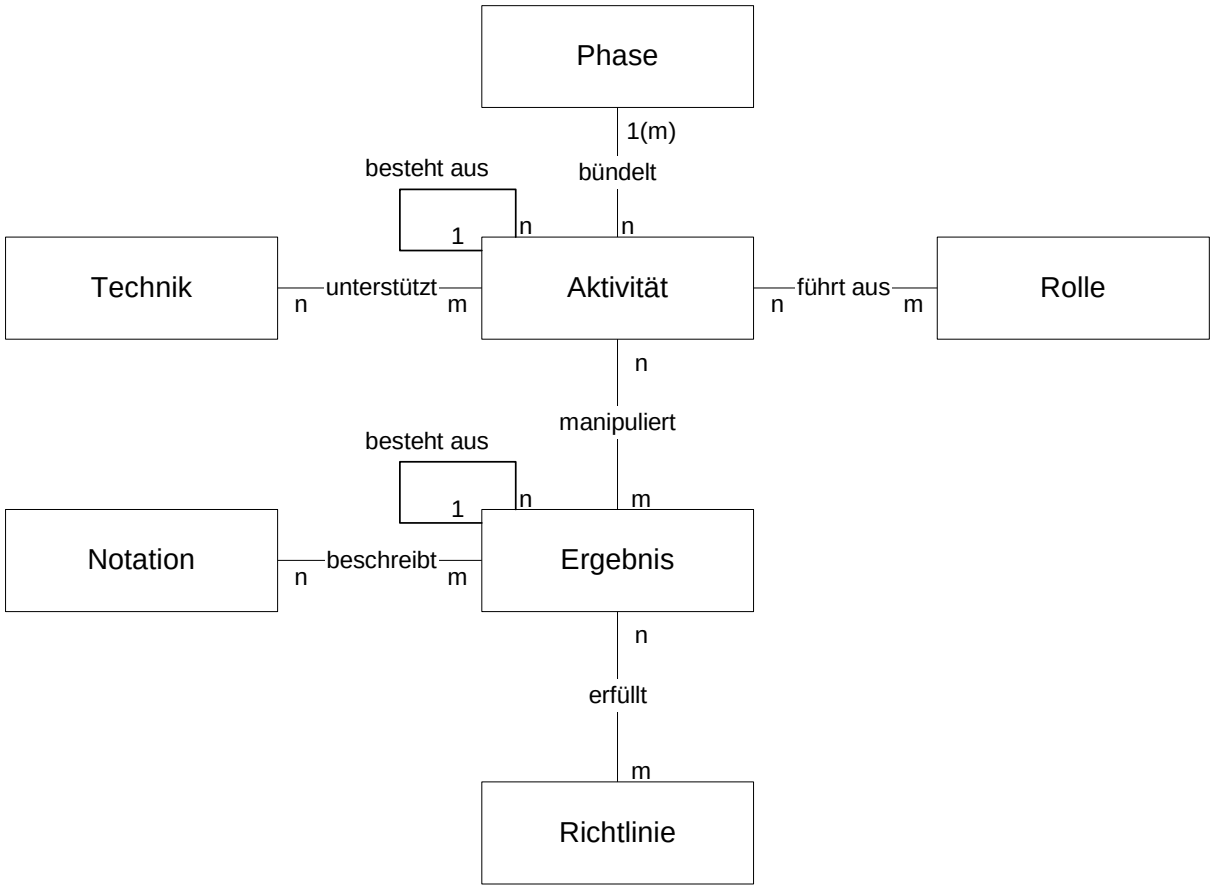

Bild 2: Grundlegende Terminologie (Quelle: [NoSc1999, S. 169])

3.2 Terminologie

Es wird folgende neutrale Terminologie für diese Untersuchung eingeführt:[NoSc1999, S. 168-170]

- Eine Phase bezeichnet eine Gruppe von Aktivitäten. Phasen sind grundlegende Einheiten zur Planung und Steuerung eines Projektes.
- Eine Aktivität ist eine Beschreibung der durchzuführenden Arbeitsschritte. Aktivitäten erhalten bestimmte Ergebnisse als Eingabe und erzeugen Ergebnisse als Ausgabe. Ferner ist es möglich, Vor- und Nachbedingungen von Aktivitäten zu bestimmen. Aktivitäten können ebenso in Unteraktivitäten aufgeteilt werden.

- Bei Ergebnissen handelt es sich um Dokumente, die bei der Durchführung von Aktivitäten erzeugt werden. Ergebnisse können sich wiederum aus Teilergebnissen zusammensetzen.
- Eine Rolle definiert die notwendigen Fähigkeiten, Kenntnisse und Erfahrungen von Personen, die am Entwicklungsprozess beteiligt sind.
- Eine Technik beschreibt, wie eine Aktivität durchzuführen ist, wohingegen eine Aktivität beschreibt, was auszuführen ist.
- Eine Richtlinie beschreibt einen Standard, der es erlaubt, Ergebnisse vergleichbarer und messbarer Qualität zu erzeugen sowie die Ergebnisse verschiedener Projekte austauschbar zu gestalten.
- Eine Notation definiert, mit welchen Zeichen die Ergebnisse in Dokumenten festgehalten werden.

In Bild 2 werden die eingeführten Begriffe im Zusammenhang graphisch dargestellt.

3.3 Klassifizierung

Komponentenorientierte Vorgehensmodelle können folgendermaßen klassifiziert werden:

- Art: Es wird hinsichtlich evolutionären und revolutionären Vorgehensmodellen unterschieden. Vorgehensmodelle werden in diesem Kontext als evolutionär bezeichnet, wenn sie durch Anpassung vorhandener Vorgehensmodelle für die komponentenorientierte Entwicklung angepasst und tauglich gemacht wurden. Dahingegen sind revolutionäre Vorgehensmodelle vollständig neu konzipiert und betrachten ausschließlich die komponentenorientierte Softwareentwicklung.
- Komponentenbildung: Es wird zwischen Top-Down- und Bottom-Up-basierten Vorgehensmodellen unterschieden.[Schr2001, S. 82] Beim Top-Down-Ansatz wird die Analyse und der Entwurf eines Softwaresystems unabhängig von bereits vorhandenen Komponenten durchgeführt. Dahingehen werden bei Bottom-Up-basierten Ansätzen schon zu Beginn des Entwicklungsprozesses vorhandene Komponenten berücksichtigt. In einem Entwicklungsprozess können beide Prinzipien der Komponentenbildung angewendet werden.
- Systemgranularität: Es wird unterschieden zwischen der Entwicklung einer einzelnen Komponente und eines vollständigen Softwaresystems, das aus mehreren Komponenten besteht. Ferner ist es denkbar, dass das Vorgehensmodell sowohl die Entwicklung einer einzelnen Komponente als auch eines vollständigen Softwaresystems unterstützen kann.

3.4 Komponentenbegriff

Der in Kapitel 1 eingeführte Komponentenbegriff kann auf verschiedene Merkmale zurückgeführt werden. Es wird untersucht, inwieweit die Vorgehensmodelle die Merkmale ebenso beschreiben:

- Software-Artefakte: Eine Komponente setzt sich aus mehreren Software-Artefakten wie Spezifikation, Implementierung, Benutzerhandbuch etc. zusammen.

- Wiederverwendbarkeit: Eine Komponente kann in verschiedenen Projekten verwendet werden.
- Abgeschlossenheit: Die Software-Artefakte, die zu einer Komponente gehören, können eindeutig abgegrenzt werden.
- Vermarktbarkeit: Eine Komponente kann prinzipiell auf Komponentenmärkten (Commercial off-the-shelf (COTS) Komponente) gehandelt werden. Alternativ ist ebenso eine Vermarktung in einem unternehmensinternen Komponenten-Repositorium denkbar.
- Dienstangebote: Eine Komponente bietet unterschiedliche Dienste wie bspw. Benutzer-, Daten- oder Geschäftsdienste an.
- Wohldefinierte Schnittstelle: Die Schnittstelle der Komponente zur Nutzung der Dienste ist explizit definiert. Dies schließt ebenso die explizite Definition der Dienste ein, die von der Komponente nachgefragt werden.
- Verbirgt ihre Realisierung: Die programmtechnische Implementierung einer Komponente ist nicht sichtbar.
- Kombinierbarkeit: Eine Komponente kann mit anderen Komponenten benutzt werden, ohne dass dies zur Entwicklungs- oder Übersetzungszeit der Komponenten geplant wurde. Vielmehr können Komponenten ohne Änderungen ihrer Implementierungen, quasi zur Laufzeit, zusammengesetzt werden.

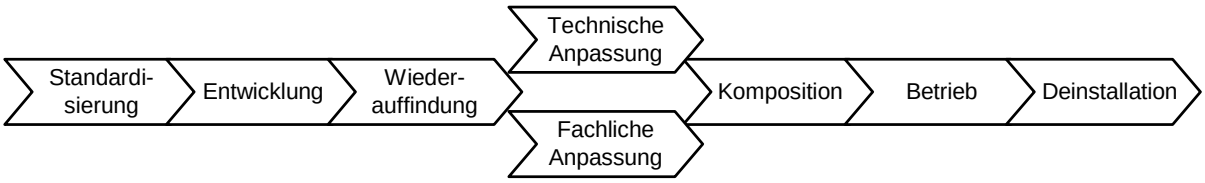

Bild 3: Lebenszyklus einer Komponente (in Anlehnung an [Turo1999, S. 13])

3.5 Abdeckung des Lebenszyklus einer Komponente

Einzelne Komponenten bzw. komponentenorientierte Softwaresysteme können hinsichtlich ihres Lebenszyklus strukturiert werden. Grundlage dieser Arbeit ist das Lebenszyklus-Konzept in [Turo2001, S. 41-48]. Dieses unterscheidet folgende Phasen:(vgl. Bild 3)

- Standardisierung: Diese Phase umfasst die Vereinheitlichung und Spezifikation von Komponenten.
- Entwicklung: Diese Phase umfasst die vollständige Realisierung einer Komponente mit Ausnahme ihrer Standardisierung.
- Technische Anpassung: Diese Phase umfasst die Behebung implementierungsbedingter, technischer Inkompatibilitäten.
- Fachliche Anpassung: Diese Phase umfasst die fachliche Parametrisierung einer Komponente.
- Komposition: Diese Phase umfasst die fachliche und technische Integration einer Komponente in ein Softwaresystem.

- Betrieb: Diese Phase umfasst alle Aufgaben zur Anpassung und zum Austausch einer Komponente nach ihrer erstmaligen Installation.
- Deinstallation: Diese Phase umfasst alle Aufgaben, die mit der Entfernung einer Komponente aus einem Softwaresystem zusammenhängen.

Dieses Lebenszyklus-Konzept wird ergänzt um die Phase der Wiederauffindung, der im Rahmen der komponentenorientierten Softwareentwicklung eine besondere Bedeutung beigemessen wird.[FeLo2000; FeLo2001] Diese Phase umfasst sowohl die Suche nach Komponenten, die Selektion grundsätzlich geeigneter Komponenten, die Auswahl einer Komponente und die Beschaffung der ausgewählten Komponente. Diese Phase wird nach der Phase Entwicklung in das Lebenszyklus-Konzept eingeführt.

3.6 Abdeckung der Tätigkeitsbereiche

Die Aktivitäten, die im Rahmen von Vorgehensmodellen definiert werden, können auf einer allgemeinen Ebene vier Tätigkeitsbereichen zugeordnet werden:[FBM+1998, S. 16-26]

- Projektmanagement: Dieser Tätigkeitsbereich umfasst die Planung, Steuerung und Kontrolle eines Softwareentwicklungsprojektes.
- Konfigurationsmanagement: Dieser Tätigkeitsbereich umfasst die „Identifikation der Konfigurationen eines Systems zu diskreten Zeitpunkten zum Zwecke der systematischen Steuerung von Konfigurationsänderungen und der Aufrechterhaltung der Vollständigkeit und Verfolgbarkeit der Konfigurationen während des gesamten SW-Lebenszyklus.“[FBM+1998, S. 24]
- Qualitätsmanagement: Dieser Tätigkeitsbereich umfasst die Planung, Steuerung und Kontrolle der Qualität eines Softwaresystems (Produkt) sowie des Prozesses seiner Erstellung.
- Systementwicklung: Dieser Tätigkeitsbereich umfasst die unmittelbare Erstellung des Softwaresystems. Dagegen nehmen die anderen drei Tätigkeitsbereiche nur eine mittelbare Rolle bei der Systementwicklung ein.

3.7 Prozessarchitektur

Einen groben Überblick über die Anordnung und den Ablauf der Phasen und der zugehörigen Aktivitäten wird Prozessarchitektur genannt.[NoSc1999, S. 171] Traditionelle Modelle wie das Wasserfall-Modell beruhen bspw. auf einem sequentiellen Modell, in dem alle Phasen linear durchlaufen werden. Neuere Modelle besitzen i. d. R. eine ausgefeiltere Ablaufreihenfolge.

3.8 Prozesssteuerung

Die Prozesssteuerung beschreibt, welches Prinzip dem Vorgehensmodell zur Definition der Ablaufreihenfolgen zugrunde liegt.[NoSc1999, S. 175] Hierbei werden drei verschiedene Prinzipien unterschieden:

- Aktivitätsorientierte Vorgehensmodelle: Diese Modelle beschreiben, welche Schritte in welcher Reihenfolge durchzuführen sind, um bestimmte Aktivitäten bzw. Phasen erfolgreich zu durchlaufen.

- Ergebnisorientierte Vorgehensmodelle: Werden die zu erstellenden Ergebnisse, deren Status und ihre Transformation in den Vordergrund gestellt, handelt es sich um ergebnisorientierte Vorgehensmodelle.
- Entscheidungsorientierte Vorgehensmodelle: Bei dieser Prozesssteuerung werden bestimmte Bedingungen definiert, die beschreiben, wann welche Aktivitäten durchzuführen sind. Damit werden bestimmte Entwicklungsmuster situationsbedingt festgehalten und beschrieben.

Es sei darauf hingewiesen, dass die beschriebenen Prinzipien sich nicht gegenseitig ausschließen, sondern komplementär verwendet werden können. Konkrete Vorgehensmodelle verwenden i. d. R. mehrere Prinzipien. Dabei steht meist ein Prinzip im Vordergrund bei der Beschreibung des Vorgehensmodells.

3.9 Rollenabdeckung

Unter diesem Aspekt werden die im Vorgehensmodell definierten Rollen näher beschrieben. Dabei wird ein besonderer Fokus auf diejenigen Rollen gelegt, die im Kontext der komponentenorientierten Entwicklung eine besondere Bedeutung haben.

3.10 Adaption

Vorgehensmodelle streben nach einem bestimmten Grad an Allgemeingültigkeit, um unabhängig von bestimmten Anwendungsbedingungen einsetzbar zu sein. Auch wenn Vorgehensmodelle eine Standardisierung der Prozesse der Systementwicklung anstreben, ist es meist unumgänglich, bei der Einführung eines Vorgehensmodells bestimmte Anpassungen an diesem vorzunehmen.[NoSc1999, S. 177] Die Möglichkeiten, die ein Vorgehensmodell dazu erlaubt, werden unter dem Aspekt Adaption näher beschrieben.

4 Gegenüberstellung und Vergleich

4.1 Terminologie

In Bild 4 werden die in den ausgewählten Vorgehensmodellen verwendeten Begriffe der hier eingeführten Terminologie gegenübergestellt, um so ein besseres Verständnis und eine höhere Transparenz über die verwendete Sprache zu erhalten.

	Catalysis	Perspective	RUP	V-Modell
Phase	-	Stage	Phase	Phase
Aktivität	Activity	Task	Workflow, Activity, Step	Aktivität, Teilaktivität
Ergebnis	Deliverable	Deliverables, Inputs, Outputs	Artifact	Produkt, Teilprodukt
Rolle	-	Team Role	Worker	Rolle
Technik	Technique	Technique	-	Elementar-methode
Richtlinie	-	Practical Guidelines	Work Guidelines	Handbuchsamm-lung
Notation	Notation	Notation	Language	-

Bild 4: Terminologische Einordnung

4.2 Klassifizierung

Eine Klassifizierung der ausgewählten Vorgehensmodelle erfolgt in Bild 5.

	Catalysis	Perspective	RUP	V-Modell
Art	revolutionär	revolutionär	evolutionär	evolutionär
Komponenten-bildung	Top-Down und Bottom-Up	Bottom-Up	Top-Down	Top-Down
System-granularität	Anwendung	Komponente und Anwendung	Anwendung	Anwendung

Bild 5: Klassifizierung der Vorgehensmodelle

4.3 Komponentenbegriff

In Bild 6 werden die von den untersuchten Vorgehensmodellen eingeführten Komponenten-begriffe gegenübergestellt. Hierbei wurde differenziert, ob Merkmale explizit angeführt oder implizit unterstellt worden sind. Falls keine Aussagen möglich waren, wurde dies ebenso vermerkt. Ergänzend sei darauf hingewiesen, dass bei der Auswertung der Literatur es ebenso vorgesehen war, dass bestimmte Merkmale von den Autoren der Vorgehensmodelle explizit ausgeschlossen werden. Von dieser prinzipiellen Möglichkeit machte allerdings keiner der Autoren Gebrauch.

	Catalysis	Perspective	RUP	V-Modell
Bezeichnung	Component	Component	Component	Fertigprodukt
Software-Artefakte	(+)	(+)	(+)	(+)
Wiederverwendbarkeit	(+)	+	+	+
Abgeschlossenheit	+	+	(+)	(+)
Vermarktbarkeit	?	?	?	(+)
Dienstangebote	+	+	+	?
Wohldefinierte Schnittstelle	+	+	+	?
Verbirgt ihre Realisierung	(+)	+	?	?
Kombinierbarkeit	(+)	(+)	?	?

Legende:

+ Merkmal explizit angeführt
(+) Merkmal implizit unterstellt
? keine Aussage möglich

Bild 6: Gegenüberstellung des Komponentenbegriffs

Das V-Modell unterschiedet hinsichtlich der Herkunft der Fertigprodukte in kommerzielle Fertigprodukte, die auf dem freien Markt angeboten werden, und Fertigprodukten, die innerhalb der eigenen Organisation vorliegen. Das V-Modell weist darauf hin, dass Fertigprodukte auf allen Ebenen der Erzeugnisstruktur des V-Modells, also nicht nur auf Quellcode-Ebene, sondern bspw. auch auf der Architektur- oder Anforderungs-Ebene verwendet werden können. Dieser weite Komponentenbegriff wird insofern von den Autoren wieder relativiert, indem bei der Entwicklungsprozessbeschreibung stets davon ausgegangen wird, dass ausschließlich Fertigprodukte für die Realisierung eingekauft werden.[o.V.1997a, Teil 3: Handbuchsammlung - Szenarien]

Ferner soll auf zwei weitere sprachliche Schwierigkeiten hingewiesen werden. Erstens verwendet der RUP das Wort „component“ sowohl in einer allgemeinen Bedeutung für bspw. Datenbank-Tabellen, Quell-Code-Dateien u. ä. Darüber hinaus wird das Wort ebenso in dem hier unterstellten Komponentenbegriff verwendet (vgl. Definition „component“ im Glossar des Vorgehensmodells). Zweitens finden sich auch bei Catalysis sprachliche Unschärfen. Catalysis führt zunächst einen Komponentenbegriff im Allgemeinen und im Besonderen ein.[DSWi1998, S. 386f.] Davon abweichend wird indes ebenso ein unreflektierter Kompo-

nentenbegriff verwendet, der bspw. auch organisatorische Einheiten umfasst.[DSWi1998, S. 414]

4.4 Abdeckung des Lebenszyklus einer Komponente

4.4.1 Überblick

Bild 7 gibt einen Überblick darüber, welche Phasen im Lebenszyklus einer Komponente von den betrachteten Vorgehensmodellen (gut) unterstützt werden. Dabei wird jeweils der Name der Phase bzw. Aktivität angegeben, wie er von dem entsprechenden Vorgehensmodell verwendet wird. In den folgenden Unterabschnitten werden verschiedene Aspekte aufgegriffen, die explizit bei einer komponentenorientierten Bedeutung von Interesse sind.

	Catalysis	**Perspective**	**RUP**	**V-Modell**
Standardisierung	How to Specify a Component	-	-	-
Entwicklung	How to Implement a Component	Component Process	Process Workflows	Systemerstellung
Wiederauffindung	-	(Search for possible areas of reuse)	(Project Management)	Realisierbarkeitsuntersuchung, Marktsichtung
Technische Anpassung	Reuse and Pluggable Design Frameworks in Code	-	-	(Systemkonfektion)
Fachliche Anpassung	-	-	-	(Systemkonfektion)
Komposition	Components and Connectors	(Roll Out)	Deployment	Integration
Betrieb	-	-	-	-
Deinstallation	-	-	-	-

Legende:

(...) Phase im Lebenszyklus einer Komponente wird zwar genannt, inhaltliche Ausgestaltung bleibt (teilweise) unklar

\- Phase im Lebenszyklus einer Komponente wird nicht abgedeckt

Bild 7: Abdeckung des Lebenszyklus einer Komponente

4.4.2 Catalysis

Catalysis beschreibt Aktivitäten zur Spezifikation von Komponenten, die von diesem Vorgehensmodell als eine Voraussetzung zur erfolgreichen Standardisierung von Komponenten angesehen werden. Hierbei wird insbesondere auf eine formale Spezifikation Wert gelegt. Komponenten sollen ohne weitere Änderungen, sondern nur durch Anpassung mit Parametri-

sierung an eine Systemumgebung angepasst werden können. Um die Komposition zu ermöglichen, wird eine sogenannte Kit-Architektur vorgeschlagen, die standardisierte Techniken bereithält. Ebenso werden Techniken zur Verknüpfung heterogener Komponenten dargestellt. Um die verschiedenen Techniken nutzen zu können, werden entsprechende Muster angegeben.[DSWi1998]

4.4.3 Perspective

Von Perspective wird ein expliziter Entwicklungsprozess für die Entwicklung einer Komponente vorgestellt. Aktivitäten, welche die anderen Phasen des Lebenszyklus einer Komponente betreffen, werden zwar genannt. Die inhaltliche Ausgestaltung dieser Phasen bleibt allerdings unklar.

4.4.4 RUP

RUP erläutert explizit für jede Phase und den jeweiligen Workflow die Auswirkungen bei einer komponentenorientierten Vorgehensweise. Dabei handelt es sich allerdings meist nicht um eine Darstellung konkreter Aktivitäten. Vielmehr werden entsprechende zusätzliche zu bedenkende Rahmenparameter beschrieben. Beispielsweise wird darauf hingewiesen, dass bei der Beschaffung von Komponenten bei Fremdanbietern zwar neue Risikofaktoren entstehen, gleichzeitig allerdings Entwicklungszeiten verkürzt werden können.

4.4.5 V-Modell

Bei dem V-Modell handelt es sich um ein Top-Down-basiertes Modell. Nachdem die Anforderungen an ein System beschrieben sind, wird im Rahmen einer Realisierbarkeitsuntersuchung eine Marktsichtung vorgenommen.[o.V.1997a, Teil 3: Handbuchsammlung – Szenarien, S. SA-13] Grundlage der Auswahl einer Komponente bildet ein auf Basis der Anforderungen erstellter Kriterienkatalog. Ergebnis der Realisierbarkeitsuntersuchung kann sein, dass die Anforderungen von einer am Markt erhältlichen Komponente gedeckt werden. Ist dies nicht der Fall, so wird ein sogenanntes Forderungscontrolling eingeleitet, indem überprüft wird, ob die definierten Anforderungen ohne Projektgefährdung in der Art angepasst werden können, dass evtl. zum Einsatz kommende Komponenten doch noch verwendet werden können. Der Einsatz von Komponenten erfordert nach ihrer Beschaffung sogenannte Konfektionierungsarbeiten, um die verwendeten Komponenten so anzupassen, dass sie den definierten Anforderungen genügen. Darüber hinaus werden ebenso notwendige Integrationsaktivitäten definiert, um die vorhandenen Komponenten lauffähig zu gestalten. Die inhaltliche Ausgestaltung der Konfektionierungs- sowie Integrations-Aktivitäten werden inhaltlich nicht weiter erläutert. Die Beschaffung der Komponente obliegt dem Projektmanagement. Dieses stößt evtl. das der Softwareentwicklung unterliegende Forderungscontrolling an. Eine explizite Durchführung von Prüfungsaktivitäten bei der Beschaffung von Komponenten wird im Rahmen des Qualitätsmanagements nicht als erforderlich erachtet. Vielmehr obliegt es dem Projektmanagement, die Qualität der eingekauften Komponenten zu gewährleisten. Anders dagegen bei den durchzuführenden Konfektionierungsaktivitäten. Hier werden diese durch Maßnahmen des Qualitätsmanagements überwacht. Das V-Modell schreibt vor, dass fremd- und eigenentwickelte Komponenten demselben Konfigurationsmanagement zu unterliegen haben. Dabei wird insbesondere darauf hingewiesen, dass die Weiterentwicklung der Komponenten nicht von neuen Anforderungen an das System ausgehen, sondern von dem Anbieter der

Komponente. Die Weiterentwicklung der Komponente ist demnach hauptsächlich vom Markt getrieben und nicht von den Anwenderanforderungen.

4.5 Abdeckung der Tätigkeitsbereiche

Bild 8 beschreibt, welche Tätigkeitsbereiche von den untersuchten Vorgehensmodellen aufgegriffen werden.

	Catalysis	**Perspective**	**RUP**	**V-Modell**
Projekt-management	-	-	Project Management	Projekt-management
Qualitäts-management	-	-	-	Qualitätssicherung
Konfigurations-anagement	-	-	Configuration & Chanage Mgmt	Konfigurations-management
Entwicklung	Catalysis Process	Perspective Process	Core Process Workflows	Systemerstellung

Legende:

- Tätigkeitsbereich wird nicht abgedeckt

Bild 8: Abdeckung der Tätigkeitsbereiche

4.6 Prozessarchitektur

4.6.1 Catalysis

Das Vorgehensmodell Catalysis nimmt eine Sonderstellung unter den Vorgehensmodellen ein, da das Modell keine Vorschriften bezüglich zu durchlaufender Phasen und Aktivitäten kennt. Statt dessen definiert das Vorgehensmodell eine Reihe von sogenannten Prozess-Mustern (Process Patterns), die beschreiben, unter welchen Voraussetzungen und Randbedingungen bestimmte Aktivitäten durchzuführen sind. Die Prozess-Muster sind in folgende Kategorien gegliedert:

- Main Process Patterns (4 Patterns)
- (Business) Modeling Patterns (13 Patterns)
- Patterns for Specifying Components (13 Patterns)
- Patterns for Designing to a Meet a Specification (13 Patterns)
- Detailed Design Patterns (5 Patterns).

Ein Beispiel für ein Main Process Pattern ist das Pattern Short-Cycle Development. Dieses Pattern beschreibt, dass kurze Entwicklungszyklen gewählt werden sollen, um Projekte unter unsicheren Bedingungen durchführen zu können. Andere Muster wie bspw. Object Development from Scratch übernehmen primär die Funktion andere Muster zu bündeln. Ferner existie-

ren ebenso Mustern, die für den Entwurf von Softwaressystemen geeignet sind. Beispielsweise sind die Detailed Design Patterns mit den Mustern in [GHJV1995] vergleichbar.

Nichtsdestotrotz weisen die Autoren darauf hin, dass der Ablauf sämtlicher Aktivitäten bei Catalysis einem Grundgerüst folgt. Dieses besitzt eine Form aus Modellierung, Entwurf, Implementierung und Test.[DSWi1998] Die Aktivitäten sind allerdings nicht umfassend definiert und erläutert, sondern haben nur eine allgemeine Bedeutung und können sich auf unterschiedliche Bezugsobjekte beziehen. Dabei werden die Bezugsobjekte allerdings nicht festgeschrieben. Beispielsweise nennen die Autoren die Bezugsobjekte: Business (Domain) Model, System Context, System Specification, Component Models, System Architecture, System Detailed Design.[DSWi1998, S. 528-530] Die exemplarisch genannten Bezugsobjekte können allerdings variieren und werden im Projektkontext jeweils neu festgelegt.

Aufgrund der großen Freiheitsgrade im Entwicklungsprozess charakterisieren die Autoren ihn als nicht-linear, iterativ und parallel. Nicht-linear bedeutet, dass alternative Pfade im Entwicklungsprozess möglich sind. Iterativ bedeutet, dass verschiedene Aktivitäten wiederholt durchgeführt werden. Darüber hinaus ist es möglich, dass bestimmte Aktivitäten parallel durchgeführt werden können. [DSWi1998, S. 512-514]

Aufgrund der genannten Eigenschaften ist es nicht möglich, eine allgemeine Prozessarchitektur zu beschreiben. Eine exemplarische Darstellung eines Entwicklungsprozesses wird in [DSWi1998, S. 522-526] gegeben. Diese soll hier aus Gründen des beschränken Umfanges der Untersuchung nicht rekapituliert werden.

4.6.2 Perspective

Auf einer abstrakten Ebene besteht das Vorgehensmodell Perspective aus zwei Prozessen: Ein sogenannter Solution Process erstellt ausgehend von spezifischen Anforderungen ein unternehmensindividuelles Softwaresystem. Dagegen werden im Component Process auf Basis allgemeingültiger Anforderungen spezifischer Domänen einzelne Komponenten entwickelt. Beide Prozesse werden in unterschiedliche Phasen aufgeteilt, die jeweils nicht strikt sequentiell, sondern iterativ und inkrementell durchlaufen werden können (vgl. Bild 9).

Der Solution Process besteht aus sieben Phasen:[AlFr1998, S. 271-301] Im Rahmen der Phase Feasibility wird eine Durchführbarkeitsstudie erstellt. Die Anforderungen an ein einzelnes Systeminkrement werden im Rahmen der Phase Analysis erhoben, die dabei durch gezieltes Entwickeln von benutzungsfähigen Prototypen unterstützt wird (Phase Prototype). Innerhalb der Phase Plan Increment wird ein Plan erstellt, der beschreibt, welche Systeminkremente, welche Funktionalitäten erfüllen sollen und in welcher Reihenfolge diese ausgeliefert werden. Der Entwurf und die Implementierung eines Inkrements erfolgen in der Phase Design and Build. Diese Phase wird überlagert von der Phase User Acceptance, um die Erfüllung der Benutzer-Anforderungen sicherzustellen. Abgeschlossen wird die Entwicklung mit der Einführung des Systems in der Phase Roll Out.

Der Component Process besteht aus acht Phasen:[AlFr1998, S. 303-327] Ein allgemeiner Überblick über die Anforderungen an eine Komponente werden innerhalb der Phase Architectural Scoping erstellt. Innerhalb der Phase Assessment wird überprüft, welche Nachfrage nach bestimmten Diensten von Projekten im Solution Process bestehen. Die Phase Plan Services erstellt einen Plan aus dem hervorgeht, welche Dienste in welchen Komponenteninkrementen bereitgestellt werden. Die Phasen Design and Build, Acceptance sowie Roll Out sind analog

zu den entsprechenden Phasen im Solution Process, beziehen sich allerdings hier auf die Entwicklung einer Komponente. Zwei zusätzliche Phasen werden durchlaufen, wenn im Rahmen des Assessment festgestellt wird, eine neue Komponente zu entwickeln und nicht eine vorhandene Komponente zu erweitern. Dann werden ebenso die Phasen Feasibility und Analysis abgearbeitet.

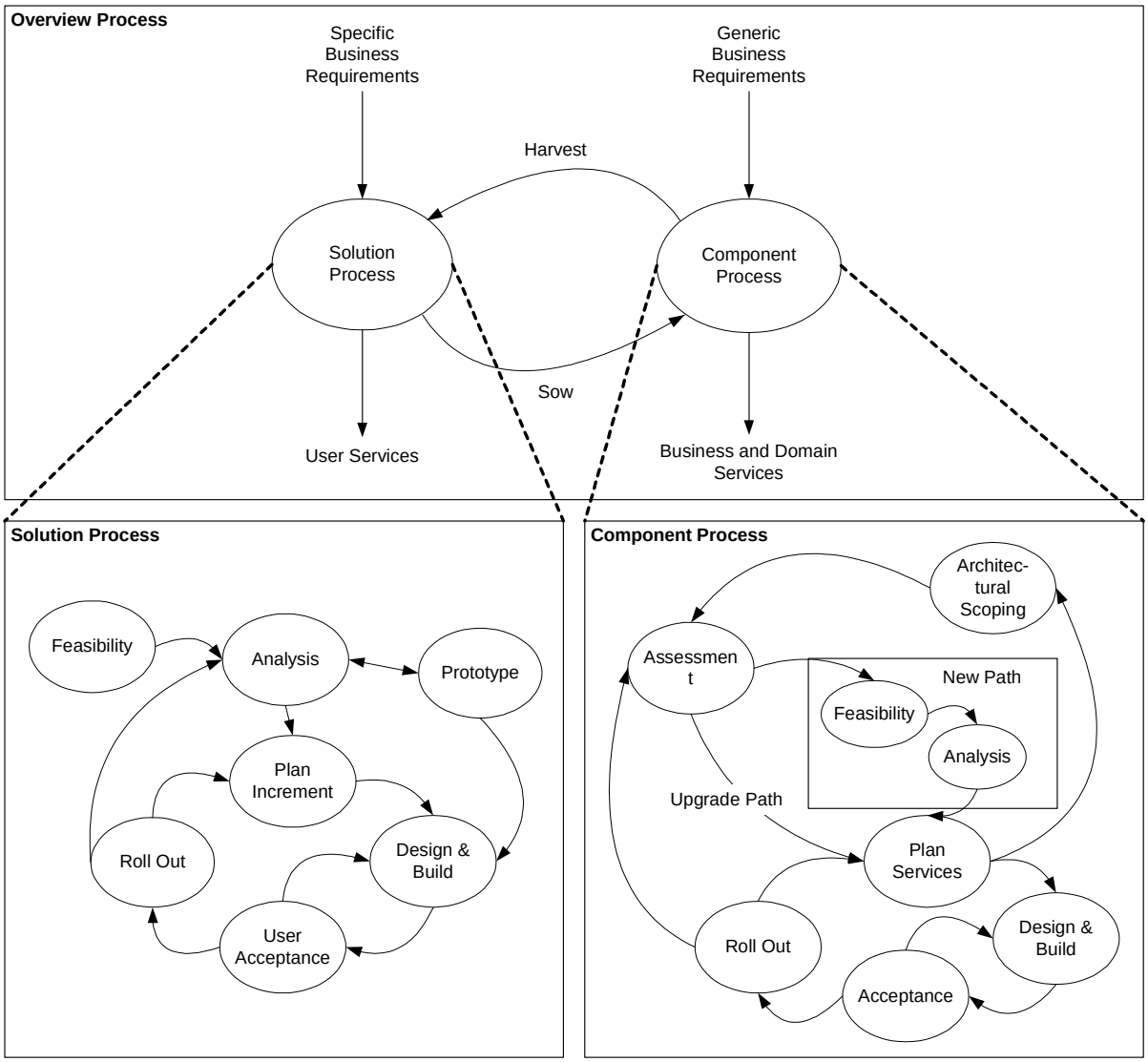

Bild 9: Prozessarchitektur von Perspective (in Anlehnung an: [AlFr1998, S. 264, 275, 309])

4.6.3 RUP

Ein Charakteristikum des RUP ist die konzeptionelle Trennung des Entwicklungsprozesses in Phasen und Arbeitsabläufe (Workflows). Die Trennung in Phasen spiegelt dabei eine zeitliche Unterscheidung wider. Die Trennung in Arbeitsabläufe orientiert sich an sachlichen Kriterien. Es werden vier Phasen unterschieden: In der Phase Inception wird eine Kernidee für ein Softwaresystem entwickelt und eine Grundvorstellung über die Leistungsfähigkeiten des Produktes hervorgebracht. Im Rahmen der Phase Elaboration wird die Architektur des Systems defi-

niert. Die Implementierung der Architektur geschieht im Rahmen der Phase Construction. Die Phase Transition umfasst die Auslieferung des Softwaresystems bei einem Anwender.

Innerhalb jeder Phase können mehrere Iterationen eingeführt werden, um eine inkrementelle Systementwicklung zu ermöglichen. Innerhalb jeder Iteration werden prinzipiell sämtliche Arbeitsabläufe (Business Modeling, Requirements, Analysis & Design, Implementation, Test, Deployment, Configuration Management, Project Management und Environment) durchgeführt. Dabei verschiebt sich allerdings der Schwerpunkt der Arbeitsabläufe: In frühen Iterationen liegt der Schwerpunkt eher bei konzeptionellen Tätigkeiten, während sich dieser zu späteren Projektphasen eher bei implementierungstechnischen Tätigkeiten befindet. Vgl. hierzu die entsprechenden Kurven in Bild 10, die als exemplarische Belastungsprofile in einem typischen Projekt interpretiert werden können. Jeder Arbeitsablauf wird durch ein UML Activity Diagramm weiter verfeinert. Diese Darstellung ist allerdings nicht explizit abhängig von der entsprechenden Phase, sondern für alle Phasen identisch. Eine weitere Differenzierung findet sich erst wieder in den verfeinerten, rein textuellen Darstellungen.

Prinzipiell kann der RUP als eine Variante des Spiralmodells nach Boehm [Somm1996, S. 13-16] verstanden werden, die inhaltlich detaillierter ausformuliert wurde und vornehmlich optisch anders dargestellt ist.

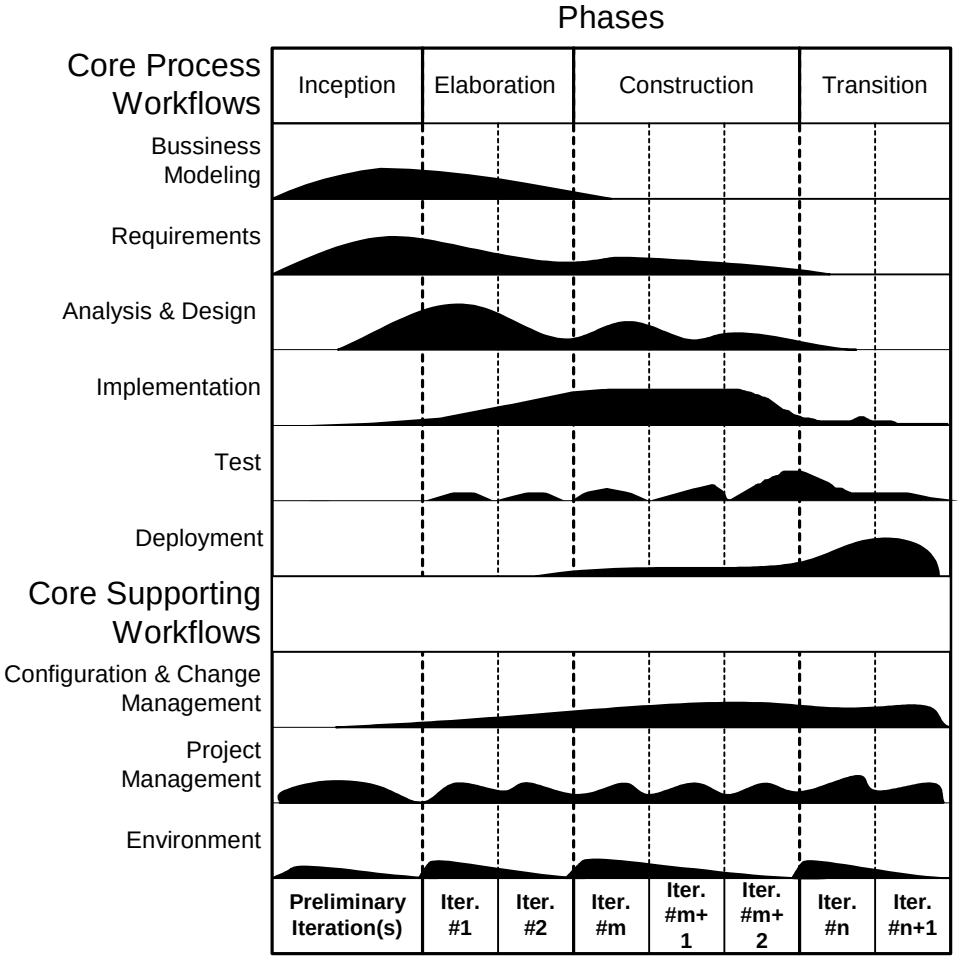

Bild 10: Prozessarchitektur von RUP (Quelle: [Rati2001])

4.6.4 V-Modell

Die gesamten Aktivitäten des V-Modell sind in vier Submodelle untergliedert:

- Das Projektmanagement (PM) plant, steuert, kontrolliert und informiert die drei Submodelle Systemerstellung (SE), Qualitätssicherung (QS) und Konfigurationsmanagement (KM).
- Das KM verwaltet die Produkte, die im Rahmen der übrigen Aktivitäten erstellt werden. Ein Produkt wird als das Ergebnis einer Aktivität bzw. als Gegenstand einer Aktivität des V-Modells verstanden. Dieses Submodell stellt sicher, dass alle Produkte eindeutig identifizierbar, dass verschiedene Versionen eines Produktes unterscheidbar und dass Änderungen an Produkten kontrolliert durchführbar sind.
- Die QS erarbeitet einerseits entsprechende Qualitätskriterien, Prüfpläne, Prüfkriterien und Standards. Andererseits werden die in den übrigen Modellen entwickelten Produkte den entsprechenden Prüfungen unterzogen.
- Die SE umfasst alle Aktivitäten, die unmittelbar dem Prozess der Softwareentwicklung zuzurechnen sind: System-Anforderungsanalyse, System-Entwurf, Grobentwurf, Feinentwurf, Implementierung, Integration, System-Integration und Überleitung in die Nutzung. Diese Aktivitäten können als Phasen verstanden werden und werden graphisch in Form des Buchstabens V angeordnet. Dabei werden die Aktivitäten auf der linken Seite der Anordnung der Entwicklung von Systemteilen, die auf der rechten Seite der Integration der Systemteile zugeordnet.

Trotz der Definition neuer Konzepte und verschiedener Entwicklungsszenarien ist festzuhalten, dass das V-Modell im Kern als ein um die Qualitätssicherung erweitertes Wasserfall-Modell mit Rückkopplung verstanden werden kann.[Balz1998, S. 101.] Das Zusammenspiel der verschiedenen Modelle wird in Bild 11 dargestellt.

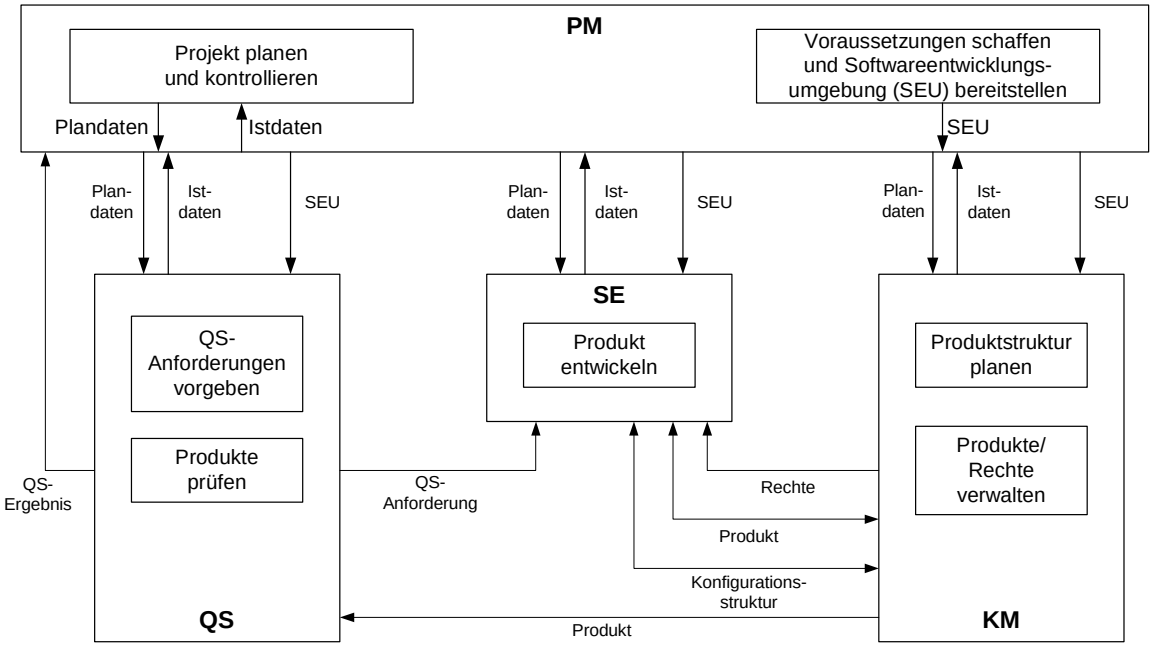

Bild 11: Prozessarchitektur vom V-Modell (Quelle: [o.V.1997b])

4.7 Prozesssteuerung

4.7.1 Catalysis

Catalysis beschreibt keine geschlossene Menge von durchzuführenden Aktivitäten oder zu erstellenden Ergebnissen. Vielmehr wird der Entwicklungsprozess durch eine Menge von Prozess-Mustern beschrieben, aus denen hervorgeht, in welcher Situation bzw. unter welchen Bedingungen bestimmte Arbeitsschritte durchzuführen sind. Es handelt sich demnach um eine entscheidungsorientierte Prozesssteuerung.

4.7.2 Perspective

Perspective basiert auf einer aktivitätsorientierten Prozesssteuerung. Jede der Phasen wird durch Angabe von durchzuführenden Aktivitäten und zugehörigen Unteraktivitäten beschrieben. Hierbei werden auf der Ebene der Aktivitäten ebenso die notwendigen Ergebnisse zur Bearbeitung der Aktivität sowie die zu erzeugenden Ergebnisse formuliert. Ergänzend enthält das Vorgehensmodell auch Aspekte einer ergebnisorientierten Prozesssteuerung. Beispielsweise wird in [AlFr1998, S. 278f., 311] beschrieben, welche Ergebnisse zu welchem Zeitpunkt in welchem Zustand vorliegen sollten. Diese nehmen allerdings im Vergleich zur aktivitätsorientierten Beschreibung nur eine untergeordnete Rolle ein.(anderer Meinung [NoSc1999, S. 177])

4.7.3 RUP

RUP ist primär aktivitätsorientiert definiert. Zu jedem Arbeitsablauf werden detaillierte Darstellungen vorgenommen, die eine schrittweise Abarbeitung ermöglichen. Ergänzend werden im Modell auch Aussagen über die benötigten und erzeugten Ergebnisse zusammengestellt. Dabei ist teilweise ersichtlich, welche Ergebnisse welche Zustände im Entwicklungsprozess zu durchlaufen haben.

4.7.4 V-Modell

Das V-Modell basiert vorwiegend auf einer aktivitätsorientierten Prozesssteuerung. Zu jedem der vier Submodelle werden jeweils die Hauptaktivitäten und deren Beziehungen bzw. Ablaufreihenfolgen genannt. Diese werden in weiterführenden Beschreibungen in Unteraktivitäten gegliedert. Darüber hinaus wird zu jeder durchzuführenden Aktivität beschrieben, welche Ergebnisse sich in welchen Zuständen befinden, wann die Aktivität gestartet werden kann bzw. durchgeführt worden ist. Mit anderen Worten wird ebenso z. T. eine ergebnisorientierte Prozesssteuerung unterstellt.

Bild 12 beschreibt zusammenfassend die verschiedenen Prinzipien der Prozesssteuerung der betrachteten Vorgehensmodelle.

	Catalysis	**Perspective**	**RUP**	**V-Modell**
Prozesssteuerung	entscheidungs-orientiert	aktivitäts-orientiert	aktivitäts-orientiert	aktivitäts-orientiert

Bild 12: Gegenüberstellung der Prozesssteuerung

4.8 Rollenabdeckung

4.8.1 Catalysis

In Catalysis werden keine Rollen definiert.

4.8.2 Perspective

Perspective legt hohen Wert darauf, dass die Entwicklungsarbeit in Teams durchgeführt wird, die jeweils aus nicht mehr als sechs Personen bestehen.[AlFr1998, S. 329-344] Innerhalb dieser Teams soll eine ausgewogenes Verhältnis zwischen fachlich und technisch ausgerichteten Rollen existieren. Prinzipiell wird zwischen Rollen im Solution- und im Component-Process sowie Rollen für die technische Infrastruktur unterschieden. Dabei werden insgesamt 21 Rollen definiert. Im Hinblick auf die komponentenorientierte Entwicklung sind folgende Rollen herauszuheben:

- Reuse Identifier: Aufgabe dieser Rolle ist es, im Rahmen des Solution Process Möglichkeiten zur Gestaltung wiederverwendbarer Dienste zu identifizieren sowie die Wiederverwendung existierender Dienste sicherzustellen.
- Reuse Librarian: Diese Rolle ist verantwortlich für den Betrieb eines Komponenten-Repositoriums, indem Komponenten archiviert und recherchiert werden.
- Reuse Assessor: Im Rahmen dieser Rolle werden neue Möglichkeiten der Wiederverwendung identifiziert, bewertet und vorhandene Möglichkeiten werden in die verschiedene Solution Process Projekte getragen.
- Reuse Architect: Diese Rolle entwickelt eine Gesamtvision, wie wiederverwendbare Komponenten zu entwickeln sind und wie diese sich in eine Gesamtarchitektur einfügen.

4.8.3 RUP

Insgesamt werden im RUP 30 Rollen definiert, die in fünf Gruppen zusammengestellt werden: Analysts, Developer, Testing Professionals, Manager und Other Workers. Dabei werden im Vorgehensmodell keine im Hinblick auf eine komponentenorientierte Entwicklung besonderen Rollen eingeführt. Rollen, die einen vermutlichen engen Bezug zur Komponentenerstellung haben, sind Architect, Designer, Implementer und Integrator. Bei der Beschreibung der Fähigkeiten und Kompetenzen dieser Rollen zeigt sich allerdings, dass zwar der Begriff Component verwendet wird. Dort allerdings in einer unspezifischen Bedeutung gebraucht wird. Daher wird die Beschreibung der eingeführten Rollen hier nicht weiter vertieft.

4.8.4 V-Modell

Insgesamt werden im V-Modell 23 Rollen definiert.[o.V.1997a, Teil 3: Handbuchsammlung – Rollenkonzept, S. A-1] Spezifische Rollen für die komponentenorientierte Entwicklung werden nicht definiert. Daher wird an dieser Stelle davon abgesehen, dass Rollenverständnis im V-Modell weiter zu vertiefen.

4.9 Adaption

4.9.1 Catalysis

Catalysis kann flexibel durch Einsatz und Anpassung von Prozess-Mustern (Process Patterns) gestaltet werden, ohne dass eine feste Aktivitätenreihenfolge vorgeschrieben wird. Es werden Erscheinungsformen des Prozesses für typische Projekttypen wie die Entwicklung ohne Wiederverwendung (Build Route) oder die Komposition von Komponenten (Assembly Route) dargestellt.[DSWi1998]. Es muss allerdings kritisch eingeräumt werden, dass diese Darstellungen eher exemplarischen Charakter haben. Damit geben sie nur verhältnismäßig wenig systematische Unterstützung.

4.9.2 Perspective

Im Vorgehensmodell Perspective wird darauf hingewiesen, dass das Modell an spezifische Belange angepasst werden kann und auch sollte.[AlFr1998, S. 260f.] Konkrete Aktivitäten werden allerdings nicht beschrieben.

4.9.3 RUP

Um den RUP an unternehmensspezifische Besonderheiten anzupassen, werden im Rahmen des Arbeitsablaufes Environment verschiedene Aktivitäten definiert, die beschreiben, wie eine Einführung des Prozesses in eine Organisation abgewickelt werden kann. Die Anpassungsmaßnahmen beruhen auf verschiedenen Assessment-Aktivitäten, die ermitteln, welche Kompetenzen, Schwachpunkte etc. eine Organisation bei der Softwareentwicklung besitzt. Ergänzend werden darüber hinaus konkrete sogenannte Roadmaps definiert, die eine bestimmte Adaption des Prozesses im Kontext eines speziellen Entwicklungsumfeldes beschreiben. Es werden neben der Adaption für eine komponentenorientierte Entwicklung ebenso Adaptionen für die Entwicklung von E-Business-Anwendungen sowie für die Entwicklung kleiner Projekte angeboten.

4.9.4 V-Modell

Das V-Modell gibt umfassende Hilfestellung für eine Adaption der Vorgehensweise. Zum einen beschreibt das Vorgehensmodell verschiedene Entwicklungsszenarien, die einen prinzipiellen Ablauf der Entwicklung demonstrieren (bspw. „Inkrementelle Entwicklung“, „Grand Design“, „Objektorientierte Entwicklung“, „Einsatz von Fertigprodukten“). Diese Szenarien geben einen Einblick in den Ablauf der Aktivitäten in einem bestimmten Entwicklungsrahmen.[o.V.1997a, Teil 3: Handbuchsammlung - Szenarien] Darüber hinaus wird ein umfassender Leitfaden beschrieben, aus dem hervorgeht, unter welchen Umständen bzw. Randbedingungen bestimmte Aktivitäten des Modells (bedingt) entfallen können bzw. zwingend durchgeführt werden müssen. Zur Typisierung von Anforderungen werden Merkmale wie Projektgrößeneinstufung, Komplexitätseinstufung von Daten und Funktionen, Quantifizierung der Wartbarkeitsanforderungen u. a. genannt.[o.V.1997a, Teil 3: Handbuchsammlung – Tailoring und projektspezifisches V-Modell]

5 Resümee und Ausblick

Bei den untersuchten komponentenorientierten Vorgehensmodellen handelt es sich sowohl um revolutionäre (Catalysis, Perspective) als auch um evolutionäre Modelle (RUP, V-Modell). Während RUP und V-Modell ebenso Projekt-, Qualitäts- und Konfigurationsmanagement unterstützen, behandeln Catalysis und Perspective ausschließlich die Entwicklung eines Softwaresystems. Einschränkend ist allerdings darauf hinzuweisen, dass V-Modell und RUP keine komponentenspezifischen Aspekte des Projekt-, Qualitäts- und Konfigurationsmanagements aufweisen. Auf die Unterscheidung zwischen der Entwicklung einer einzelnen Komponente und eines Komponentensystems wird insbesondere von Perspective hingewiesen, das für beide Fälle verschiedene Vorgehensmodelle beschreibt. Obwohl die ausgewählten Vorgehensmodelle speziell auf eine komponentenorientierte Entwicklung ausgerichtet sind, zeigt sich, dass wesentliche Phasen im Lebenszyklus einer Komponente von den Vorgehensmodellen nur rudimentär behandelt werden. Eine Ausnahme bildet hier Catalysis, wo umfangreiche Aktivitäten zur Standardisierung, zur technischen Anpassung sowie zur Komposition von Komponenten beschrieben werden. Spezifische Rollen für eine komponentenorientierte Entwicklung wie bspw. ein Komponentenbibliothekar werden nur von Perspective eingeführt. Daher ist zu vermuten, das speziell die Wiederauffindung von Komponenten von den anderen Vorgehensmodellen nur unbefriedigend ermöglicht wird. Die Adaption des Vorgehensmodells wird vom RUP und V-Modell umfassend, von Catalysis und Perspective nur unzureichend dargelegt.

Es verbleibt die Frage, welches Vorgehensmodell ein Softwarehersteller für die komponentenorientierte Entwicklung verwenden soll. Um eine fundierte Entscheidungsempfehlung aussprechen zu können, sind weitere Untersuchungen notwendig, die insbesondere auf empirischen oder experimentellen Methoden fußen sollten. Abgesehen von diversen Parametern wie bspw. Unternehmensgröße, Erfahrungshintergrund der Organisation, Projektgröße, Anwendungsdomäne etc. sollten vertiefende Untersuchungen insbesondere drei strategische Handlungsoption eines Softwareherstellers berücksichtigen:

- Komponentenentwickler: Der Softwarehersteller konzentriert sich ausschließlich auf die Entwicklung einzelner Komponenten und fungiert als Zulieferer für Komponenten.
- Komponentenassemblierer: Der Softwarehersteller konzentriert sich auf die Wiederauffindung, technische und fachliche Anpassung sowie auf die Komposition von Komponenten zu vollständigen Softwaresystemen.
- Komponentengeneralist: Der Softwarehersteller entwickelt auf Basis eines komponentenorientierten Architektur-Paradigmas und beherrscht alle Phasen des Lebenszyklus einer Komponente. Dabei ist allerdings die Menge der zugekauften bzw. fremdentwickelten Komponenten verhältnismäßig gering.

Es wird hier die Hypothese formuliert, dass für die unterschiedlichen Handlungsoptionen jeweils speziell angepasste Vorgehensmodelle benötigt werden, welche die jeweiligen Schwerpunkte der Tätigkeitsbereiche optimal unterstützen.

Literatur

[o.V.1997a] Das V-Modell - Allgemeiner Umdruck Nr. 250: Vorgehensmodell - Planung und Durchführung von IT-Vorhaben - Entwicklungsstandard für IT-Systeme des Bundes. http://www.v-modell.iabg.de/, Abruf am 2002-02-28. 1997a.

[o.V.1997b] Das V-Modell - Entwicklungsstandard für IT-Systeme des Bundes - Kurzbeschreibung. http://www.v-modell.iabg.de/, Abruf am 2002-02-28. 1997b.

[o.V.2001] Crystal software development methodologies. http://www.crystalmethodologies.org/, Abruf am 2002-02-28.

[Acke+2002] *Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Klein, U.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T.; Turowski, K.*: Vereinheitlichte Spezifikation von Fachkomponenten - Memorandum des Arbeitskreises 5.10.3 Komponentenorientierte betriebliche Anwendungssysteme. http://wi2.wiso.uni-augsburg.de/gi-memorandum.php.htm, Abruf am: 2002-03-01. Augsburg 2002.

[AlFr1998] *Allen, P.; Frost, S.*: Component-Based Development for Enterprise Systems - Applying the Select Perspective. Cambridge University Press, Cambridge 1998.

[Balz1998] *Balzert, H.*: Lehrbuch der Software-Technik - Software-Management, Software-Qualitätssicherung, Unternehmensmodellierung. Heidelberg, Berlin 1998.

[Beck1999] *Beck, K.*: Embracing Change with Extreme Programming. In: IEEE Computer 32 (1999) 10, S. 70-77.

[Beck2000] *Beck, K.*: Extreme Programming Explained - Embrace Change. Addison-Wesley, Boston et al. 2000.

[BeFo2000] *Beck, K.; Fowler, M.*: Planning Extreme Programming. Boston et al. 2000.

[Brem1998] *Bremer, G.*: Genealogie von Entwicklungsschemata. In: *R. Kneuper; G. Müller-Luschnat; A. Oberweis (Hrsg.)*: Vorgehensmodelle für die betriebliche Anwendungsentwicklung. Teubner, Stuttgart, Leipzig 1998, S. 32-59.

[Burg+2000] *Burg, W. D.; Hawker, S.; Hale, D. P.; McInnis, K.; Parrish, A.; Sharpe, S.; Woolridge, R.*: Exploring a Comprehensive CBD Method - Use of CBD/e in Practice. Third International Workshop on Component-Based Software Engineering. 2000.

[DSWi1998] *D'Souza, D. F.; Wills, A. C.*: Objects, Components, and Frameworks with UML - The Catalysis Approach. Addison-Wesley, Reading, MA, et al. 1998.

[FeLo2000] *Fettke, P.; Loos, P.*: Komponentendokumentationen - Eine systematische Bewertung von Ordnungssystemen aus formaler Sicht. In: *K. Turowski (Hrsg.)*: Modellierung und Spezifikation von Fachkomponenten: Workshop im Rahmen der MobIS 2000 Modellierung betrieblicher Informationssysteme, Siegen, Deutschland, 12. Oktober 2000, Tagungsband. Siegen 2000, S. 51-70.

[FeLo2001] *Fettke, P.; Loos, P.* Fachkonzeptionelle Standardisierung von Fachkomponenten mit Ordnungssystemen - Ein Beitrag zur Lösung der Problematik der Wiederauffindbarkeit von Fachkomponenten. Working Papers of the Research Group Information Systems & Management, Paper 3. Chemnitz 2001.

[FBM+1998] *Fischer, T.; Biskup, H.; Müller-Luschnat, G.*: Begriffliche Grundlagen für Vorgehensmodelle. In: *R. Kneuper; G. Müller-Luschnat; A. Oberweis (Hrsg.)*: Vorgehensmodelle für die betriebliche Anwendungsentwicklung. Teubner, Stuttgart, Leipzig 1998, S. 13-31.

[GHJV1995] *Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.*: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, et al. 1995.

[Grif1998] *Griffel, F.*: Componentware - Konzepte und Techniken eines Softwareparadigmas. Heidelberg 1998.

[HeCo2001] *Heineman, G. T.; Councill, W. T.*: Component-Based Software Engineering. Addison-Wesley, Boston et al. 2001.

[Hump1995] *Humphrey, W. S.*: A Discipline for Software Engineering. Addison-Wesley, Reading, MA, et al. 1995.

[JBR1998] *Jacobson, I.; Booch, G.; Rumbaugh, J.*: The Unified Software Development Process. Addison-Wesley, Reading, MA, et al. 1998.

[JCJÖ1992] *Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.*: Object-Oriented Software Engineering. Addison-Wesley, Reading, MA, et al. 1992.

[McIn1999] *McInnis, K.* An Overview of CBD/e. http://www.cbd-hq.com/articles/1999/991115km_overviewcbde.asp, Abruf am 2002-02-28.

[NoSc1999] *Noack, J.; Schienmann, B.*: Objektorientierte Vorgehensmodelle im Vergleich. In: Informatik-Spektrum 22 (1999), S. 166-180.

[Rati2001] *Rational Software Corporation* Rational Unified Process. http://www.rational.com/tryit/rup/index.jsp, Abruf am 2002-03-01.

[Same1997] *Sametinger, J.*: Software Engineering with Reusable Components. Berlin et al. 1997.

[STR2000] *Sandmann, C.; Teschke, T.; Ritter, J.*: Ein Vorgehensmodell für die komponentenbasierte Anwendungsentwicklung. In: *B. Britzelmaier; S. Geberl (Hrsg.)*: Information als Erfolgsfaktor. Teubner Verlag, 2000.

[Schr2001] *Schryen, G.*: Komponentenorientierte Softwareentwicklung in Softwareunternehmen: Konzeption eines Vorgehenmodells zur Einführung und Etablierung. Deutscher Universitäts-Verlag, Wiesbaden 2001.

[Somm1996] *Sommerville, I.*: Software Engineering. 5. Aufl., Addison-Wesley, Harlow et al. 1996.

[Szyp1999] *Szyperski, C.*: Component Software - Beyond Object-Oriented Programming. Harlow, England, et al. 1999.

[Turo1999] *Turowski, K.*: Ordnungsrahmen für komponentenbasierte betriebliche Anwendungssysteme. In: *K. Turowski (Hrsg.)*: Tagungsband des 1. Workshops Komponentenorientierte betriebliche Anwendungssysteme (WKBA 1). Magdeburg 1999, S. 3-14.

[Turo2001] *Turowski, K.*: Fachkomponenten - Komponentenbasierte betriebliche Anwendungssysteme. Habil.-Schr. Magdeburg 2001.