Technische Universität Dresden

Fakultät Wirtschaftswissenschaften
Lehrstuhl Wirtschaftsinformatik,
insbesondere Systementwicklung

Musterbasierte Systementwicklung
Eine empirische Untersuchung

Diplomarbeit
zur Erlangung des akademischen Grades
"Diplom-Wirtschaftsinformatiker"

Bearbeitet von:	Thoralf Czichy
Matrikelnummer:	XXXXXXX
Adresse:	XXXXkatu XXX 00XXX Helsinki (Finnland)
Betreuer:	Andreas Dietzsch
Bearbeitungszeitraum:	20. März 2001 - 19. Oktober 2001

Zusammenfassung

Während die Ursprünge von Mustern der Systementwicklung in der Anwendung von Entwurfsmustern liegen, haben sich seit Mitte der 90er Jahre zahlreiche weitere Arten von Mustern entwickelt. Beispielhaft seien hier Analysemuster, Architekturmuster, aber auch Prozess- und Organisationsmuster aufgeführt. Ziel dieser Arbeit ist es, über die Nutzung von Entwurfsmustern hinaus die Anwendung von Mustern in der Praxis der Softwareentwicklung anhand einer empirischen Untersuchung zu beschreiben.

Dazu wird im ersten Teil der Abhandlung zunächst ein theoretischer Rahmen abgesteckt, an dem sich die empirische Untersuchung, die in Form eines Internetfragebogens durchgeführt wird, orientiert. Im zweiten Teil der Arbeit werden dann die Ergebnisse der Befragung dargestellt und interpretiert.

Kern der theoretischen Abhandlung ist zunächst eine Einordnung der verschiedenen Musterarten anhand der Abstraktions-Transformations-Sichtweise auf den Prozess der Systementwicklung. Im Anschluss daran wird dann ein Mikroprozess für die Produktion und den Konsum von Mustern vorgestellt. Anhand dieses Mikroprozesses lassen sich dann verschiedene Ansätze zur Charakterisierung und Automatisierung von Mustern analysieren. Vor- und Nachteile der Anwendung von Mustern werden untersucht. Dabei wird Bezug auf die Grundsätze ordnungsgemäßer Modellierung genommen. Die theoretischen Betrachtungen schließen, basierend auf der Klassifikation von Wissen nach NONAKA, mit einer Einordnung von Mustern in den Wissenskreislauf einer Organisation ab.

Im zweiten Teil der Arbeit werden dann die Ergebnisse der Befragung präsentiert und diskutiert. Dabei werden Aussagen zu folgenden sieben Analysegebieten getroffen: (1) Verständnis von Mustern, (2) Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung, (3) Arten der Verbreitung von Mustern in der Organisation, (4) Musterproduktion und Musternutzung im Vorgehensmodell, (5) Charakterisierung von Mustern, (6) Gesammelte Erfahrungen mit Mustern und (7) Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil.

Inhaltsverzeichnis

0. Ziele und Vorgehen der Studie ... 1

1. Modelle in der Systementwicklung ... 2

1.1 Der Prozess der Systementwicklung ... 2

1.2 Softwareorientierte Systementwicklung ... 6

1.2.1 Die Aktivitäten der softwareorientierten Systementwicklung ... 6

1.2.2 Das Wasserfallmodell als Phasenmodell ... 10

1.2.3 Ansätze zur Überwindung der Schwächen des Wasserfallmodells ... 12

1.3 Implikationen für die empirische Studie ... 18

2. Muster ... 19

2.1 Der Begriff Muster in der Umgangssprache ... 19

2.2 Muster in der Systementwicklung ... 21

2.2.1 Eigenschaften von Mustern in der Systementwicklung ... 21

2.2.2 Muster im Vergleich zu anderen Ansätzen ... 23

2.2.3 Ein Musterklassifikationsschema nach Aktivitäten der Systementwicklung ... 27

2.3 Implikationen für die empirische Studie ... 35

3. Ein musterbasierter Systementwicklungsprozess ... 36

3.1 Muster in den Phasen der Softwareentwicklung ... 36

3.1.1 Ein allgemeiner Mikroprozess für Muster ... 36

3.1.2 Die Charakterisierung von Mustern ... 41

3.1.3 Potenzial für die Automatisierung im Mikroprozess für Muster ... 46

3.2 Das Zusammenwirken von Mustern im gesamten Prozess der Systementwicklung ... 50

3.3 Implikationen für die empirische Studie ... 59

4. Eine Bewertung des Einsatzes von Mustern ... 60

4.1 Eine Bewertung anhand der Grundsätze ordnungsgemäßer Modellierung 60

4.2 Vorteile und Nachteile bei der Nutzung von Mustern 62

4.3 Muster als Wettbewerbsfaktor 65

4.4 Implikationen für die empirische Studie 68

5. Studie über den Einsatz musterbasierter Systementwicklung in der Praxis 69

5.1 Zusammenfassung der Analysegebiete 69

5.2 Zusammenhang zu anderen Studien 71

5.3 Aufbau und Durchführung der Studie 72

6. Auswertung der Studie 75

6.1 Allgemeine Angaben 75

6.2 Signifikanz der abgeleiteten Aussagen 78

6.3 Betrachtungen zu den einzelnen Analysegebieten 79

6.3.1 Das Verständnis von Mustern 79

6.3.2 Arten der Verbreitung von Mustern in der Organisation 81

6.3.3 Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung 86

6.3.4 Musterproduktion und Musternutzung im Vorgehensmodell 94

6.3.5 Die Charakterisierung von Mustern – Musterbeschreibungsformen und Musterorganisationsschemata 95

6.3.6 Gesammelte Erfahrungen mit Mustern 99

6.3.7 Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil 103

7. Zusammenfassung und Ausblick 106

Abbildungsverzeichnis I

Abkürzungsverzeichnis III

Literaturverzeichnis IV

Anhang XV

0. Ziele und Vorgehen der Studie

Neben verschiedenen Erfahrungsberichten, zumeist einzelner Unternehmen, sind in der Fachliteratur nur wenige Studien zu finden, die sich mit der Anwendung von Mustern in der Systementwicklung beschäftigen (vgl. [Bec+01]; [Pre+97a]; [Pre+97b]; [Unge00]). Dabei beschränken sich diese Studien und Berichte zumeist auf die Analyse der Anwendung von Mustern, die sich auf den Detailentwurf von Softwaresystemen beziehen – sogenannten Entwurfsmustern.

Während die Ursprünge des Wiederverwendungsansatzes Muster zweifelsohne in der Entwicklung von Entwurfsmustern liegen (vgl. [Copl96b]), hat sich die Anwendung von Mustern seit Mitte der 90er Jahre jedoch auch auf andere Aktivitäten der Systementwicklung ausgedehnt. Populäre Beispiele sind Analyse-, Architektur- sowie Prozessmuster (vgl. [Fowl97], S.8; [Mar+98], S. 143; [Ambl98], S.1).

Ziel dieser Arbeit ist es, einen Überblick über die Nutzung und das Verständnis von Mustern in der Praxis, über die Anwendung von Entwurfsmustern hinaus, zu geben. Dabei werden neue Erkenntnisse anhand einer empirischen Studie ermittelt. Aus den Ergebnissen der Studie lassen sich dann Aussagen zu folgenden sieben Analysegebieten ableiten:

1. Das Verständnis von Mustern
2. Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung
3. Arten der Verbreitung von Mustern in der Organisation
4. Musterproduktion und Musternutzung im Vorgehensmodell
5. Die Charakterisierung von Mustern - Musterbeschreibungsformen und Muster-organisationsschemata
6. Gesammelte Erfahrungen mit Mustern
7. Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil.

In den ersten Kapiteln dieser Arbeit wird dabei ein theoretischer Rahmen geschaffen, an dem sich dann die empirische Untersuchung orientiert. Dazu wird die Anwendung von Mustern

zunächst in den allgemeinen Prozess der Systementwicklung eingeordnet. Im Anschluss daran werden Muster anhand ihrer Merkmale von anderen verwandten Ansätzen der Systementwicklung, wie z.B. Rahmenwerken, abgegrenzt. Aktivitäten, die sich aus der Analyse des Systementwicklungsprozesses ergeben, bilden dann den Ausgangspunkt für eine Klassifikation existierender Arten von Mustern. Die erstellte Klassifizierung wird in der empirischen Untersuchung beibehalten (Kapitel 1 und 2).

Im Anschluss daran wird ein Mikroprozess für die Anwendung von Mustern entwickelt, der dann Grundlage für die Analyse verschiedener Charakterisierungsansätze (Musterbeschreibungsformen und Musterorganisationsschemata) sowie Automatisierungsansätze ist. Die praktische Anwendung dieser Ansätze wird anhand der empirischen Studie untersucht (Kapitel 3).

Der Einsatz von Mustern wird anhand der Grundsätze ordnungsgemäßer Modellierung bewertet und in der Fachliteratur zu findende Vor- und Nachteile werden für die Nutzung in der empirischen Studie zusammengefasst. Die theoretischen Betrachtungen schließen mit einer kurzen Einordnung von Mustern als Wettbewerbsvorteil, basierend auf deren Wissenscharakter, ab (Kapitel 4).

Nach einer Zusammenfassung der allgemeinen Eigenschaften der Studie werden dann die Ergebnisse der empirischen Untersuchung zur Praxis musterbasierter Systementwicklung im verbleibenden Teil der Arbeit gebündelt dargestellt und analysiert (Kapitel 5 und 6).

1. Modelle in der Systementwicklung

1.1 Der Prozess der Systementwicklung

Ziel der Systementwicklung (engl.: Systems Engineering) ist die Erstellung eines realen Systems, wobei **reale Systeme** „als Teilausschnitt der realen Welt“ ([FeSi94], S. 16) aufgefasst werden. Ein **System** umfasst Komponenten und Beziehungen zwischen diesen Komponenten (vgl. [Habe97], S. 5). Komponenten, die entsprechenden systemeingrenzenden Eigenschaften genügen, werden dabei als Bestandteil des Systems aufgefasst. Diejenigen Komponenten, die diesen Eigenschaften nicht genügen, werden der zum System disjunktiven Umwelt zugerechnet (vgl. [FeSi94], S. 16f). Die Schnittstelle zwischen System und Umwelt wird dabei als Systemgrenze bezeichnet.

Das zentrale Thema dieser Arbeit liegt auf der Betrachtung der Zusammenhänge zwischen Mustern und der Erstellung von realen Systemen, in denen die Softwarekomponente den Hauptbestandteil des zu entwickelnden Systems ausmacht (vgl. [Jac+92], S. 10). Die systemeingrenzenden Eigenschaften können dabei unterschiedlich weit ausgelegt werden.

In einer engen Definition stimmen die zu entwickelnde Softwarekomponente (z.B. ein Programm) und zu erstellendes reales System überein. Damit wäre der Prozess der Systementwicklung im Rahmen einer Fremdentwicklung beispielsweise mit der Übergabe des Programms abgeschlossen.

Um auf Aspekte außerhalb der reinen Erstellung von Software, insbesondere auf typische Wartungsaspekte wie Änderung und Erweiterung besser eingehen zu können, wird der Systembegriff im Folgenden weiter ausgelegt und die Systemgrenze schließt Komponenten wie Nutzer und Hardware, die in einer engen Definition der Umwelt zuzurechnen wären, mit ein. Die Softwarekomponente ist damit nur ein Teil des zu entwickelnden Systems (vgl. [Ott91], S. 3).

Im Rahmen einer hierarchischen Betrachtung von (Ober-)Systemen und Subsystemen (vgl. [Schü98], S. 37ff) bildet die Softwarekomponente des Systems ein Softwaresubsystem. Im Folgenden werden Software, Softwarekomponente, Softwaresystem und Softwaresubsystem synonym verwendet und beschreiben die Softwarekomponente eines zu erstellenden Systems im Sinne des oben entwickelten erweiterten Systembegriffs.

Wie wirkt sich diese Betrachtungsweise nun auf die Modelle für den Prozess der Systementwicklung aus? Generell kann dieser Prozess als die Transformation eines Problems in eine Lösung verstanden werden. Die Eigenschaft Transformationsgrad kann entweder als Entfernung von Problem oder Lösung verstanden werden - im Folgenden wird unter **Transformationsgrad** die Entfernung vom Problem verstanden. Den höchsten Transformationsgrad besitzt die Lösung (vgl. [Reic00], S. 37ff).

Sowohl Problem als auch Lösung sind reale, komplexe Subsysteme der Diskurswelt. Um den Systemcharakter von Problem und Lösung hervorzuheben, wird im Folgenden von Problemsystem und Lösungssystem gesprochen. Die Komplexität beider Systeme macht die

Transformation zu einem im Normalfall nichttrivialen Prozess. Der Komplexitätsabstand zwischen Problemsystem und Lösungssystem wird oft auch als semantische Lücke bezeichnet (vgl. [FeSi94], S. 269; [Kent90], S. 3).

Da der Transformationsprozess sich über einen gewissen Zeitraum erstreckt, existieren Problemsystem und Lösungssystem nie zur selben Zeit in der Diskurswelt. Das Lösungssystem als Ergebnis des Transformationsprozesses ist stets dem Problemsystem zeitlich nachgelagert.

Grundlegendes Mittel zur Beherrschung der Komplexität von Problemsystem, Lösungssystem und Transformationsprozess ist das Mittel der Abstraktion, wie es schon im Rahmen der Zerlegung des realen Systems in Softwaresubsystem und systemeigene Umgebung zur Anwendung kam (vgl. [Schü98], S. 39f). Dabei ist **Abstraktion** definiert als „ ... ein Abziehen, Herauslösen von Teilgehalten, Aspekten, Merkmalen aus einem konkreten Ganzen ... “ ([Broc97]). Mittels der Abstraktion ist es möglich, diejenigen Komponenten eines Systems zu ignorieren, die für den aktuellen Zweck nicht von Bedeutung sind (vgl. [CoYo94], S. 28ff).

Im Rahmen eines mehrstufigen Vorgehens durchläuft ein sequenzieller Transformationsprozess zunächst die Stufen aufeinander aufbauender Abstraktionsphasen, die dann darauf folgend in Konkretisierungsphasen ausgearbeitet werden (vgl. [Pres00], S. 335f). Der Prozess der Konkretisierung verringert den aufgebauten Abstraktionsgrad.

In Anlehnung an JACOBSON (vgl. [Jac+92], S. 14) wird im Folgenden das Ergebnis eines Abstraktionsschrittes bzw. Konkretisierungsschrittes als Modell im Sinne des konstruktionsorientierten Modellbegriffs, der über den reinen Abbildungscharakter eines Modells hinausgeht, bezeichnet (vgl. [Balz98], S. 559).[1] Die im Rahmen des Abstraktions- und Transformationsprozesses gebildeten Modelle bilden den **Modellraum**.

[1] Für eine detaillierter Darstellung unterschiedlicher Auffassungen über den Modellbegriff siehe auch SCHÜTTES Ausführungen (vgl. [Schü98], S.40ff).

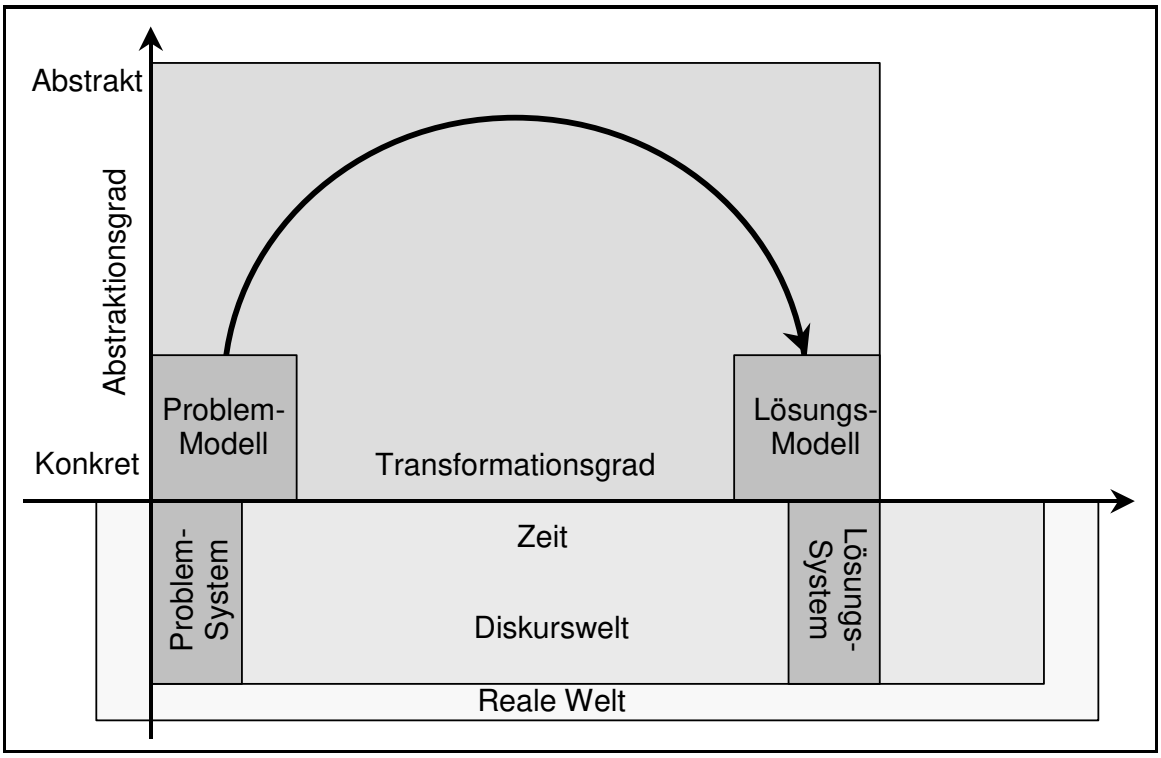

Abbildung 1: Der Übergang vom Problemsystem zum Lösungssystem (vgl. [Essw93], S. 177)

Mit der Übergabe bzw. der Implementierung des Softwaresystems wäre der Prozess der Systementwicklung gemäß dem eng gefassten Systembegriff beendet. Der in dieser Arbeit verwendete erweiterte Systembegriff erstreckt sich jedoch auch auf die direkte Umwelt des Softwaresystems - also im Rahmen der Systementwicklung auf das Lösungssubsystem der Diskurswelt. Damit ergibt sich eine dem Transformationsprozess zeitlich nachgelagerte Phase, die zunächst nicht durch den für die Entwicklung des Softwaresystems typischen Prozess von Abstraktion und Konkretisierung geprägt ist.

Diese der Systemerstellung nachgelagerte Phase ist durch den aktiven Einsatz des Softwaresystems innerhalb des Lösungssubsystems der Diskurswelt gekennzeichnet. Die in dieser Phase mit der Wartung des Systems notwendigen Änderungen und Erweiterungen des Systems, z.B. ausgelöst durch Veränderungen in der Umwelt, können erneute Transformationszyklen verursachen. In Analogie zum Produktlebenszyklus der Betriebswirtschaftslehre ist der Prozess der Systementwicklung im Verständnis dieser Arbeit erst mit der Degeneration des Systems als Ganzes abgeschlossen.

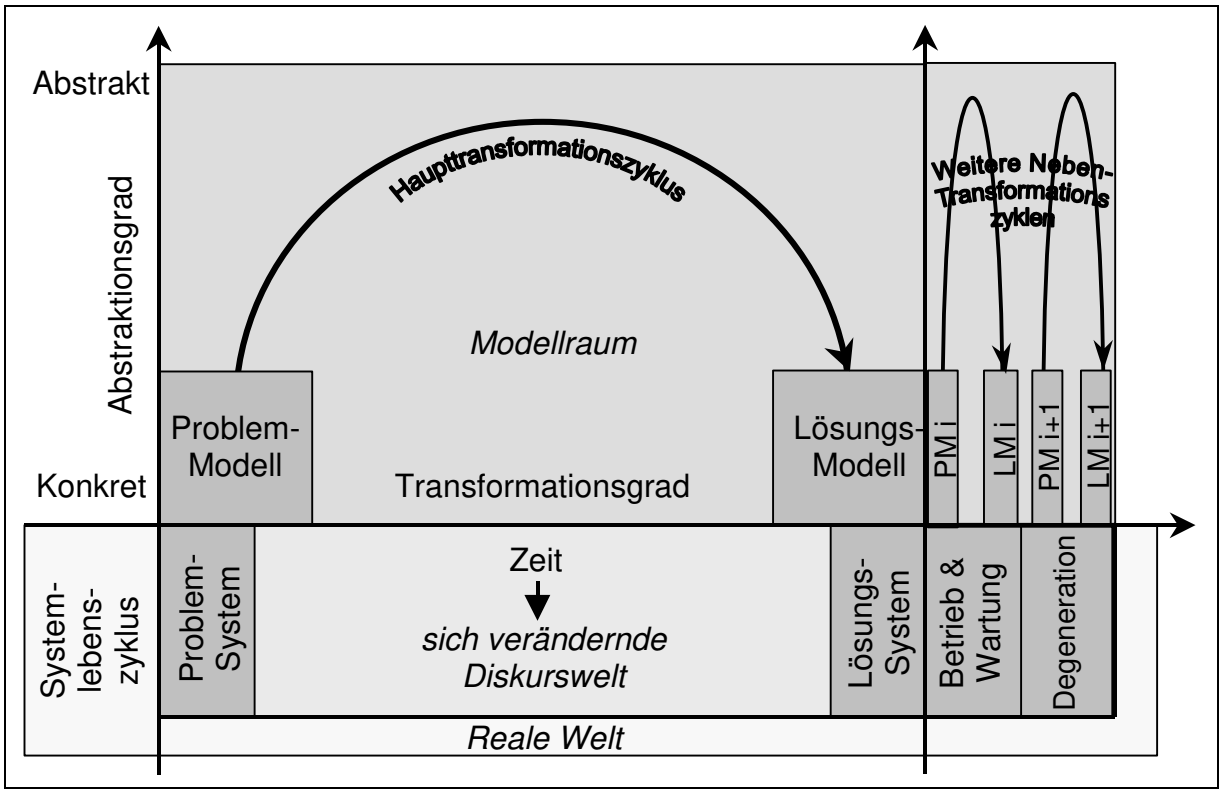

Abbildung 2: Überlagerung von Transformationszyklen und Systemlebenszyklus

Diese strikte Trennung von Transformationszyklen im Modellraum, geprägt durch den Prozess von Abstraktion und Konkretisierung, und dem Systemlebenszyklus wird im späteren Verlauf dieser Arbeit die Analyse von Mustern, deren Eigenschaften und deren Einsatz vereinfachen.

1.2 Softwareorientierte Systementwicklung

1.2.1 Die Aktivitäten der softwareorientierten Systementwicklung

Die im vorhergehenden Abschnitt beschriebene Abfolge von Abstraktions- und Konkretisierungsschritten führt abhängig vom Abstraktions- und Transformationsgrad zu unterschiedlich typischen Aktivitäten der Systementwicklung. Da diese in Abschnitt 2.2.3 eine Grundlage für die Klassifizierung von Mustern bieten, werden diese hier zunächst definiert.[1]

[1] Vergleiche auch ([Booc94], S. 250ff) und ([DeCh93], S. 13ff).

Ak1. Anforderungsanalyse

„The software requirements specification is the primary result of the requirements engineering process“ ([Hofm00], S. 9). Inhalt der Anforderungsspezifikation sind sowohl funktionelle als auch nichtfunktionelle Eigenschaften. Typische funktionelle Eigenschaften sind z.B. die Beschreibung der Benutzerschnittstelle oder die Beschreibung des Verhaltens des zu entwickelnden Systems. Typische nichtfunktionelle Eigenschaften sind wirtschaftliche Beschränkungen sowie die Verlässlichkeit, die Handhabung, die Wartbarkeit und die Leistung des Softwaresystems (vgl. [Hofm00], S. 5ff; [Kruc00], S. 157f). An der Erstellung der Anforderungsspezifikation sind neben Entwicklern auch zukünftige Nutzer und der Auftraggeber beteiligt. Die Anforderungsspezifikationen müssen die Erwartungen aller Beteiligten wiedergeben. Da die Anforderungsspezifikation Grundlage aller anderen Aktivitäten ist, kommt dieser Aktivität eine besondere Bedeutung zu (vgl. [Broo95], S. 199f). Ein typisches Mittel der Anforderungsanalyse sind Use Cases (vgl. [Jac+99], S. 33ff).

Ak2. Analyse

Ziel der Analyse ist die Entwicklung eines Analysemodells. „The analysis model is a precise, concise representation of the problem that permits answering questions and building a solution“ ([Rumb91], S. 149). Die wichtigsten Eigenschaften des Problemsystems werden so beschrieben, dass darauf aufbauend sowohl der Architekturentwurf als auch der Detailentwurf entwickelt werden kann. Die Analyse unterscheidet sich von der Anforderungsanalyse insofern, dass sie keinen Einfluss auf die Anforderungen hat, und damit die Beteiligung der Auftraggeber nicht erfordert. Stattdessen zielt die Analyse darauf ab, eine abstrakte Darstellung des Problemsystems zu erstellen. Beschränkungen, die sich aus der Implementierungsumgebung oder aus nichtfunktionellen Aspekten der Anforderungsanalyse ergeben, werden zunächst nicht betrachtet (vgl. [Kruc00], S. 172).

Ak3. Architekturentwurf

„The architectural design defines the relationship between major structural elements of the software ... “ ([Pres00], S. 330). „The architecture provides the context in which more detailed decisions are made in later design stages“ ([Rumb91], S. 199). Damit ist der

Architekturentwurf ein grobe Beschreibung der Lösung und gibt folgenden Aktivitäten ein Rahmenwerk, das die Möglichkeiten der detaillierten Lösung beschränkt.

Ak4. Detailentwurf

Obwohl eine in vielen Vorgehensmodellen zu beobachtende Abfolge von Architektur- und Detailentwurf nahe legt, dass der Detailentwurf einzig auf den Ergebnissen des Architekturentwurfs basiert, werden im Detailentwurf auch die Ergebnisse der Analyse genutzt. Der Übergang von Analyse zu Design wird in der Literatur häufig als der schwierigste Teil der Systementwicklung gesehen, da hier letztlich die semantische Lücke zwischen Problem- und Lösungssystem überwunden werden muss (vgl. [DeCh93], S.221ff; [Grah94], S. 196).

„ ... the designer is concernded with the job of moving analysis results ... into a particular hardware/software implementation“ ([CoYo91], S. 5). So wird im Rahmen des Detailentwurfs z.B. die endgültige Benutzerschnittstelle entworfen und eine detaillierte Spezifikation von Systemkomponenten (z.B. Klassen) erstellt.

Ak5. Implementierung

Im Rahmen der Implementierung wird die Spezifikation des Detailentwurfs letztlich in Programmkode umgesetzt.

Ak6. Test

Neben dem Test der Ergebnisse der Implementierungsaktivitäten, also dem Programmkode selbst, wird der Testvorgang auch auf die Ergebnisse anderen Aktivitäten angewandt (vgl. [McGr01]). Es gibt viele verschiedene Arten, diese Tests durchzuführen. Allen gemeinsam ist es zu verifizieren, dass die Ergebnisse einzelner oder mehrerer Aktivitäten den gesteckten Zielen bzw. Anforderungen entsprechen (vgl. [StHa97] S. 313ff).

Ak7. Einführung

Hier wird das entwickelte Softwaresystem in das Lösungssystem der Diskurswelt überführt, also in Betrieb genommen. Neben der Verteilung der verschiedenen Softwaresubsysteme auf

reale Basismaschinen wird in dieser Phase auch die initiale Konfiguration des Softwaresystems erstellt.

Ak8. Betrieb und Wartung

Betrieb ist der aktive Einsatz der Software im entwickelten System.

„Maintenance is the activity of managing postdelivery evolution.“ ([Booc94], S. 263). Die durch den Gedanken der Evolution ausgedrückte Weiterentwicklung entspricht der Initiierung neuer Nebentransformationszyklen. Es wird zwischen korrektiver und progressiver Wartung unterschieden. Dabei ist unter korrektiver Wartung die Beseitigung von Fehlern sowie die Stabilisierung und Optimierung zu verstehen und unter progressiver Wartung die Anpassung des Softwaresystems an Änderungen in der Diskurswelt zu verstehen (vgl. [Snee91], S. 26). Mitunter wird progressive Wartung auch als adaptive bzw. perfektionierende Wartung bezeichnet (vgl. [Müll97], S. 6ff).

Ak9. Dokumentation

Unabhängig von der Nutzerdokumentation, also Dokumenten, die dem Nutzer das Verständnis des Softwaresystems erleichtern, ist es Ziel der Dokumentation, den Prozess der Entwicklung eines konkreten Systems, insbesondere mit Hinblick auf die spätere Wartung nachvollziehbar zu beschreiben.

Ak10. Wiederverwendungsanalyse

Im Rahmen geplanter Wiederverwendung, „ … bei der Teile des Systementwurfs und Programmbestandteile schon zum Zeitpunkt ihrer Erstellung für eine spätere Wiederverwendung konzipiert werden“ ([StHa97], S. 345), muss das Wiederverwendungspotenzial dieser Bestandteile im Idealfall schon vor ihrer Erstellung geklärt werden. Weiterhin ist die Analyse existierender Lösungen auf ihre Anwendbarkeit auf die konkrete Aktivität ebenfalls Bestandteil der Wiederverwendungsanalyse.

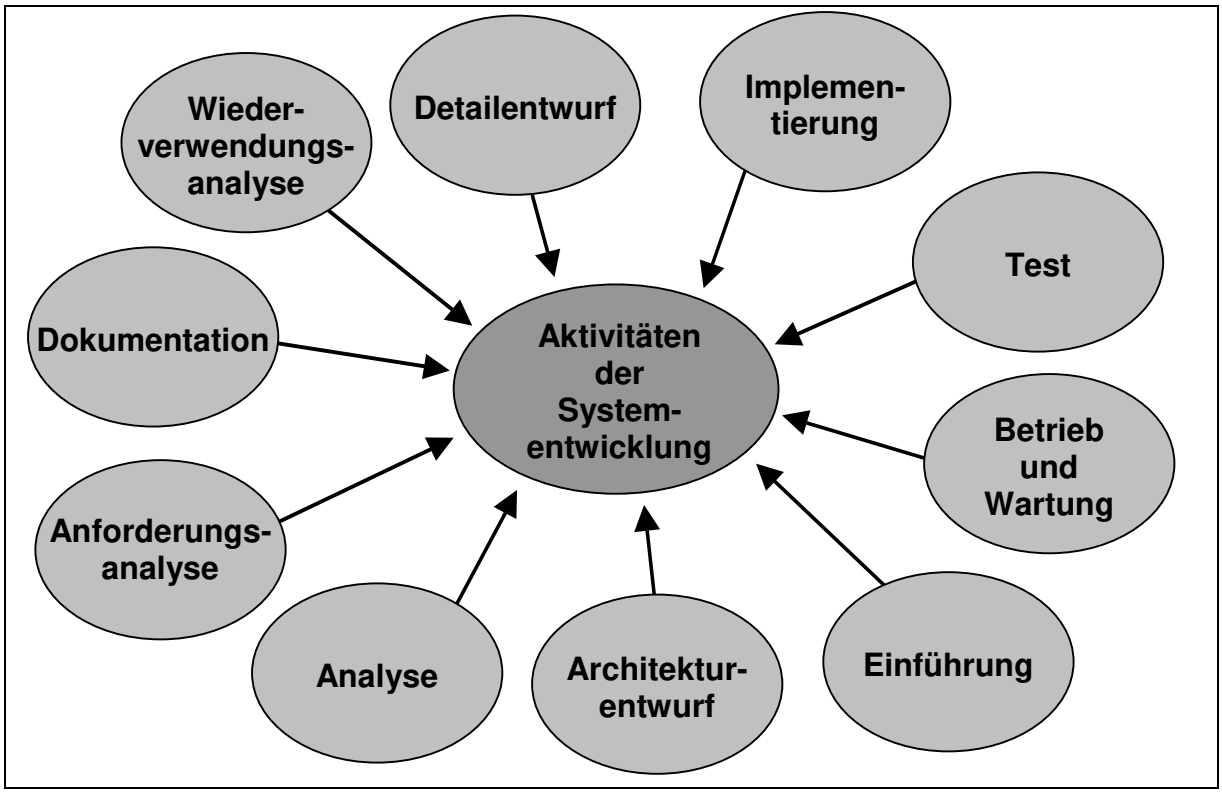

Abbildung 3: Die Aktivitäten der Entwicklung von Softwaresystemen

1.2.2 Das Wasserfallmodell als Phasenmodell

Die Grundidee des Wasserfallmodells ist es, den Prozess der Systementwicklung in mehrere aufeinanderfolgende Phasen zu gliedern (vgl. [Royc70], S. 1ff). Dabei werden diese Phasen sequenziell durchlaufen. Erst mit dem Abschluss einer vorhergehenden Phase beginnt, aufbauend auf deren Ergebnissen, die nächste Phase. Die Bezeichnung und der Inhalt der einzelnen Phasen variiert in der Literatur (vgl. [Boeh76], S. 1226ff; [Ott91], S. 12; [Jenn95], S.62ff; [StHa97], S. 242ff). Ebenfalls variieren die Darstellungen darin, ob es erlaubt ist, zu einer schon durchlaufenen Phase zurückzukehren, falls es sich in einer späteren Phase herausstellt, dass die Ergebnisse der vorherigen Phase(n) korrigiert werden müssen. Dies wird auch als Wasserfallmodell mit Rücklauf bzw. Validation bezeichnet (vgl. [Jenn95], S. 63). Im Wasserfallmodell werden, aufbauend auf den im vorherigen Abschnitt beschriebenen Aktivitäten, verallgemeinernd acht Phasen unterschieden (vgl. [Pres87], S. 21f; [Davi88], S. 1453):

P1. Anforderungsanalyse,

P2. Analyse,

P3. Architekturentwurf,

P4. Detailentwurf,

P5. Implementierung,

P6. Test,

P7. Einführung,

P8. Betrieb und Wartung.

Der Übergang vom Problemsystem zum Modellraum ist Aufgabe der Anforderungsanalyse (P1). Während die Analysephase (P2) durch einen zunehmenden Abstraktionsgrad gekennzeichnet ist, nimmt der Abstraktionsgrad während des Architekturentwurfs (P3), des Detailentwurfs (P4) und der Implementierung (P5) wieder ab. In der Testphase (P6) verändern sich weder Abstraktionsgrad noch Transformationsgrad. In der Einführungsphase (P7) wird das Ergebnis der Implementierung aus dem Modellraum in das Lösungssystem der Diskurswelt überführt. Der Transformationszyklus ist damit abgeschlossen. Die anschließende Phase Betrieb und Wartung (P8) ist eine Phase des Systemlebenszyklus. Dabei kann die Wartung des Systems neue Transformationszyklen auslösen.

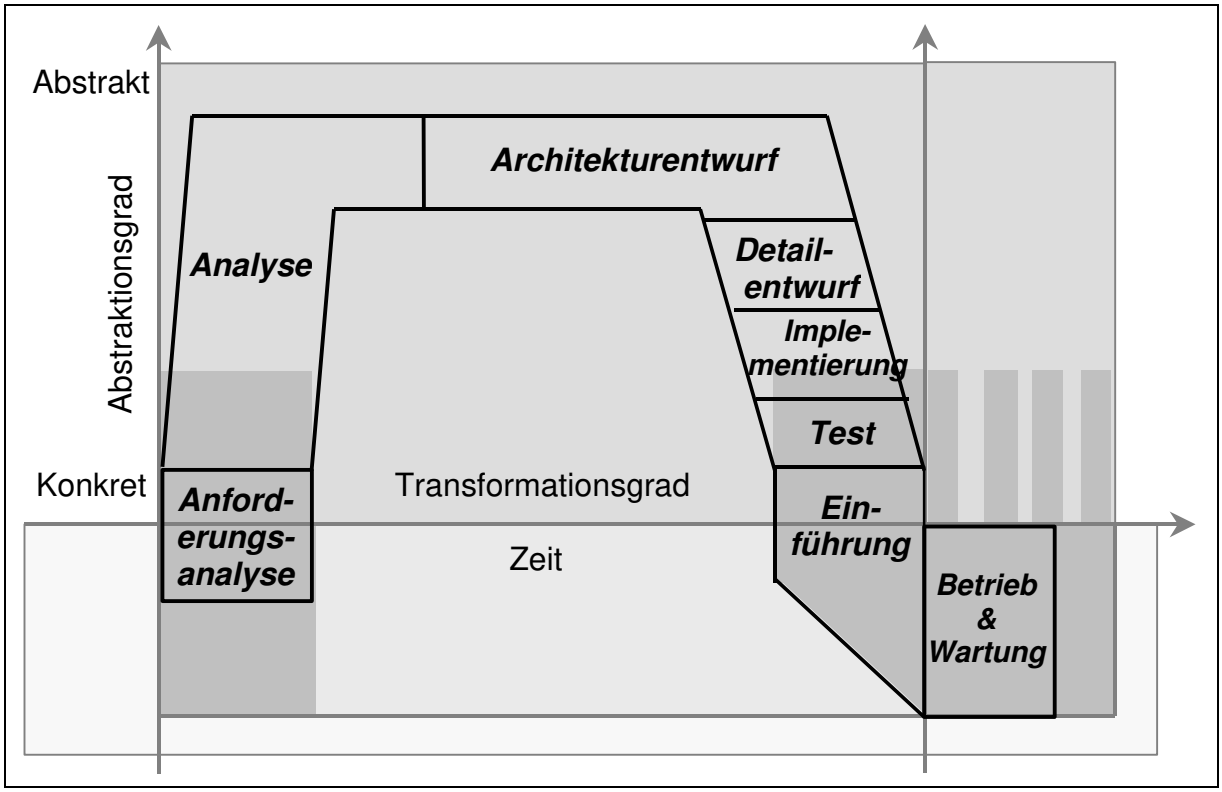

Abbildung 4: Die Einordnung des Wasserfallmodells in den allgemeinen Prozess der Systementwicklung

Da das Wasserfallmodell mit dem Abschluss jeder Phase auf „ ... validierten Zwischenergebnissen(-produkten) ... “ ([Jenn95], S. 62) aufbaut, erleichtert es, in Verbindung mit einer klaren Strukturierung in Phasen, das Projektcontrolling. Da sich die einzelnen Phasen in ihren typischen Anforderungen an die Aufgabenträger (Mensch, Maschine) unterscheiden, erleichtert es auch das Konfigurationsmanagement, da es z.B. die Spezialisierung auf bestimmte Aufgabentypen ermöglicht.

Obwohl das Vorgehen nach dem Wasserfallmodell einige Vorteile, insbesondere gegenüber einer unstrukturierten Vorgehensweise mit sich bringt, hat es gerade wegen seiner Einfachheit einige Nachteile, die neuere Ansätze für Vorgehensmodelle zu umgehen versuchen. Diese Ansätze werden, ausgehend von den Nachteilen des Wasserfallmodells, im nächsten Abschnitt analysiert.

1.2.3 Ansätze zur Überwindung der Schwächen des Wasserfallmodells

Für eine Darstellung der Ansätze zur Überwindung der Schwächen des Wasserfallmodells wird zunächst nur der Haupttransformationszyklus betrachtet. Die Lebenszyklusaktivitäten aus Betrieb und Wartung sind jedoch weiterhin Bestandteil der Systementwicklung. Sowohl Dokumentations- als auch Wiederverwendungsaktivitäten bleiben, obwohl nicht explizit erwähnt, Bestandteil der Systementwicklung. Idealerweise lassen sich beide Aktivitäten in allen Phasen des Haupttransformationszyklus wiederfinden.

Im Folgenden werden zunächst zwei Ansätze betrachtet, die die sequenzielle Abfolge der acht Phasen des Wasserfallmodells abändern. Neben diesen beiden Ansätzen werden drei weitere Ansätze diskutiert, die zwar helfen, Schwächen des Wasserfallmodells zu überwinden, allerdings ohne weiteres auch ohne Abänderung des sequenziellen Prinzips angewandt werden können.

Zu den Schwächen des Wasserfallmodells zählen unter anderem (vgl. [Tayl92], S. 76; [Jac+92], S. 25ff; [CoYo91], S. 10ff).

S1. Es liefert selten die Lösung, die der Kunde hinsichtlich funktioneller sowie nichtfunktioneller Anforderungen erwartet. Im Falle des Wasserfallmodells mit Validation ist es praktisch unmöglich, signifikante Modifikationen vorzunehmen, ohne den kompletten Prozess zum Beginn zurückzuverfolgen.

S2. Im Falle einer langen Entwicklungszeit verändert sich das Problemsystem. Ursprünglich in der Anforderungsanalyse festgelegte Anforderungen fallen weg, verändern sich oder neue Anforderungen kommen hinzu.

S3. Es erfordert spezialisiertes Personal für die einzelnen Aktivitäten, jeweils korrespondierend zu den einzelnen Phasen. Besser wäre ein kontinuierlicher Einsatz des spezialisierten Personals von Beginn bis Ende des Softwaresystementwicklungsprozesses.

S4. Die sequenzielle Ausführung der einzelnen Phasen verursacht lange Entwicklungszeiten. Die Verzögerung in einer Phase pflanzt sich über alle folgenden Phasen fort.

S5. Die sequenzielle Ausführung, insbesondere mit einer Testphase erst am Ende des Entwicklungsprozesses verschiebt die Auffindung und Korrektur von Fehlern unnötigerweise an das Ende des Erstellungsprozesses.

S6. In keiner der acht Phasen wird explizit auf die Nutzung wiederverwendbarer Komponenten als Mittel zur Aufwandsreduktion eingegangen. Dies macht die Nutzung wiederverwendbarer Komponenten unmöglich.

Mitunter sind die in 1.2.1 genannten Vorteile gleichzeitig auch Ausgangspunkt für die Schwächen des Wasserfallmodells.

Die hier diskutierten Ansätze sind die inkrementelle Entwicklung, die Entwicklung von explorativen Prototypen, die Entwicklung eines evolutionären Prototyps, die Wiederverwendungsanalyse sowie die Verwendung objektorientierter Konzepte. Die Entwicklung explorativer Prototypen wird in der Literatur auch als Revolutionary Prototyping und Entwicklung von Rapid Throwaway Prototypen bezeichnet (vgl. [Davi88], S. 1453f; [Grah93], S. 354f).

K1. Inkrementelle Entwicklung

Der Ansatz der inkrementellen Entwicklung versucht die Nachteile S3, S4, S5 des Wasserfallmodells zu vermeiden. Während alle Phasen bis zum Architekturentwurf (P1 bis P3) denen des Wasserfallmodells gleichen, werden die Phasen von Detailentwurf bis zur

Einführung (P4 bis P7) mehrfach durchlaufen. Die Lebenszyklusphase Betrieb und Wartung (P8) überlappt sich, entgegen dem Wasserfallmodell, in der inkrementellen Entwicklung mit den Phasen der Softwaresystementwicklung.

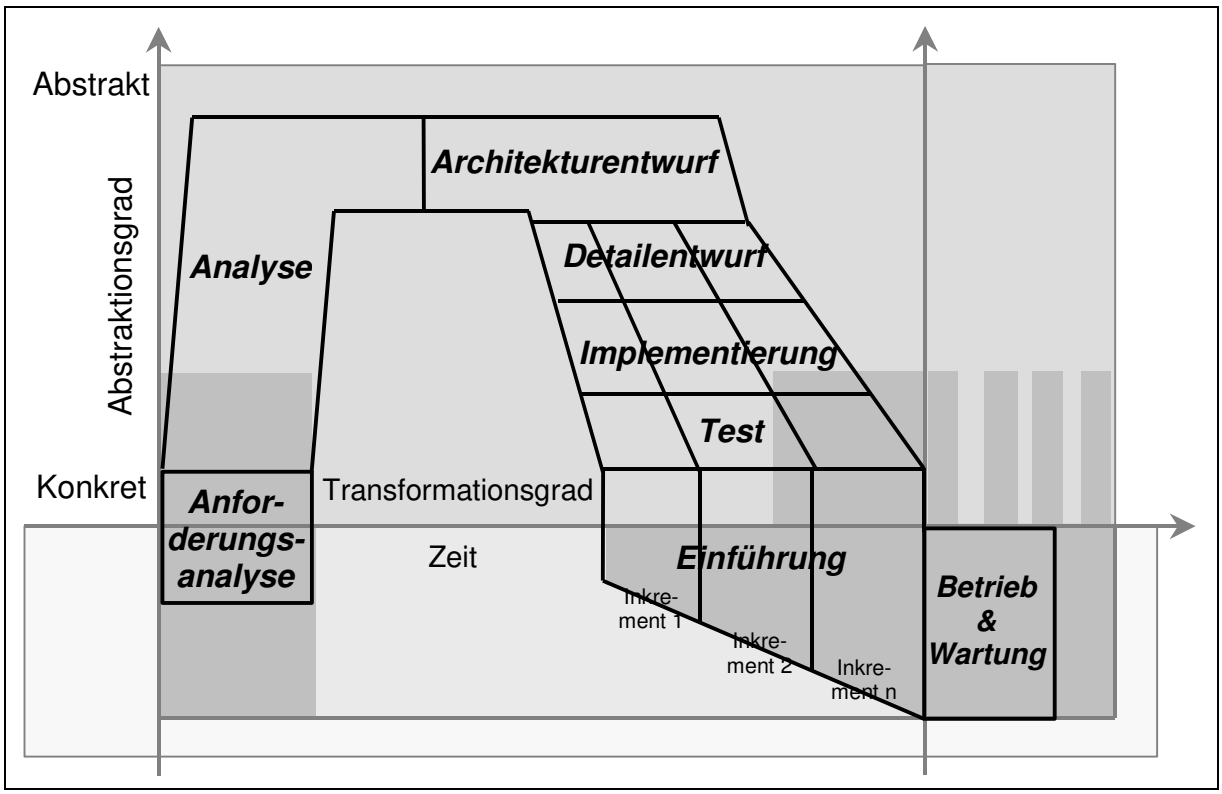

Abbildung 5: Die Einordnung inkrementeller Entwicklung in den allgemeinen Prozess der Systementwicklung

Grundidee der inkrementellen Entwicklung ist es, abgrenzbare Komponenten des zu erstellenden Softwaresystems unabhängig voneinander zu entwickeln. Dabei folgt die Entwicklung jeder dieser Komponenten auf der Basis der schon entwickelten Teile des Softwaresystems. Letztlich wird das komplette Softwaresystem schrittweise durch das wiederholte Durchschreiten der Phasen Detailentwurf, Implementierung, Test und Einführung (P4 bis P7) erstellt. Die Vorteile gegenüber dem Wasserfallmodell ergeben sich aus der Möglichkeit, schon vor der Fertigstellung des kompletten Systems dem Kunden ein, wenn auch noch unvollständiges, Produkt anzubieten, was dann auch schon von Fehlern bereinigt werden kann. Eine Verzögerung in einer der parallel ablaufenden Teilentwicklungsprozesse wirkt sich nicht notwendigerweise auf den Ablauf der anderen Teilentwicklungsprozesse aus. Die Parallelität der Teilentwicklungsprozesse ermöglicht eine bessere Ressourcenverteilung (vgl. [Jac+92], S. 25ff; [Broo95], S. 200f, [Davi88], S. 1454).

K2. Entwicklung mittels eines evolutionären Prototyps

„Ein **Prototyp** ist ein vorläufiges oder temporäres Produkt, mit dem ausgewählte Eigenschaften oder Aspekte des zu entwickelnden endgültigen Produktes erfahrbar und beurteilbar gemacht werden sollen“ ([OOSE01]). Bei der Entwicklung mit einem evolutionären Prototyp wird versucht, durch die Entwicklung eines vorläufigen Produktes die Schwächen S1 und S2 zu umgehen. Dabei wird zu einem möglichst frühen Zeitpunkt im Transformationszyklus ein erster Prototyp entwickelt, der dann den Nutzern für eine erste Evaluation vorgestellt wird. Ziel dieser Evaluation ist es, fehlende, geänderte und nicht mehr benötigte Anforderungen aufzudecken und in die Anforderungsanalyse (P1) und Analyse (P2) mit einfließen zu lassen. Fallen Änderungen in Anforderungsanalyse (P1) und Analyse (P2) an, wird versucht, die Ergebnisse schon abgeschlossener Phasen im Nachhinein zu korrigieren. Im Unterschied zur inkrementellen Entwicklung wird in der Entwicklung mittels evolutionären Prototyps nicht davon ausgegangen, dass die Anforderungsspezifikation mit Abschluss der Anforderungsanalyse abgeschlossen ist (vgl. [Davi88], S. 1454).

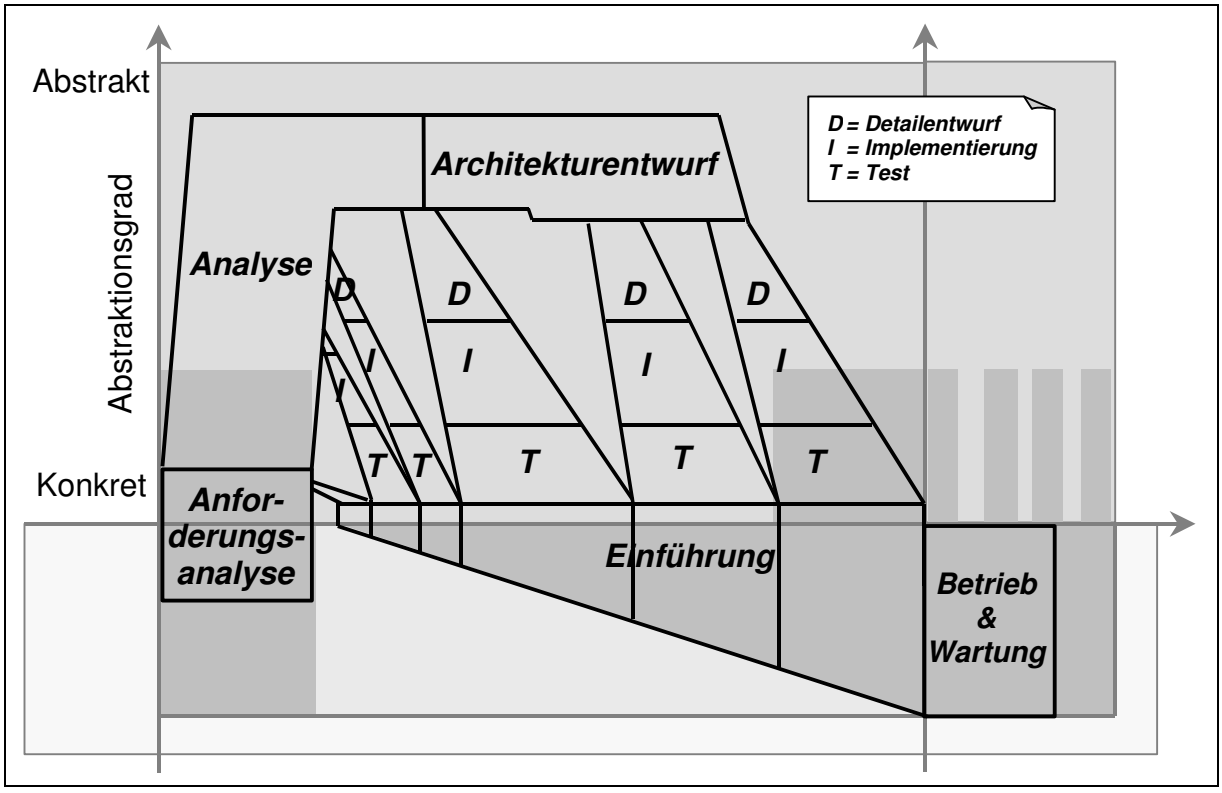

Abbildung 6: Die Einordnung der Entwicklung mittels evolutionären Prototyps in den allgemeinen Prozess der Systementwicklung

Vorteil der Entwicklung mittels eines evolutionären Prototyps gegenüber dem Wasserfallmodell ist eine verbesserte Anforderungsanalyse, die den zukünftigen Nutzer besser einbindet. Weiterhin ist es möglich, auf während des Transformationszyklus entdeckte Veränderungen in den Anforderungen einzugehen und notwendige Korrekturen in der Entwicklung vorzunehmen.

K3. Entwicklung mittels explorativer Prototypen

Im Gegensatz zur Entwicklung mittels evolutionären Prototyps ist hier der Prototyp nur ein temporäres Produkt, das nach Evaluation unter Beteiligung der Nutzer nicht weiterverwendet wird. Typisches Beispiel sind Prototypen grafischer Benutzeroberflächen. Ziel ist es, eine verbesserte Anforderungsspezifikation zu erstellen und das gemeinsame Verständnis der Anforderungen zu erhöhen. Damit beschränkt sich die Nutzung explorativer Prototypen letztlich auf die Anforderungsanalyse (vgl. [Oest97], S. 104ff [Davi88], S. 1454). Diese Beschränkung impliziert, dass durch Anwendung explorativer Prototypen die Vorgehensweise des Wasserfallmodells nicht geändert wird. Die Schwäche S1 des Wasserfallmodells wird durch die bessere Abstimmung der Anforderungsspezifikation umgangen.

K4. Wiederverwendung

Die Nutzung wiederverwendbarer Komponenten wird mitunter auch als Konzept zur Überwindung der Schwächen des Wasserfallmodells gesehen und zielt speziell auf die Schwäche S6 ab (vgl. [Hofm00], S. 99ff). Dabei führt die Verwendung fertiger Komponenten nicht dazu, dass sich der Ablauf des Wasserfallmodells verändert. Zwar müssen die Aktivitäten der einzelnen Phasen um die in Abschnitt 1.2.1 beschriebenen Aktivitäten der Wiederverwendungsanalyse ergänzt werden, allerdings ändert dies nicht die sequenzielle Abfolge der acht grundlegenden Phasen.

K5. Verwendung objektorientierter Konzepte

Ebenfalls wird die Verwendung objektorientierter Konzepte oft auch als Ansatz zur Überwindung der Schwächen des Wasserfallmodells aufgeführt (vgl. [Tayl92], S. 77), allerdings handelt es sich auch hier eher um ein Konzept, das nicht notwendigerweise die Struktur des Wasserfallmodell abändert. Die Wiederverwendung auf Basis vorgefertigter

Klassen lässt sich durch die strikte Trennung von Definition und Implementierung sowie die mögliche gemeinsame Kapselung von Zustand und Verhalten als Ansatz zur Überwindung der Schwäche S6 sehen. Es ist möglich, das ursprüngliche Wasserfallmodell unverändert unter gleichzeitiger Anwendung objektorientierter Konzepte zu verwenden. Wasserfallmodell und objektorientierte Konzepte sind einander orthogonal einsetzbar. Der Einsatz objektorientierter Konzepte im Rahmen objektorientierter Methoden führte jedoch unumstritten zur Entwicklung unterschiedlicher Vorgehensmodelle (vgl. [Jac+94], S. 35ff, 71ff; [Kruc00], S. 142).

Zusammenfassend ergeben sich somit zwei Ansätze zur Abänderung des Wasserfallmodells und drei Ansätze zur Erweiterung des Wasserfallmodells.

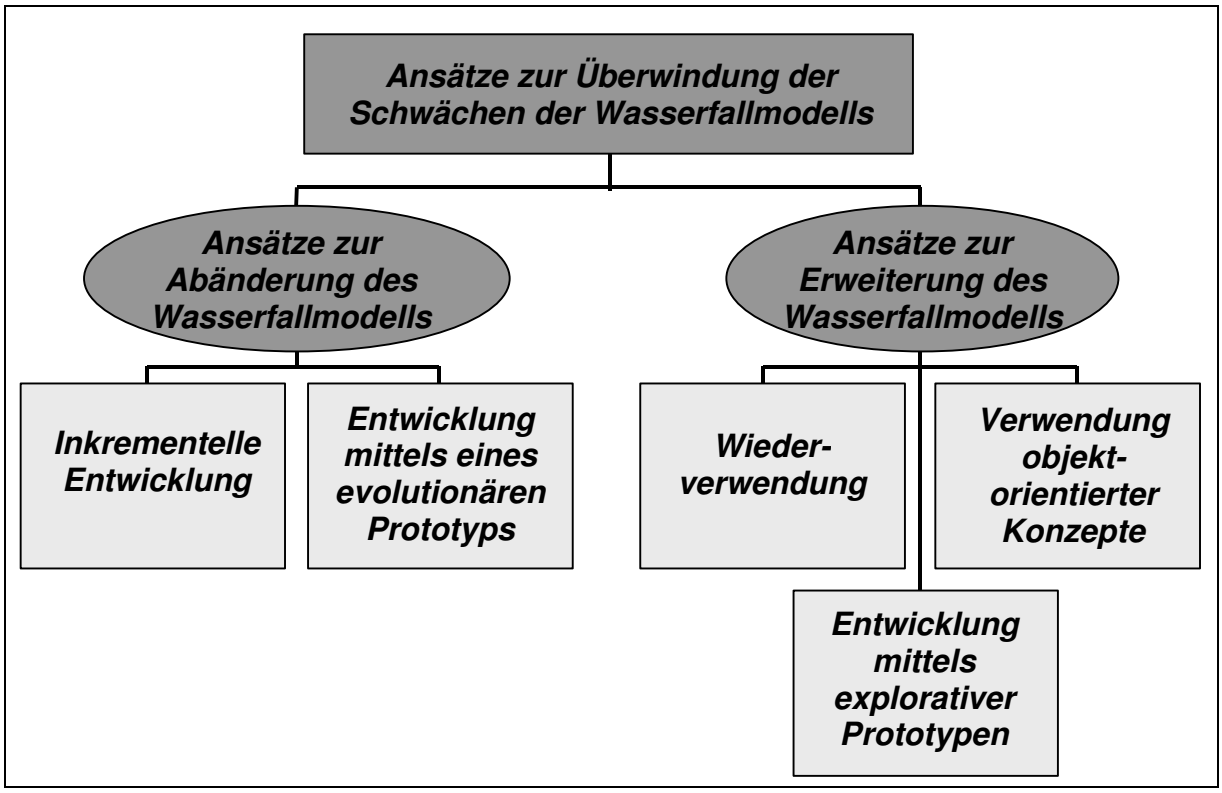

Abbildung 7: Die Ansätze zur Überwindung der Schwächen des Wasserfallmodells

Beispielhaft sei hier der Rational Unified Process anhand der oben herausgearbeiteten Konzepte zur Überwindung der Schwächen des Wasserfallmodells analysiert. So lässt dieser sich als inkrementelles Vorgehensmodell (K1), das die Entwicklung von Prototypen (K2 oder K3) unterstützt, kennzeichnen (vgl. [Jac+99], S. 7f, S. 346, S. 387f). Idealerweise sind diese Prototypen evolutionär (K2) (vgl. [Kruc01]). Dabei zeigt der Rational Unified Process derzeit

noch Schwächen in der Wiederverwendungsanalyse, insbesondere mit Hinsicht auf die Analyse potenzieller Wiederverwendungsmöglichkeiten auf Basis von Komponenten (vgl. [Ambl01]).

1.3 Implikationen für die empirische Studie

Hier werden die Aspekte zusammengefasst, die sich aus den Ausführungen in den vorherigen Abschnitten des Kapitels 1 ergeben und im Rahmen der empirischen Studie näher untersucht werden.

Die dort herausgearbeiteten Aktivitäten der Systementwicklung bilden die Grundlage für eine Klassifizierung von Musterarten in Abschnitt 2.2.3. Um den Grad der Unterstützung der einzelnen Aktivitäten durch Muster zu analysieren, wird anhand des Fragebogens zunächst analysiert, inwiefern diese Aktivitäten Bestandteil der täglichen Arbeit der Befragten sind. Mit Hinsicht auf eine Automatisierung der Nutzung ergibt sich die Frage nach der generellen Unterstützung dieser Aktivitäten durch Werkzeuge, z.B. durch die Anwendung grafischer Entwurfswerkzeuge wie Rational Rose (Frage 8A, 8B). Der Grad der Unterstützung der in Abschnitt 3.1.3 herauszuarbeitenden Automatisierungsansätze wird dann mit dem Grad der Unterstützung durch allgemeine Werkzeuge verglichen.

Die Anwendung und Erstellung von Mustern wird in Abschnitt 3.1.1 in einen allgemeinen Mikroprozess für Muster eingeordnet. Dieser Mikroprozess legt nahe, dass die Nutzung von Mustern in das genutzte Vorgehensmodell integriert werden kann. Daraus ergibt sich die Frage, ob die Softwareentwicklung in der konkreten Organisation einem Vorgehensmodell folgt (Frage 4A).

Die im Abschnitt 3.2 beschriebenen Zusammenhänge zwischen der Nutzung von Mustern und den Konzepten zur Überwindung der Schwächen des Wasserfallmodells (K1 bis K5) werden in ihrer praktischen Anwendung näher untersucht. Aufgrund der Art der Umfrage, die nur die Befragung von Musternutzern einschließt, lassen sich jedoch keine Aussagen zu einem eventuell existierenden (kausalen) Zusammenhang zwischen der Nutzung eines der Konzepte zur Überwindung der Schwächen des Wasserfallmodells und der Nutzung von Mustern machen (Frage 4B).

2. Muster

2.1 Der Begriff Muster in der Umgangssprache

Die Benutzung eines umgangssprachlichen und vielseitig verwendeten Begriffs wie Muster bringt in der Systementwicklung mitunter Probleme hinsichtlich einer klaren Abgrenzung mit sich. Um so wichtiger ist es, eine einheitliches Verständnis zu erlangen. Der BROCKHAUS definiert Muster (engl.: Pattern) sowohl allgemein als auch für spezielle Anwendungsgebiete folgendermaßen:

- Allgemein: „A) (Arbeits-) Vorlage, Modell, wonach etwas hergestellt wird; B) Etwas in seiner Art Vollkommenes, Vorbild“ ([Broc97]).
- Kunst (speziell): „regelmäßig wiederkehrender Flächendekor, der auf der Kombination eines einzelnen oder mehrerer Elemente (Figuren) beruht“ ([Broc97]).
- Jazz (speziell): „Variationsform, bei der die Melodie bei jeder Wiederholung verändert, das rhythmische Schema jedoch beibehalten wird“ ([Broc97]).
- Sprachwissenschaften (speziell): „ein Strukturmodell für einen Satz, nachdem eine unbegrenzte Anzahl gleichstrukturierter Sätze mit unterschiedlichen lexikalischen Elementen gebildet werden kann, z.B. Artikel + Adjektiv + Substantiv + Verb“ ([Broc97]).

Daraus lassen sich verallgemeinernd folgende Aspekte von Mustern sowohl aus der allgemeinen Begriffsbestimmung, als auch aus der Anwendung in anderen Fachgebieten ableiten:

E1. Ein Muster ist eine Vorlage.

E2. Ein Muster ist vollkommen.

E3. Ein Muster ist ein Vorbild.

E4. Ein Muster erscheint regelmäßig immer wieder.

E5. Ein Muster kann wieder und wieder angewandt werden.

E6. Ein Muster beruht auf der Kombination einzelner Elemente.

E7. Ein Muster verändert sich bei jeder Anwendung, das generelle Schema wird jedoch beibehalten.

Zusätzlich zu den oben schon aufgeführten Anwendungsgebieten wird der Begriff Muster in der Systementwicklung häufig mit den Werken ALEXANDERS, der den Begriff für den Anwendungsbereich der Architektur geprägt hat, in Verbindung gebracht (vgl. [Ale+77]). „Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution. ... As an element of language, a pattern is an instruction, which shows how this spatial configuration can be used, over and over again, to resolve the given system of forces, wherever the context makes it relevant“ ([Alex79], S. 247).

Die Unterteilung in Kontext, Problem und Lösung zielt auf den Vorlagencharakter eines Musters ab und wird im Rahmen der Systementwicklung häufig als Grundlage für die Gliederung einer Musterbeschreibung benutzt (vgl. Abschnitt 3.1.2). Die Eigenschaft der wiederholten Anwendung (E5) ist auch in ALEXANDERS Definition wiederzufinden.

BROWN et al. prägten den Begriff Antimuster (engl.: AntiPattern) (vgl. [Bro+98]). Ähnlich der Bezeichnung negative Muster (vgl. [KiBe96], S. 76 und S. 81) handelt es sich trotz des abweichenden Begriffs auch hier um Muster im Sinne ALEXANDERS, angewandt auf den Bereich der Systementwicklung.

„The **AntiPattern** is a standard format that represents recurring software development problems and their refactored solutions“ ([Bro+00], S. XXVII).

Sowohl die beiden Teile Problem und Lösung lassen sich direkt in dieser Definition wiederfinden. Der Kontext, von ALEXANDER definiert als „ ... a certain system of forces which occurs repeatedly ... “ ([Alex79], S. 247) verlagert sich für Antimuster auf die Gründe und Symptome einer ungünstigen Lösung: „An AntiPattern examines the causes, symptomes, and consequences of implementing the bad solution ... “ ([Bro+00], S. XXVII). Obwohl der Begriff Antimuster aus der Anwendung in der Systementwicklung entstanden ist, ist es durchaus auch denkbar, die zugrundeliegende Idee auf andere Anwendungsgebiete, z.B. die Architektur, anzuwenden.

2.2 Muster in der Systementwicklung

2.2.1 Eigenschaften von Mustern in der Systementwicklung

Zusätzlich zu den schon erarbeiteten Eigenschaften aus Abschnitt 2.1 werden hier Aspekte von Mustern ausgearbeitet, die sich speziell aus deren Anwendung in der Systementwicklung ergeben. Dabei wird in diesem Abschnitt noch keine Bewertung hinsichtlich potenziellen Nutzens oder möglicher Nachteile des Einsatzes von Mustern vorgenommen (siehe dazu Abschnitt 4).

Obwohl Muster nicht notwendigerweise in expliziter Form zur Verfügung stehen müssen, werden sie durch Experten(-gruppen) mehr oder weniger unbewusst genutzt. Durch eine schriftliche Fixierung dieser Muster wird das Wissen dieser Experten auch Nichtexperten verfügbar (vgl. [Gam+94], S. 2). Muster beschreiben dabei auch ein gemeinsames Vokabular mit einer einheitlichen Semantik, dessen sich Experten in der Kommunikation mit anderen Experten bedienen (vgl. [Bus+98], S. 6). Abgeleitet aus dem Aspekt der Abhängigkeit von Expertenwissen und der Eigenschaft der wiederholten Nutzung (E4, E5) ergibt sich die Forderung, dass Muster mehrfach in realen Systemen erfolgreich benutzt wurden. Aus dieser Forderung ergibt sich auch, dass Muster nie neue, innovative Lösungen beschreiben (vgl. [Pesc97], S. 130), sondern stets auf Vorgehensweisen beruhen, die sich in der Praxis bewährt haben (vgl. [Gam+94], S. 2; [Coa+95], S. XIV).

E8: Muster fixieren Expertenwissen.

E9: Muster vereinheitlichen die Kommunikation von Experten.

E10: Muster wurden schon mehr als einmal erfolgreich angewandt - „Rule of Three“ (vgl. [Risi01a]).

E11: Muster sind bewährte praktische Vorgehensweisen, keine innovativen Lösungen.

Die Grundlage eines jeden Musters ist die Verbindung von Problem und Lösung. Dabei wird in Anlehnung an ALEXANDER häufig auch der Kontext, also die Beschreibung der allgemeinen Situation und der Umstände bzw. Kräfte, die in der Lösung zu berücksichtigen sind, mit aufgenommen (vgl. [MoMa97], S. 7; [Vlis98], S. 147). Aufbauend auf dem Tripel von Problem, Lösung und Kontext entwickelten sich mehrere Formen für die Beschreibung

einzelner Muster (vgl. Abschnitt 3.1.2). In der Regel lässt sich ein einzelnes Muster isoliert beschreiben, obwohl es dabei meist auf weitere abgrenzbare, verwandte Muster Bezug nehmen muss. Die Beschreibung eines Musters nimmt dabei selten mehr als einige Seiten in Anspruch.

E12: Ein Muster beschreibt ein Problem-Lösungs-Paar.

E13: Ein Muster beschreibt die Kräfte bzw. Umstände, durch die es in seiner Lösung beschränkt ist.

E14: Muster lassen sich isoliert und in relativ kurzer Form beschreiben.

Eine oft diskutierte Frage ist, ob Muster dem objektorientierten Paradigma (K5) folgen (vgl. [Vlis98], S. 8; [Copl96a]). Obwohl beispielsweise alle Muster aus [Gam+94] dem objektorientierten Konzept folgen, existieren genauso gegenteilige Auffassungen (vgl. [Hay95]; [Whit95]). „ ... the notion of patterns is not tied to any methodology or language“ [Risi01a].

E15: Muster sind unabhängig von objektorientierten Konzepten.

Entsprechend der in Kapitel 2.1 aufgeführten Definition ALEXANDERS werden für den weiteren Verlauf dieser Arbeit **Muster der Systementwicklung** relativ allgemein definiert:

„Each pattern is a three-part rule, which expresses a relation between a certain context, a problem, and a solution" ([Alex79], S. 247).

Eine **Definition** ist " … die Bestimmung eines Begriffs durch die Angabe seiner wesentlichen Merkmale" ([Broc97]). Aufgrund der Allgemeinheit der ALEXANDERSCHEN Definition werden Muster in dieser Arbeit zusätzlich anhand der Merkmale E1 bis E15 bestimmt. Im Verständnis dieser Arbeit erfüllt ein Muster der Systementwicklung die Eigenschaften E1 bis E15. Im nächsten Abschnitt werden Muster anhand dieser Eigenschaften von anderen verwandten Ansätzen der Systementwicklung abgegrenzt.

Dabei legen die verschiedenen Musterarten, wie sie im Abschnitt 2.2.3 herausgearbeitet werden, durchaus unterschiedlichen Wert auf die Ausprägung der beschriebenen Eigenschaften. So entspricht die in Eigenschaft E6 aufgeführte Kombination von

Komponenten den Klassen und Objekten in objektorientierten Entwurfsmustern, wohingegen im Falle von Organisationsmustern die Kombination von Komponenten schwieriger wiederzufinden ist und am ehesten mit der Kombination von Organisationseinheiten innerhalb eines Organisationsdiagramms zu vergleichen ist (vgl. Abschnit 2.2.3).

2.2.2 Muster im Vergleich zu anderen Ansätzen

Um das Verständnis von Mustern einzugrenzen, werden im Folgenden Muster gegen andere Ansätze der Systementwicklung, insbesondere gegen Strategien/Techniken, Algorithmen, Rahmenwerke, Klassenbibliotheken, Referenzmodelle und Komponenten abgegrenzt. Alle Ansätze haben das Ziel gemeinsam, aus der Erfahrung bzw. den Ergebnissen abgeschlossener Projekte zu profitieren. Teilweise wurden diese Ansätze bereits von Reichardt anhand von Transformationsniveau, Abstraktionsgrad, Granularität, Geschlossenheit und Formalisierungsgrad zueinander in Beziehung gesetzt (vgl. [Reic00]).

Strategien/Techniken

„A **strategy** is a plan of action intended to accomplish a specific objective“ ([Coa+95], S. XIV). Damit haben die beiden Begriffe Strategie und Technik im Rahmen der Systementwicklung die gleiche Bedeutung (vgl. [Bac+95], S. 16f; [Czic98], S. 5ff). Eine Vorgehensweise, um zu entscheiden, welcher Klasse eine Methode zugeordnet werden soll, ist beispielsweise eine Strategie (vgl. [Coa+95], S. 409).

Während Strategien und Muster zum Beispiel beide Expertenwissen fixieren (E8), wieder und wieder angewandt werden (E5) sowie als Vorbild charakterisiert werden können (E3), haben Strategien bzw. Techniken jedoch nicht den Vollkommenheitsanspruch von Mustern (E2) und lassen sich auch nicht als Kombination einzelner Komponenten (E6) bzw. als Problem-Lösungs-Paar unter Bezugnahme auf die einwirkenden Kräfte (E12, E13) beschreiben.

Algorithmen

Während Algorithmen in der Literatur nur selten als Muster bezeichnet werden, haben sie doch zahlreiche Gemeinsamkeiten mit Mustern. SEDGEWICK definiert **Algorithmen** als „ein Verfahren zur Lösung eines Problems, ... das für eine Realisierung in Form eines Programms geeignet ist“ ([Sedg91], S. 22). Mit Einschränkungen kennzeichnen sämtliche Eigenschaften

von Mustern auch Algorithmen. Wichtigster Unterschied ist, dass Algorithmen im Allgemeinen zunächst in einem innovativen Prozess entwickelt werden und damit der Eigenschaft der praktischen Anwendung widersprechen (E10, E11). Algorithmen lassen sich auch selten als Kombination einzelner Komponenten beschreiben (E6) und helfen stattdessen bei „der Lösung von Berechnungsproblemen“ (vgl. [Bus+98], S. 421). Zumeist unterscheiden sich die Ergebnisse verschiedener Anwendungen von Algorithmen weniger voneinander (E7) als im Falle der Anwendung von Mustern. Auffällig ist, dass Algorithmen genauso wie Muster die Problem-Lösungs-Beziehung hervorheben (E12) und auch auf die Beschreibung der Umstände bzw. Kräfte, die mit der Lösung einhergehen, besonderen Wert legen (E13) (vgl. [Sedg91], S. 23, S. 93ff).

PREE geht sogar so weit, Algorithmen als Entwurfsmuster zu bezeichnen. „As a consequence, books on algorithms also fall into the same category of general design patterns“ ([Pree94], S. 61).

Referenzmodelle

„Anhand charakteristischer Eigenschaften wird ein Sachverhalt mit seinen gültigen Ausprägungsformen in einem **Referenzmodell** allgemein beschrieben“ ([Schm95], S.206). Der Begriff Referenzmodell ist ein vielseitig benutzter Begriff. Typische Beispiele sind das ISO/OSI Basis-Referenzmodell für die Kommunikation offener Systeme und das SAP R/3 Referenzmodell (vgl. [Czic98], S. 7ff).

Referenzmodelle beziehen sich im Allgemeinen auf die Aktivitäten der Analyse und des Architekturentwurfs (vgl. [Reic00], S. 58ff). Damit korrespondieren sie am ehesten mit Analysemustern und Architekturmustern.[1] So lässt sich die Anwendung des Schichtenmusters (vgl. [Bus+98], S. 32ff) beispielsweise im ISO/OSI Basis-Referenzmodell für die Kommunikation offener Systeme wiederfinden. Muster als auch Referenzmodelle haben den Anspruch der Vollkommenheit (E2), dienen als Vorbild (E3), kombinieren einzelne Elemente (E6), fixieren Expertenwissen (E8) und sind bewährte praktische Vorgehensweisen (E11). Während Muster sich jedoch isoliert in relativ kurzer Form beschreiben lassen (E14), ist die Beschreibung von Referenzmodellen meist um einiges umfangreicher.

[1] Analyse- und Architekturmuster werden in Abschnitt 2.2.3 abgegrenzt.

Klassenbibliotheken

Klassenbibliotheken sind genauso wie Muster ein Ansatz zur Wiederverwendung (E5) von Ergebnissen vorangegangener Projekte. Dabei bewegen sich **Klassenbibliotheken**, definiert als „problembereichsunabhängige Sammlung zusammengehöriger Klassen, die sich in der Regel auf die Programmierung beziehen und auf eine Programmiersprache beschränkt sind“ ([StHa97], S. 354), auf einer im Vergleich zu Mustern konkreteren Ebene. Die Nutzung von Klassenbibliotheken impliziert die Wiederverwendung von ausprogrammiertem Kode und widerspricht damit der Eigenschaft der Veränderlichkeit (E7) von Mustern. Während bei der Nutzung von Mustern die wiederverwendete Einheit selbst eine Kombination einzelner Komponenten ist (E6), wird erst bei der Nutzung von Klassenbibliotheken aus einer Vielzahl relativ unabhängiger Klassen eine Kombination von Komponenten gebildet. Der Begriff der Klasse impliziert die Verwendung objektorientierter Konzepte (E15).

Anderseits sind im Vergleich zu Mustern auch einige Gemeinsamkeiten zu finden. So haben Klassenbibliotheken auch einen Vorlagencharakter (E1) und sind bewährte praktische Vorgehensweisen (E11). Dabei lassen sich die Klassen von Klassenbibliotheken ähnlich wie Muster isoliert und in relativ kurzer Form beschreiben (E14). Muster, insbesondere Entwurfsmuster und Idiome spielen eine besondere Rolle im Entwurf und der Dokumentation von Klassenbibliotheken.

Rahmenwerke (engl.: Frameworks)

„Ein **Rahmenwerk** ist eine Menge kooperierender Klassen, die unter Vorgabe eines Ablaufes eine generische Lösung für eine Reihe ähnlicher Aufgabenstellungen bereitstellen“ ([OOSE01]). Die Anwendung eines Rahmenwerks bewegt sich damit immer in einer bestimmten Domäne. Typischerweise diktiert die Nutzung eines bestimmten Rahmenwerks die Architektur eines zu entwickelnden Systems. Um Rahmenwerke von Klassenbibliotheken zu unterscheiden, wird oft die Umkehr der Kontrolle, also die Übergabe der Kontrolle an das Rahmenwerk, als Charakteristikum von Rahmenwerken genannt (vgl. [CoSc95], S. 1ff; [Gam+94], S. 27f).

Reichardt ordnet Rahmenwerken einen niedrigeren Abstraktionsgrad, einen höheren Formalisierungsgrad und eine höhere Granularität zu. Im Grad der Geschlossenheit ordnet er Muster zwischen der starken Geschlossenheit von Black-Box-Rahmenwerken, die keine Informationen über die innere Struktur bereitstellen, und der geringen Geschlossenheit von White-Box-Rahmenwerken ein. Mit Ausnahme von Entwurfsmustern und Idiomen ist der Transformationsgrad von Mustern geringer als der von Rahmenwerken (vgl. [Reic00], S. 52ff).

Sowohl Muster als auch Rahmenwerke werden wieder und wieder verwendet (E5), fixieren Expertenwissen (E8) und kombinieren einzelne Elemente (E6). Eigenschaften wie die Beschreibung als Problem-Lösungs-Paar (E12) mit Blick auf die wirkenden Kräfte (E13) oder die Möglichkeit einer isolierten, kurzen Beschreibung gelten nicht für Rahmenwerke. Trotzdem verspricht die Kombination von Rahmenwerke und Mustern, insbesondere für den Bereich der Dokumentation hohe Synergieeffekte (vgl. [CoSc95], S. 5).

Die Erstellung von Rahmenwerken folgt letztlich dem Prozess der Systementwicklung, wobei das Rahmenwerk die Rolle des Lösungssystems übernimmt. Daraus folgt, dass sich die positiven Effekte der Nutzung von Mustern (vgl. Abschnitt 4) auch in der Erstellung von Rahmenwerken niederschlagen. Damit lassen sich im Rahmen des Transformationszyklus auch alle Muster, die mit Aktivitäten des Transformationszyklus bzw. des Systemlebens-zyklus in Beziehung stehen, anwenden.

(Software-) Komponenten

Komponenten sind ebenso wie Muster ein Ansatz der Wiederverwendung (E5). „Eine **Komponente** ist ein unabhängig verwendbares Stück Software, bestehend aus einer Schnittstelle, einem Typmodell, einer Implementierung und einem Executable“ ([OOSE01]). Die Beschreibung als Software schließt ähnlich wie für Rahmenwerke die Vorteile der Nutzung von Mustern innerhalb des Transformationszyklus bei der Erstellung von Komponenten mit ein. Komponenten lassen sich in Analogie zu Mustern isoliert in relativ kurzer Form beschreiben (E14), sie fixieren Expertenwissen (E8) und sind im Allgemeinen bewährte praktische Vorgehensweisen (E11).

Komponenten teilen nicht den Vorlagencharakter von Mustern (E1). Komponenten sind fertig programmierte Software für ein bestimmtes Problem und verändern sich nicht oder nur gering in ihrer Anwendung (E7).

	Muster	Strategien, Techniken	Algorithmen	Referenzmodelle	Klassenbibliothek	Rahmenwerke	Komponenten
E1	+	-	-	-	+	-	-
E2	+	-	-	+	-	-	-
E3	+	+	-	+	-	-	-
E4	+	-	+/-	+	+	+	+
E5	+	+	+	+/-	+	+	+
E6	+	-	-	+	-	+	-
E7	+	-	-	-	-	+	-
E8	+	+	+	+	-	+	+
E9	+	-	+/-	+/-	-	-	-
E10	+	+	-	-	-	-	-
E11	+	+	-	+	+	-	+
E12	+	-	+	-	-	-	+/-
E13	+	-	+	-	-	-	+/-
E14	+	+	+/-	-	+	-	+
E15	+	+/-	+/-	+	-	+	+

\+ Ansatz unterstützt die entsprechende Eigenschaft.
\- Ansatz unterstützt die entsprechende Eigenschaft nicht.
+/- Ansatz unterstützt die entsprechende Eigenschaft teilweise.

Abbildung 8: Eine Abgrenzung von Mustern im Verständnis dieser Arbeit gegen andere Ansätze der Systementwicklung anhand der Eigenschaften E1 bis E15[1]

2.2.3 Ein Musterklassifikationsschema nach Aktivitäten der Systementwicklung

Die Verwendung des Begriffs Muster ist stark durch das Werk von GAMMA et al. geprägt, das sich insbesondere auf Muster für den Detailentwurf bezieht und dafür den Begriff Entwurfsmuster (engl.: Design Pattern) geprägt hat (vgl. [Gam+94], S. 2; [Bus+98], S. 410).

[1] Eine zusammenhängende Auflistung der Eigenschaften E1 bis E15 ist in Anhang 16 zu finden.

Analog dazu lässt sich das Musterkonzept auf weitere Aktivitäten der Systementwicklung ausdehnen. Eine erste grobe Klassifikation von Mustern ergibt sich in Anlehnung an die im Wasserfallmodell in Phasen erfassten Aktivitäten Anforderungsanalyse (Ak1) bis Implementierung (Ak6), sowie Wartung und Betrieb (Ak8). Diese Muster werden im Folgenden als phasenabhängige Muster bezeichnet, was die enge, jedoch keinesfalls vollständige Abhängigkeit ihrer Anwendung von den korrespondierenden Aktivitäten in den entsprechenden Phasen ausdrückt. Muster, die sich nicht direkt einer der Aktivitäten zuordnen lassen, werden als phasenunabhängig bezeichnet. Zu den phasenabhängigen Mustern gehören damit:

M1. Anforderungsanalysemuster,

M2. Analysemuster,

M3. Architekturmuster,

M4. Entwurfsmuster,

M5. Idiome (Implementierungsmuster),

M6. Testmuster und

M7. Wartungsmuster.

Zu den phasenunabhängigen Mustern der Systementwicklung zählen im Folgenden:

M8. Prozessmuster und

M9. Organisationsmuster.

Zwei weitere verwandte Musterkategorien, deren Anwendungsgebiet jedoch nicht in der Systementwicklung liegt, seien der Vollständigkeit halber hier mit aufgeführt:

M10. Lehrmuster (engl.: Educational Patterns) und

M11. Musterformulierungsmuster (engl.: Pattern Writing Patterns).

Obwohl es durchaus denkbar wäre, dass Muster für die Aktivitäten Einführung (Ak7), Dokumentation (Ak9) und Wiederverwendungsanalyse (Ak10) erstellt werden, gibt es zum Zeitpunkt der Ausarbeitung dieser Arbeit keine Hinweise auf die Entwicklung von Mustern auf diesen Gebieten. Anforderungsanalysemuster (M1) werden nur selten in der Fachliteratur diskutiert. Mit Ausnahme von Wartungsmustern, die auf Entwurfsmustern beruhen, gilt dies auch für den Bereich der Wartungsmuster.

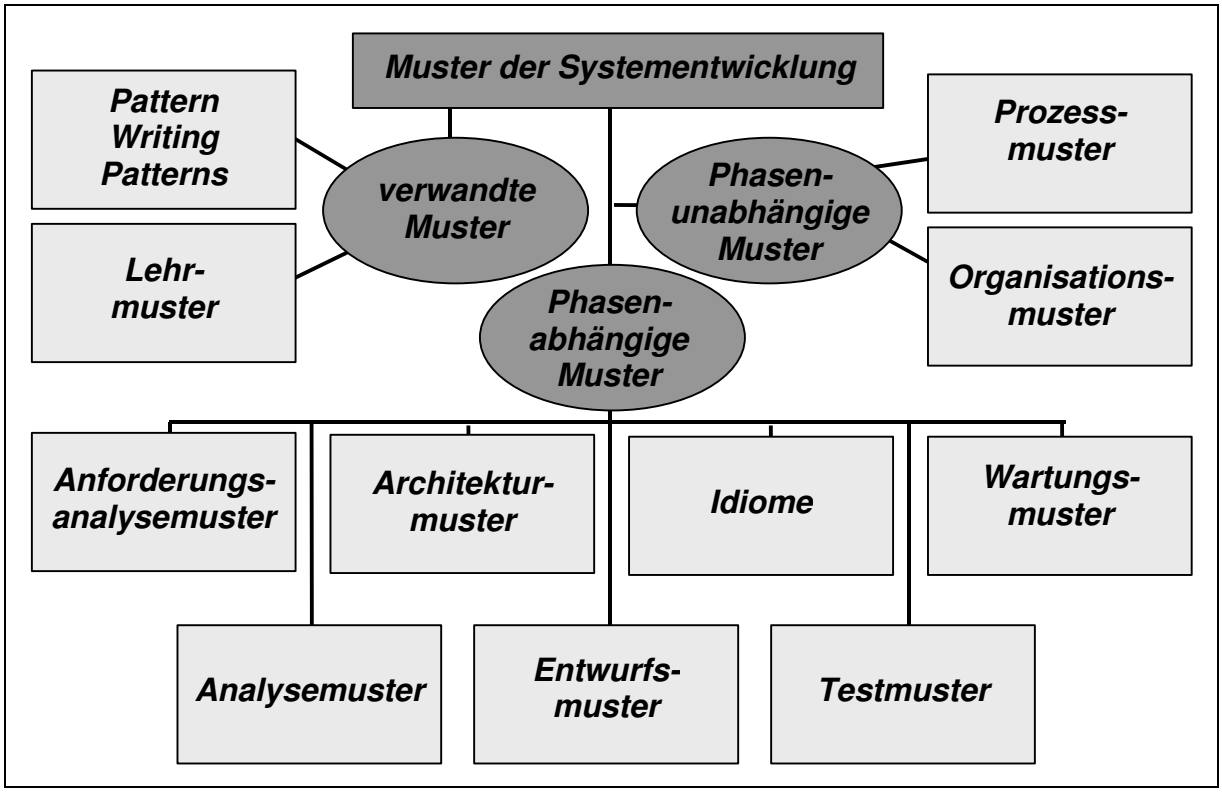

Abbildung 9: Eine Kategorisierung von Musterarten

Es folgt eine Übersicht über die einzelnen Arten von Mustern (M1 bis M11) und deren Anwendungsgebiet.

M1. Anforderungsanalysemuster

Beschreibung: Anforderungsanalysemuster sind Muster, die Techniken beschreiben, um funktionelle und nichtfunktionelle Eigenschaften von zu erstellenden Systemen zu erkennen und zu beschreiben (vgl. [Whit95], S. 259; [Cree01]).

Beispiel: Customer Rapport: „How do you build and establish a good relationship with a customer“ (vgl. [Whit95], S. 265).

M2. Analysemuster

Beschreibung: Analysemuster sind Konzepte, die allgemeine Konstrukte in der Analyse beschreiben. Dabei können sie sowohl in nur einem Anwendungsbereich als auch in mehreren Anwendungsbereichen verwendbar sein (vgl. [Fowl97], S. 8).

Beispiel: Organisation Hierarchies: Beschreibt die Abbildung von parallel existierenden Organisationshierarchien in einem objektorientierten Klassendiagramm (vgl. [Fowl97], S. 19ff).

M3. Architekturmuster

Beschreibung: Architekturmuster unterstützen die Architekturanalyse, indem sie fundamentale Strukturen eines Softwaresystems spezifizieren. Architekturmuster beschreiben Komponenten sowie deren Zuständigkeitsbereich und deren Zusammenwirken im Rahmen eines zu erstellenden Softwaresystems (vgl. [Mar+98], S. 143).

Beispiel: Layers: Beschreibt eine Art der Zerlegung von Anwendungen in Gruppen von Teilaufgaben, die auf unterschiedlichen Abstraktionsebenen liegen. Anwendungen höherer Ebenen basieren auf der Nutzung von Anwendungen niedrigerer Ebenen (vgl. [Bus+98], S. 32ff).

M4. Entwurfsmuster

Beschreibung: Ein Entwurfsmuster ist eine effektive Lösung zu wiederkehrenden Problemen der Entwurfsphase (vgl. [MoMa97], S. 6). Objektorientierte Entwurfsmuster beschreiben eine Konfiguration kommunizierender Objekte und Klassen, die ein allgemeines Entwurfsproblem in einem spezifischen Kontext lösen (vgl. [Gam+94], S. 3).

Beispiel: Observer: Beschreibt allgemein die Anordnung von Klassen und Objekten, so dass sogenannte Beobachter über Ereignisse an anderer Stelle informiert werden (vgl. [Gam+94], S. 293ff).

M5. Idiome (Implementierungsmuster)

Beschreibung: „Ein Idiom ist ein für eine bestimmte Programmiersprache spezifisches Muster auf einer niedrigen Abstraktionsebene. Es beschreibt, wie man spezielle Aspekte von Komponenten oder deren Beziehungen mit den Mitteln der Programmiersprache implementieren kann“ ([Bus+98], S. 14).

Beispiel: Virtual Constructor: Beschreibt die Erstellung von Objekten eines bekannten abstrakten Typs, jedoch ohne Kenntnis des konkreten Typs in der Programmiersprache C++ (vgl. [Copl01]).

M6. Testmuster

Beschreibung: Testmuster sind Muster, die sich mit Problemen des Tests (Ak6) beschäftigen. BINDER benutzt den Begriff Testentwurfsmuster (engl.: Test Design Patterns) und hebt damit den Entwurfscharakter von Testaktivitäten hervor (vgl. [Bind00], S. 338). Folgende Frage beschreibt das Ziel von Testmustern: "Given some implementation, what is an effective test design and execution strategy?" ([Bind00], S. 334).

Für MCGREGOR sind objektorientierte Testmuster mit objektorientierten Entwurfsmustern verwandt, insofern sie Strukturen von Objekten beschreiben, die benötigt werden, um sowohl Klassen isoliert als auch in Interaktion mit anderen Klassen zu testen. Dabei korrespondieren aus Entwurfsmustern abgeleitete Klassen mit den entsprechenden Klassen des Testmusters (vgl. [McGr99], S. 14).

Beispiel: ObserverTest Pattern: Dieses Muster ermöglicht es, die Implementierung des Entwurfsmusters Observer zu testen (vgl. [McGr99], S. 17ff; [Gam+94], S. 293ff).

M7. Wartungsmuster

Beschreibung: Wartungsmuster beschreiben Vorgehensweisen und Techniken für typische Wartungsaktivitäten wie Stabilisierung bzw. Optimierung (vgl. [Dew+99], S. 5ff). Da Wartungsaktivitäten zusätzliche Transformationszyklen auslösen können, lassen sich hier auch Adaptionen spezifischer Entwurfsmuster wiederfinden, insbesondere in Verbindung mit dem Reengineering von existierenden Softwaresystemen (vgl. [Ocal98], [Chu+99], [Kel+99]).

Beispiel: Consolidate the Program to Support Evolution and Reuse: Beschreibt ein Vorgehen, dass die Programmstabilisierung unterstützt (vgl. [FoOp95], S. 245f).

M8. Prozessmuster

Beschreibung: Prozessmuster beschreiben generelle Techniken, Aktivitäten und Aufgaben für die Entwicklung von Softwaresystemen (vgl. [Ambl98], S. 1). Dabei ergänzen sich Prozess- und Organisationsmuster in der gleichen Analogie, in der sich in der Organisationslehre Aufbau- und Ablauforganisation ergänzen. Die Unterscheidung zwischen phasenabhängigen Mustern und Prozessmustern, deren Anwendung sich auf eine einzige Aktivität beschränkt, ist mitunter schwierig vorzunehmen und erlaubt somit einen gewissen Interpretationsspielraum. Ähnliches gilt auch für die Abgrenzung von Organisationsmustern (vgl. [Copl96c]).

Beispiel: The Selfish Class: Beschreibt psychologische Probleme bei der Wiederverwendung von Komponenten und wie diese überwunden werden können (vgl. [FoYo98], S. 453ff).

M9. Organisationsmuster

Beschreibung: „Organizational Patterns describe the structure and practices of human organizations“ ([Bell01]). Organisationsmuster beschreiben damit allgemeine Management-techniken und organisatorische Strukturen, die sich für die Softwareentwicklung eignen (vgl. [Ambl98], S. 1).

Beispiel: Design by Team: Beschreibt die Aspekte, die bei der Bildung von Teams beachtet werden müssen. Dabei werden ausgehend von diesem Muster weitere verwandte Muster angewandt (vgl. [Harr96], S. 345f).

M10. Lehrmuster

Beschreibung: Lehrmuster beschreiben erprobte Techniken und Vorgehensweisen, die sich in der Vermittlung von technischem Wissen bewährt haben (vgl. [Anth96], S. 391f; [Keri99]).

Beispiel: Iterative Course Development: Beschreibt den Nutzen der iterativen Verbesserung von Kursinhalten in der Lehre (vgl. [Anth96], S. 392f).

M11. Musterformulierungsmuster

Beschreibung: Musterformulierungsmuster beschreiben typische Probleme bei der Ausarbeitung von Mustern. Dabei beziehen sie sich sowohl auf Sprachstil, die Strukturierung von Musterbeschreibungen als auch auf die Namensgebung für neue Muster (vgl. [MeDo98], [Mar+98], S.527f).

Beispiel: Code Samples: Beschreibt die Nutzung von Programmbeispielen für die Beschreibung von Mustern (vgl. [MeDo98]).

Wie ordnen sich diese Muster nun in die Abstraktions-Transformations-Sichtweise des Prozesses der Systementwicklung ein? Da M1 bis M6 direkt mit den Phasen des Wasserfallmodells korrespondieren, ergibt sich deren Position in der Abstraktions-Transformations-Matrix in Analogie zum Wasserfallmodell. Zu beachten ist, dass sowohl Abstraktionsgrad als auch Transformationsgrad ihre Bedeutung ändern, insofern beide für die Einordnung der Muster vom konkreten Problem-Lösungs-Paar verallgemeinernd betrachtet werden müssen. So weisen z.B. Idiome einen relativ geringen Abstraktionsgrad auf, insofern sie programmiersprachenabhängige, konkrete Vorgehensweisen beschreiben. Gleichzeitig werden sie am Ende des Transformationszyklus eingesetzt und befinden sich damit relativ nah an konkreten Lösungssystemen. Wartungsmuster (M7) liegen außerhalb des Transformationszyklus und beschreiben Probleme und Lösungen, die typischerweise in der Phase Wartung und Betrieb auftreten. Prozessmuster (M8) und Organisationsmuster (M9) sind phasenunabhängige Muster und können deshalb meist unabhängig vom Transformationsgrad eingesetzt werden (vgl. [Reic00], S. 52). Der Abstraktionsgrad von Prozess- und Organisationsmustern kann unterschiedlich stark ausgeprägt sein.

Interessanterweise weist Reichardt in seiner Klassifikation von Mustern, entgegen der Einordnung in dieser Arbeit, sowohl Analysemustern als auch Idiomen einen hohen Abstraktionsgrad zu, obwohl für beide gilt, dass „der Bezug zur Realwelt (Anm. d. Autors: Nicht der Diskurswelt, dem Problemsystem bzw. dem Lösungssystem!) oder einer Umsetzung in der Welt der Informationstechnologie“ ([Reic00], S. 39) leicht nachvollziehbar ist, was sogar nach der Klassifikation Reichardts ein Grund wäre, diesen beiden Typen einen geringen Abstraktionsgrad zuzuweisen.

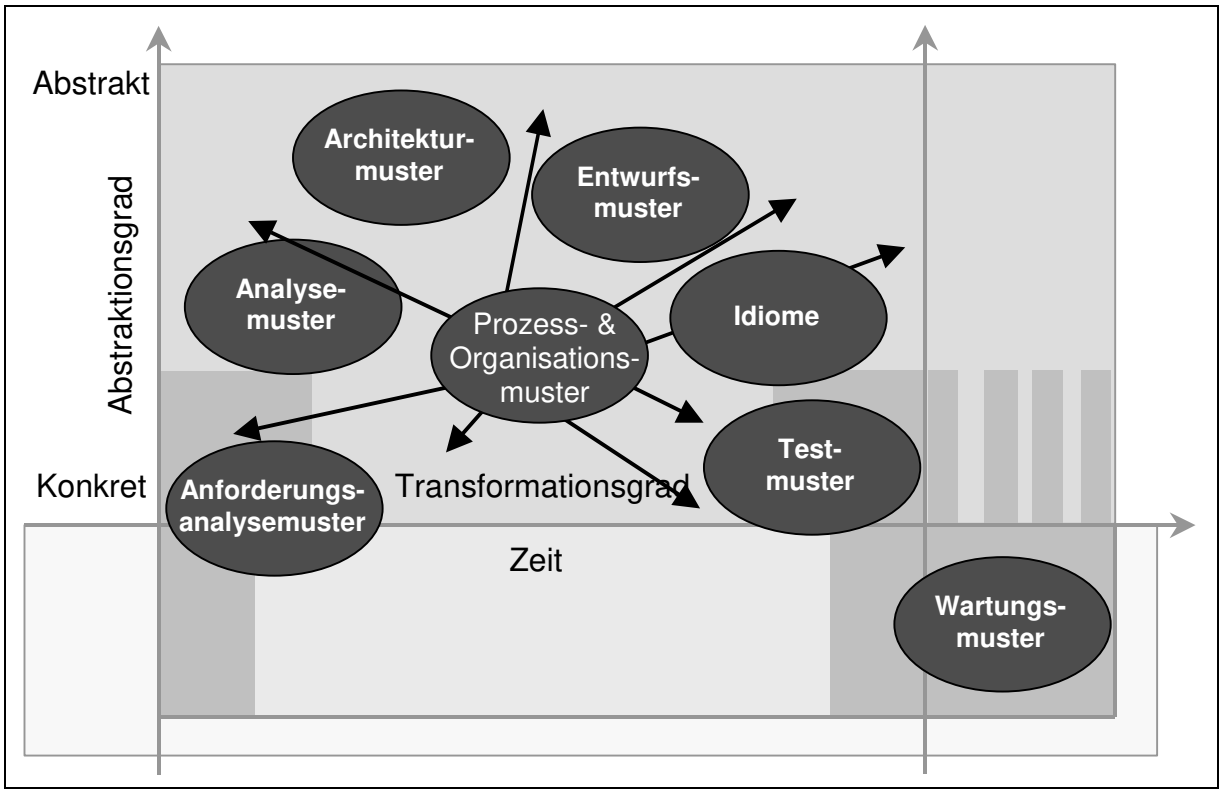

Abbildung 10: Die Einordnung der verschiedenen Musterarten in den allgemeinen Prozess der Systementwicklung

Der Übersicht halber seien hier noch einige weitere Arten von Mustern aufgeführt, die jedoch alle als Untergruppen der oben aufgeführten Musterklassifikation aufgefasst werden können. Die von HAY vorgeschlagenen Datenmodellmuster (engl.: Data Model Patterns) sind den Analysemustern zuzuordnen (vgl. [Hay95]), insofern sie typische Aktivitäten der Analyse beschreiben (vgl. Abschnitt 1.2.1). Obwohl es durchaus auch möglich ist zu argumentieren, dass HAYS Datenmodellmuster Entwurfsmuster sind, spricht dagegen, dass diese Muster, wenngleich für relationale Datenbanken gedacht, aufgrund ihrer Allgemeinheit durchaus auch z.B. unter Nutzung objektorientierter Techniken benutzt werden können (vgl. [Fowl97], S.5). Damit wären HAYS Datenmodellmuster unabhängig von der Implementierung. Diese Eigenschaft ist eine der Hauptcharakteristika von Analyseaktivitäten bzw. Analysemustern.

Die von GAMMA et al. vorgeschlagenen Performance-Tuning Patterns[1] lassen sich der Optimierung zuordnen (vgl. [Gam+94], S. 353). Die Optimierung ist eine typische Aktivität

[1] In Ermangelung einer sinnerhaltenden deutschen Übersetzung wird hier der englische Begriff beibehalten.

der Wartung (vgl. Abschnitt 1.2.1), womit Performance-Tuning Patterns als Wartungsmuster anzusehen sind. Weiterhin werden Benutzerschnittstellenmuster (engl.: User Interface Pattern) unterschieden (vgl. [Gam+94], S. 353; [BrFl98]; [MaJo98]). Da der Entwurf der Benutzerschnittstelle eine Aufgabe des Detailentwurfs ist (vgl. Abschnitt 1.2.1), zählen Benutzerschnittstellenmuster zu den Entwurfsmustern.

2.3 Implikationen für die empirische Studie

In diesem Abschnitt werden die Analysegebiete für die empirische Studie aufgestellt, die sich aus Abschnitt 2 ergeben.

Die in Abschnitt 2.1 und 2.2.1 aufgeführten Charakteristika von Mustern (E1 bis E15) bieten die Grundlage für eine Analyse darüber, welche dieser Eigenschaften den Begriff Muster, wie er in der Systementwicklung verwendet wird, am besten beschreiben. Daraus lässt sich dann eine Eingrenzung des Musterbegriffs, wie er in der Praxis angewendet wird, erstellen (Frage 1A).

Eine weitere Frage ergibt sich aus der Sichtweise auf Algorithmen. Werden diese als Muster gesehen, so deutet dies im Allgemeinen auf ein eher weitgefasstes Musterverständnis. Zu analysieren ist, ob dieses weitgefasste Musterverständnis einen Einfluss auf die Charakterisierung des Musterbegriffs hat (Frage 1B).

Während sich die Frage nach den durchgeführten Aktivitäten der Systementwicklung schon aus Abschnitt 1 ergibt, leitet sich die Frage nach der Nutzungshäufigkeit und dem Bekanntheitsgrad der einzelnen Musterarten aus der Unterstützung der einzelnen Aktivitäten durch Muster ab (Frage 3A, 3B).

Die Nutzung phasenunabhängiger Muster sowie von Musterformulierungsmustern und Lehrmustern steht keiner der Aktivitäten der Systementwicklung gegenüber und lässt sich deshalb im Gegensatz zu phasenabhängigen Mustern auch nicht als Anteil von Musternutzern für eine bestimmte Aktivität darstellen. Zu beachten ist, dass aufgrund der Beschränkung der Umfrage auf Musternutzer nur Aussagen zum Verhältnis des Nutzungsgrades der verschiedenen Musterarten, nicht jedoch zu einer generellen Verbreitung der Nutzung von Mustern in der Softwareentwicklung gemacht werden können. Die Nutzung von Mustern wird noch einmal in die persönliche Anwendung und die Anwendung innerhalb der Organisation

unterschieden. Die Anwendung innerhalb der Organisation schließt im Gegensatz zur persönlichen Anwendung die Nutzung von Mustern in der Kommunikation mit anderen Entwicklern, z.B. in Meetings oder in der Softwaredokumentation, ein.

Der Bekanntheitsgrad mit den einzelnen Musterarten wird anhand der Anzahl vertrauter Muster gemessen. Neben der Anwendung der verschieden Musterarten ergibt sich aus dem Bekanntheitsgrad ein zusätzliche Aussage über das Verhältnis der Ausbreitung der verschiedenen Musterarten.

Da sich in der Literatur keine Muster für die Aktivitäten Wiederverwendung, Dokumentation und Einführung finden ließen, ergibt sich die Frage, ob Muster für diese Aktivitäten existieren und ob es den Befragten überhaupt sinnvoll erscheint, Muster auf diesen Gebieten zu entwickeln (Frage 6C).

3. Ein musterbasierter Systementwicklungsprozess

3.1 Muster in den Phasen der Softwareentwicklung

3.1.1 Ein allgemeiner Mikroprozess für Muster

Genauso wie die Erfahrung von Experten die Anwendung von Methoden bzw. deren Vorgehensmodellen in größeren Projekten der Systementwicklung erleichtern bzw. verbessern, jedoch nicht ersetzen kann, kann die Anwendung von Mustern, als ein Ansatz zur Wiederverwendung von Expertenwissen (E8), existierende Methoden nur ergänzen (vgl. [Schm01], S. 9).

Dabei entspricht der Einsatz von Mustern im Prozess der Systementwicklung neben anderen Ansätzen, wie z.B. dem Einsatz von Komponenten, Klassenbibliotheken und Rahmenwerken, dem Konzept der Wiederverwendung (K4). Alle Ansätze lassen sich in bestehende Vorgehensmodelle, wie im Folgenden am Beispiel des Wasserfallmodells demonstriert, integrieren. Allerdings zielt der Einsatz von Mustern im Gegensatz zu anderen Wiederverwendungsansätzen auf die Wiederverwendung von Wissen (E8) über bewährte praktische Vorgehensweisen (E11) anstatt von ausprogrammierten Lösungen ab.

In Anlehnung an BOOCH wird hier deshalb zwischen Makroprozess, der den generellen Prozess der Systementwicklung im Groben beschreibt, und Mikroprozess als Repräsentation

der täglichen Aktivitäten individueller Entwickler bzw. eines kleinen Teams von Entwicklern unterschieden (vgl. [Booc94], S. 234ff). Im Folgenden wird zunächst der Mikroprozess für Muster entwickelt, um dann im Anschluss die Integration von Makroprozess und Mikroprozess zu analysieren (vgl. [Bus+98], S. 22f).

Ein Mikroprozess für Muster muss zunächst zwischen den beiden Rollen Konsument und Produzent von Mustern unterscheiden. Dabei müssen Konsument und Produzent nicht notwendigerweise unterschiedliche Personen sein (vgl. [Meye94], S. XI).

Der Mikroprozess für Musterkonsumenten

Hauptaufgabe des Mikroprozesses für Musterkonsumenten ist die Unterstützung bestehender Aktivitäten des Makroprozesses (z.B. Ak1 bis Ak9) durch die Auswahl passender Problem-Lösungs-Paare von Mustern, wobei ausgehend von der Problemspezifikation ein passendes Muster gesucht wird, was dann auf das erkannte Problem angewandt wird (vgl. [YaDo00], S. 369f; [Bus+98], S. 367ff). Damit ergibt sich folgender Mikroprozess für Musterkonsumenten:

MK1. (Mikro-)Problemspezifikation,

MK2. Suche nach anwendbaren Mustern,

MK3. Auswahl der besten Lösung und

MK4. Musteranwendung.

„Many non-experts simply don’t know what problems they have“ ([Keri00]). Häufig adressieren Muster Probleme, die nicht direkt aus der Analyse des bestehenden Sachverhalts erkennbar sind. Dazu zählen insbesondere nichtfunktionelle Eigenschaften von Softwaresystemen, wie z.B. die Wartbarkeit. Nicht zuletzt deshalb ist es sinnvoll, die Suche nach Mustern problemunabhängig durchzuführen. Dies bedeutet, dass die Problemspezifikation (MK1) des Mikroprozesses für Musterkonsumenten entfällt. Dadurch kann sichergestellt werden, dass der zusätzliche Informationsgehalt, der durch die

Anwendung von Mustern erreicht wird, in den Ergebnissen der umgebenden Aktivität dokumentiert ist.[1]

Die Anwendung eines Musters kann durchaus die Umsetzung eines weiteren Musters verlangen (vgl. [Nobl98a], S. 5). Dies bedeutet, dass nach der Anwendung des Musters weitere Zyklen des Mikroprozesses durchlaufen werden müssen. Da nicht sämtliche Probleme durch Muster erfasst sind, kann es auch geschehen, dass Phase 2, die Suche nach anwendbaren Mustern, erfolglos endet. Dies kann entweder bedeuten, dass dies ein Kandidat für ein zu entwickelndes Muster ist, oder aber dieses Problem nicht oder nur schwer durch Muster beschreibbar ist. Der erste Fall initiiert den Mikroprozess für Musterproduzenten.

Der Mikroprozess für Musterproduzenten

Es gibt viele Möglichkeiten Muster zu entwickeln. Meist sind diese Verfahren losgelöst vom aktuellen Prozess der Systementwicklung und analysieren entweder existierende Softwaresysteme, insbesondere für Entwurfsmuster oder versuchen, das Wissen von Experten durch Interviews, Workshops oder ähnliche Ansätze zu extrahieren (vgl. [Risi01b]; [Henn91], S. 270ff). Im Rahmen der Einordnung von Mustern in den Wissenszyklus werden diese Ansätze in Abschnitt 4.3 noch einmal aufgegriffen. Im Folgenden wird kurz dargestellt, inwiefern die Produktion von Mustern in den Mikroprozess eingeordnet werden kann.

Da die Erstellung eines Musters einen nicht unerheblichen Aufwand bedeutet, kann dieser letztlich nur losgelöst von den Aktivitäten des aktuellen Systementwicklungsprozesses vonstatten gehen. Deshalb beschränkt sich der Mikroprozess für Musterproduzenten auf die Analyse potenzieller neuer Muster bzw. neuer Aspekte bestehender Muster. Die konkrete Umsetzung in eine Musterbeschreibung obliegt dann einem Prozess, der außerhalb des Systementwicklungsprozesses liegen muss. Der Mikroprozess für Musterproduzenten besteht aus drei Phasen, wobei Phase MP1 und MP2 iterativ durchlaufen werden:

MP1. Musteridee fixieren

[1] Neben der Suche nach anwendbaren Mustern begleitend zur durchzuführenden Aktivität ist es auch denkbar, dass diese Suche auch für komplette Quellkodes durchgeführt wird. Dies kann typischerweise im Rahmen des Refactorings geschehen und wird kurz in Abschnitt 3.2 unter dem Aspekt der automatische Quellkodegenerierung diskutiert.

MP2. Musteridee bewerten

MP3. Musteridee umsetzen.

„Pattern explicitly capture knowledge that experienced developers already understand implicitly“ ([CoSc95], S. 70). Dieses implizite Verstehen von Problem-Lösungs-Paaren begründet die Schwierigkeit neue Muster zu erkennen und zu fixieren. VLISSIDES schlägt daher vor, Erfahrungen aus der Entwicklung von Softwaresystemen inkrementell aufzuzeichnen (vgl. [Vlis01], S. 7). Dies schlägt sich in den Phasen 1 und 2 des Mikroprozesses für Musterproduzenten nieder. Musterideen werden parallel zu sonstigen Aktivitäten der Systementwicklung entwickelt. Dabei wird ausgehend von einer initialen Idee diese Idee soweit inkrementell verfeinert, bis sie in eine erste Version in Form eines Musters umgesetzt werden kann.

Die Bewertung von Musterideen sucht nach bestehenden Mustern, die der Musteridee ähneln, verwirft Musterideen, die zu trivial sind oder aber sucht auch nach anderen Anwendungen dieses Musters.[1] Dabei muss eine Bewertung auch die Evolution, also sogar die Möglichkeit, dass existierende Muster veralten, berücksichtigen (vgl. [Bus+98], S. 376f). Die Umsetzungsphase (MP3) initiiert den Lebenszyklus des Musters (vgl. [Anti01]). In ihr wird, jedoch unabhängig vom aktuellen Systementwicklungsprozess, das Muster in eine wiederverwendbare Form gebracht (vgl. [MeDo98]).

Sowohl aus Konsumenten- als auch aus Produzentensicht handelt es sich um modifizierende Wiederverwendung (vgl. [Ott91], S. 75f). Während die Eigenschaft der Änderung eines generellen Schemas bei der Anwendung von Mustern (E7) diese Einordnung für die Konsumentensicht begründet, ergibt sich dies in der Produzentensicht daraus, dass Muster Verallgemeinerungen des ursprünglichen Problems darstellen.

Integration des Mikroprozesses in den Makroprozess

Letztlich kann die Anwendung des Mikroprozesses für Muster nur eine zusätzliche Hilfe für die kreativen Aktivitäten des Makroprozesses sein (vgl. [Bus+98], S. 419f). Eine der

[1] Eine oft zitierte Forderung an Muster ist die "Rule of Three", die letztlich auf der Eigenschaft der mehrfachen praktischen Anwendung von Mustern (E11) basiert.

wichtigen Fragen ist, zu welchem Zeitpunkt sowohl der Mikroprozess aus Konsumentensicht als auch der Mikroprozess aus Produzentensicht begonnen wird.

Während die Forderung für die Produzentensicht darin besteht, Musterideen praktisch sofort zu beschreiben, können die nachfolgenden Phasen (MP2, MP3) durchaus auf spätere Zeitpunkte verschoben werden und beispielsweise am Ende einer Phase des Makroprozesses geschlossen für alle Musterideen durchgeführt werden.

Für die Anwendung des Mikroprozesses aus Konsumentensicht gilt, dass problemverursachte Mikroprozesse, also die, die mit Phase MK1 beginnen, sofort mit dem Erkennen des Problems ausgeführt werden können. Mikroprozesse aus Konsumentensicht, die direkt mit der Suche nach anwendbaren Mustern beginnen (MK2), könnten beispielsweise gesammelt am Ende einer Phase des Makroprozesses durchgeführt werden. Dies garantiert, dass insbesondere Experten, die ihnen bekannte Muster implizit anwenden, nicht unnötig durch die Forderung nach der Dokumentation von Mustern in den Ergebnissen der Phase des Makroprozesses gestört werden.

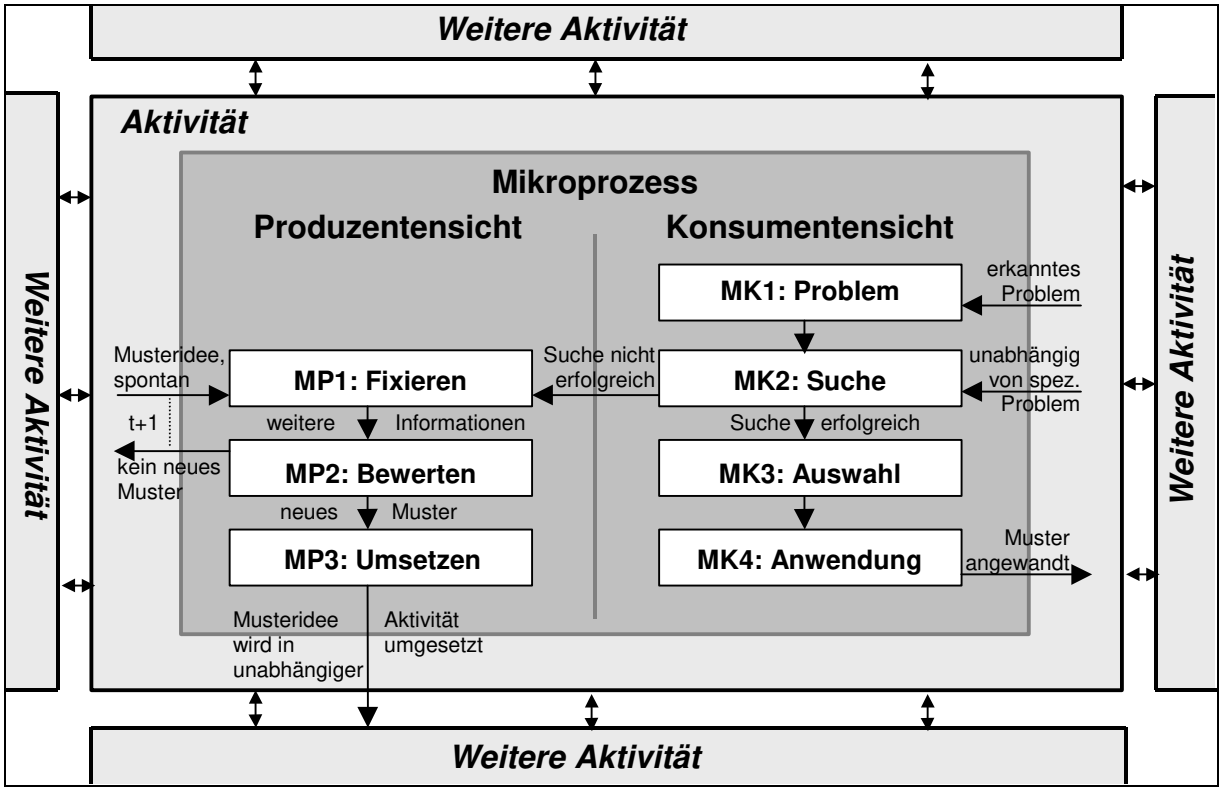

Abbildung 11: Der Mikroprozess für Muster

Durch die Allgemeinheit des Mikroprozesses für Muster ergibt sich, dass der Mikroprozess prinzipiell in alle Aktivitäten der Systementwicklung integrierbar ist. Damit lässt sich die Nutzung und die Erstellung von Mustern in die Aktivitäten des genutzten Vorgehensmodells integrieren.

Obwohl der Mikroprozess durchaus als offensichtlich bezeichnet werden kann, so bietet doch gerade seine Einfachheit einen Einstieg für die Analyse weitere Ansätze in der Musterliteratur, insbesondere auch mit Hinblick auf die Automatisierung bzw. technische Unterstützung des Mikroprozesses für Muster – die Organisation von Mustern in Musterorganisationsschemata (Abschnitt 3.1.2), die automatische Mustersuche (Abschnitt 3.1.3) und eine automatische Musterimplementierung (Abschnitt 3.1.3).

3.1.2 Die Charakterisierung von Mustern

Muster werden in Musterbeschreibungsformen und Musterorganisationsschemata charakterisiert. Während Musterbeschreibungsformen dazu dienen, bestimmte Aspekte spezifischer Muster zu charakterisieren, dienen Musterorganisationsschemata dazu, Muster anhand dieser Aspekte zueinander in Beziehung zu setzen. Um spezifische Blickwinkel auf Musterorganisationsschemata hervorzuheben, wurden Begriffe wie Musterkataloge, Mustersysteme und Mustersprachen geprägt.

Musterbeschreibung

Zunächst dient die Beschreibung dazu, ein spezifisches Muster abzugrenzen. Daher ist die Beschreibung von Mustern für die Bewertung neuer Musterideen (MP2) von extremer Wichtigkeit. Daneben vermittelt die Beschreibung von Mustern denjenigen, die mit den Mustern nicht vertraut sind, eine Möglichkeit, diese zu erlernen, und denjenigen, die mit ihnen vertraut sind, eine Referenz, um beispielsweise Details der Lösung nachzuschlagen. Damit ist eine gute Musterbeschreibung Grundlage für die Auswahl der besten Lösung (MK3) sowie für die Umsetzung des Musters im konkreten Systementwicklungsprozess (MK4).

Aufbauend auf dem Problem-Lösungs-Paar, das ein Muster charakterisiert (E12), wurden verschiedene Musterbeschreibungsformen entwickelt. Dabei werden Muster entweder informell in fließendem Text beschrieben oder aber folgen einem formalen Schema, dass verschiedene Aspekte von Mustern hervorhebt. Die Portland Form und die informelle

Beschreibungsform nach ALEXANDER sind typische Beispiele für informelle Beschreibungsformen, wohingegen die Musterform nach GAMMA und die kanonische Form typische Beispiele für die Nutzung formeller Beschreibungsformen sind (vgl. [Cunn99]; [Gam+94], S. 6f; [Bus+98], S. 19ff). Allen Beschreibungen gemeinsam ist, dass sie versuchen, dem Muster einen eindeutigen Namen zu geben, der sowohl das Muster identifiziert, als auch einen ersten Hinweis auf den Inhalt des Musters gibt. Da der Name als Referenz für den Inhalt des Musters dient, ist die Auswahl eines guten Namens ein wichtiger Bestandteil der Musterbeschreibung (vgl. [Schm01], S. 6). Häufig werden bei der Namensgebung Metapher aus dem allgemeinen Sprachgebrauch verwendet, z.B. Brücke, Besucher, u.Ä. (vgl. [Gam+94]).

Beide Beschreibungsformen haben ihre Vor- und Nachteile. So erleichtert die informelle Beschreibungsform die Beschreibung von Mustern und erlaubt mehr Freiheiten in der Gestaltung des Gedankenflusses zu Gunsten besserer Lesbarkeit. Hingegen erleichtert die formale Beschreibungsform durch seine Einheitlichkeit den Vergleich von Mustern, die schnelle Aneignung des Inhalts von Mustern sowie die Nutzung der Musterbeschreibung als Referenz. Die Darstellung von Mustern in formellen Beschreibungsformen ermöglicht im Gegensatz zu informellen Beschreibungsformen die Anwendung einiger der im Abschnitt 3.1.3 beschriebenen Ansätze zur Automatisierung der Musternutzung, insbesondere der maschinellen Musterbeschreibung (vgl. [Ede+97], S. 44f).

Aufgrund ihres allgemeinen Charakters haben informelle Beschreibungsformen den Vorteil, dass sie sich ohne Probleme an unterschiedliche Domänen bzw. Wissensgebiete anpassen. Hingegen müssen formelle Beschreibungssprachen im Allgemeinen an die Domäne angepasst werden. So lässt sich die in [Gam+94] genutzte formelle Beschreibungsform für Entwurfsmuster aufgrund ihrer spezifischen Forderung nach Beispielkode nicht notwendigerweise auf die Beschreibung anderer Musterarten, wie z.B. Analysemuster, anwenden.

Musterorganisationsschemata

Eine Organisation von Mustern in einem Schema erleichtert die Suche nach einem bestimmten Muster (MK2) anhand bestimmter Aspekte (vgl. [Bus+98], S. 16), die dieses Muster charakterisieren. Derartige Schemata ermöglichen es auch, neue Musterideen und

existierende ähnliche Muster gegeneinander abzuwägen (MP2), ohne eine neue Idee mit sämtlichen existierenden Mustern vergleichen zu müssen.

Ein erster Ansatz für die Organisation von Mustern in einem Schema ist die Klassifikation von Mustern anhand ihrer spezifischen Eigenschaften. Eine gute Klassifikation ermöglicht es, durch einen Vergleich von gewünschten Eigenschaften mit den Eigenschaften existierender Muster schnell und möglichst zielsicher die Muster zu finden, die den gewünschten Eigenschaften entsprechen. Die potenziellen Eigenschaften eines Musters werden dabei in der Literatur zumeist auf wenige Kategorien beschränkt.

So klassifizieren GAMMA et al. ihre Entwurfsmuster beispielsweise anhand des Anwendungsgebietes (engl.: Scope), mit den Ausprägungen Klasse und Objekt sowie anhand des Zwecks (engl.: Purpose), mit den Ausprägungen Erstellung, Struktur und Verhalten (vgl. [Gam+94], S. 9ff). Hingegen klassifizieren BUSCHMANN et al. Muster anhand ihrer Art sowie anhand von mehreren Problemkategorien (vgl. [Bus+98], S. 362). BUSCHMANN et al. unterscheiden dabei zunächst drei Arten: Architekturmuster, Entwurfsmuster und Idiome. Insbesondere die Klassifikation in Problemkategorien erleichtert das Auffinden von Mustern anhand einer Problemspezifikation, wie sie auch in der ersten Phase des Mikroprozesses für Musterkonsumenten erarbeitet wurde (MK1).

Prinzipiell ist der Anzahl von Kategorien für die Klassifikation von Mustern keine Grenze gesetzt. So sind durchaus weitere Kategorien vorstellbar (vgl. [BuMe95]). Idealerweise lassen sich dabei alle einzuordnenden Muster eindeutig einer einzigen Kategorie zuordnen, was jedoch in der Realität nicht immer möglich ist (vgl. [BuMe95], S. 340).

Die oben beschriebene Vorgehensweise für die Klassifikation von Mustern wird im Allgemeinen als Musterkatalog bezeichnet (vgl. [Gam+94], S. 9ff). Der Begriff Mustersystem hingegen zielt auf einen weiteren in Katalogen nicht berücksichtigten Aspekt von Mustern ab. Ein System wird dabei durch seine Komponenten und den Beziehungen zwischen diesen Komponenten charakterisiert (vgl. [Habe97], S. 5). Dabei sind die Komponenten Muster eines eingegrenzten Anwendungsgebietes bzw. Problembereichs. Die Beziehungen zwischen den Mustern werden dann analysiert. Mehrere Muster, die durch zahlreiche Beziehungen miteinander verbunden sind und mit anderen Mustern weniger stark verbunden sind, werden dabei zu Gruppen (engl.: Cluster) zusammengefasst.

Während GAMMA et al. diese Beziehungen praktisch nicht klassifizieren und stattdessen, aufbauend auf dem Inhalt des Abschnittes „Verwandte Muster“ der Musterbeschreibung, eine natürlichsprachliche Beschreibung der Beziehungen benutzen (vgl. [Gam+94], S. 11ff), lassen sich in der Literatur auch Klassifikationen für Beziehungen finden. Der zusätzliche Informationsgehalt durch die Klassifizierung von Beziehungen lässt beispielsweise einen zusätzlichen Grad an Automatisierung zu (vgl. Abschnitt 3.1.3). Dabei lassen sich in der Literatur unterschiedliche Klassifikationen für Musterbeziehungen finden. Beispielhaft seien hier drei aufgeführt:

Musterbeziehungen nach BUSCHMANN (vgl. [Bus+98], S. 361)

- Muster X wird von Muster Y verfeinert.
- Muster X kann mit Muster Y kombiniert werden.
- Muster X ist eine Alternative zu Muster Y.

Musterbeziehungen nach ZIMMER (vgl. [Zimm95], S. 349)

- Muster X benutzt Muster Y in seiner Lösung.
- Eine Variante von Muster X benutzt Muster Y in seiner Lösung.
- Muster X gleicht Muster Y.

Musterbeziehungen nach NOBLE (vgl. [Nobl98b], S. 99ff)[1]

- Primäre Beziehungen
  - Muster X benutzt Muster Y.
  - Muster X verfeinert Muster Y.
  - Muster X löst das gleiche Problem wie Muster Y auf andere Weise.

[1] NOBLE unterscheidet zwischen primären und sekundären Beziehungen. „We classify ... relationships as secondary relationships because they are used much less frequently then the primary relationships ... “ ([Nobl98b], S. 102). NOBLE argumentiert, dass sich sämtliche sekundären Beziehungen aus primären Beziehungen zusammensetzen lassen. Sekundäre Beziehungen (Muster X wird von Muster Y benutzt), die direkt aus der Inversion primärer Beziehungen (Muster Y benutzt Muster Y) entstehen, sind hier ausgelassen.

- Sekundäre Beziehungen
    - Muster X ist eine Variante von Muster Y.
    - Eine Variante von Muster X benutzt Muster Y.
    - Muster X gleicht Muster Y.
    - Muster X und Muster Y lösen zusammen ein Problem.
    - Muster X erfordert die Lösung von Muster Y.
    - Muster X benutzt sich selbst.
    - Muster X, Y, Z werden sequenziell auf unterschiedlichen Detailniveaus genutzt.

In einer häufig anzutreffenden Beziehung ergänzen sich unterschiedliche Musterarten (M1 bis M7) im Prozess der Systementwicklung (vgl. [RaSr00], S. 365; [Jac+98], S. 137). So resultiert z.B. die Anwendung des Analysemusters "Organisation Hierarchies" oft in der Anwendung des Entwurfsmuster "Composite" (vgl. [RaSr00], S. 366). Diese Beziehung wird in Abschnitt 3.2 noch einmal aufgegriffen.

Ein weiterer Ansatz für die Organisation von Mustern in einem Schema wird durch den Begriff Mustersprache charakterisiert. Dabei spricht die Verwendung des Begriffs Sprache Ähnlichkeiten zwischen Mustersprachen und Programmiersprachen, die dabei als „eine beliebige Menge von Strings (Anm. d. Autors: String wird hier synonym für Satz bzw. Wort verwendet) über einem gegebenen Alphabet“ ([Aho+88], S. 113) gesehen werden, an. Dabei stellt eine Menge von Mustern sozusagen das Alphabet der Sprache dar. Während mit Programmiersprachen letztlich komplette Programme beschrieben werden, müssten mit Mustersprachen in Analogie z.B. komplette Modelle erzeugt werden können.

Einen Ansatz für eine Mustersprache für objektorientierte Programme beschreibt NOBLE (vgl. [Nobl98a]). Dabei wird, ausgehend von einem Urmuster "Objektorientiertes Programm," ein hierarchischer Graph mit Gruppen von Architekturmustern, Entwurfsmustern und Idiomen als Knoten durchlaufen, was letztlich in der Konstruktion eines Programms resultiert. Diese Gruppen bestehen dabei im Allgemeinen aus ein bis sechs Mustern zu einem spezifischen Themengebiet.

Obwohl der Begriff Mustersprache vielfach für irreführend gehalten wird (vgl. [RiZü96], S.10), wird er trotz alledem häufig verwendet. Mustersprachen werden im Allgemeinen für bestimmte Anwendungsgebiete beschrieben (vgl. [Nobl98a], S. 4), wobei sich, im Gegensatz zum Ansatz von NOBLE, zumeist sämtliche Muster der jeweiligen Sprache genau einer der Musterarten (M1 bis M11) zuordnen lassen. Beispiele sind:

- A Pattern Language for Computer-Integerated Manufacturing (vgl. [Aar+95]),
- Crossing Chasms. A Pattern Language for Object-RDBMS (vgl. [BrWh96]),
- Episodes: A Pattern Language of Competitive Development (vgl. [Cunn96]).

3.1.3 Potenzial für die Automatisierung im Mikroprozess für Muster

Hinsichtlich einer Automatisierung von Teilen des Mikroprozesses lassen sich momentan vier Ansätze unterscheiden, die jeweils verschiedene Aspekte des Mikroprozesses unterstützen. Im Folgenden seien diese Ansätze näher analysiert. Der Begriff Modell wird dabei weiterhin als Ergebnis eines Abstraktions- bzw. Konkretisierungsschrittes gesehen und lässt sich z.B. als Klassendiagramm oder aber als Quellkode veranschaulichen (vgl. Abschnitt 1.1).

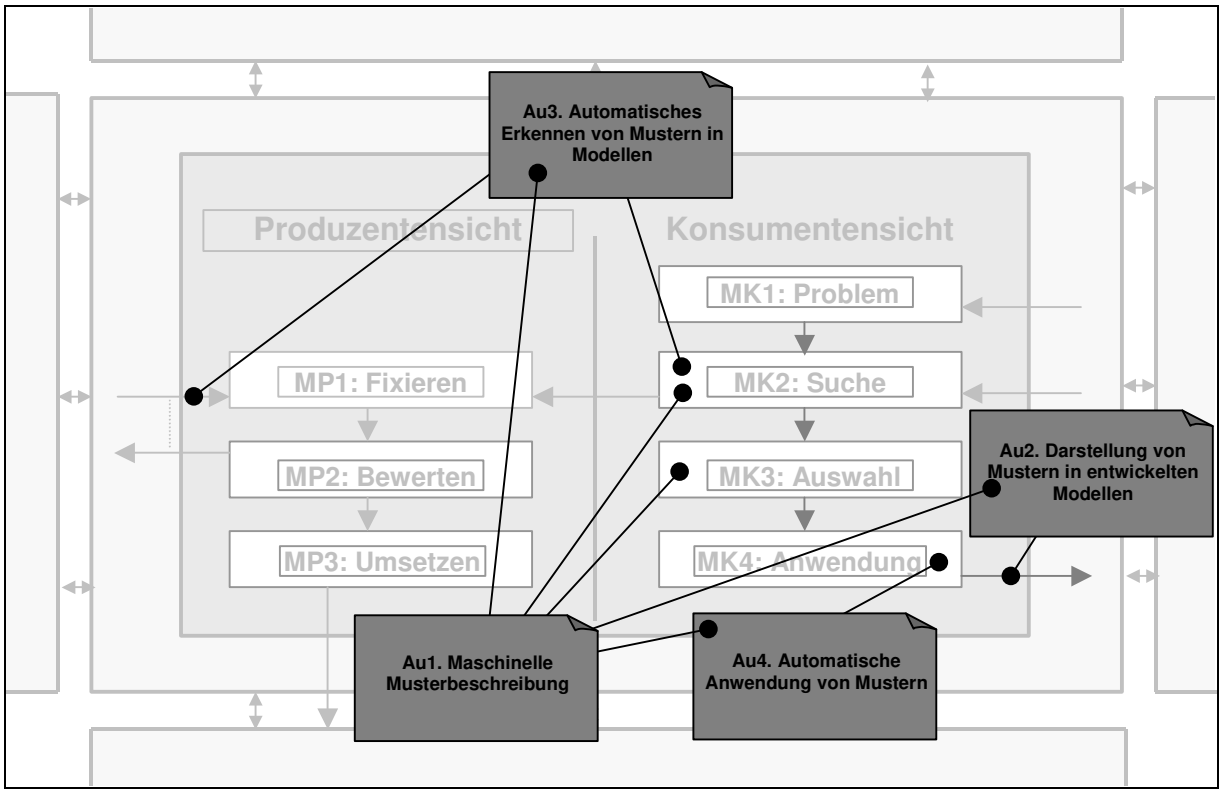

Abbildung 12: Einordnung der vier Ansätze zur Automatisierung von Mustern in den Mikroprozess (vgl. Abbildung 11)

Au1. Maschinelle Musterbeschreibung

Eine maschinelle Beschreibung von Mustern, also die Darstellung eines Musters in einer Sprache, die ein Computer verarbeiten kann, ist die Grundlage sowohl für das automatische Erkennen von Mustern (Au3) als auch für eine automatisierte Anwendung von Mustern (Au4). Maschinelle Musterbeschreibungen können anhand eines Musterorganisationsschemas ähnlich einer Datenbank in einem Pattern Repository gespeichert werden. Eine maschinenverständliche Form der Speicherung ermöglicht beispielsweise eine erweiterte Suchfunktionalität gegenüber einer herkömmlichen Beschreibung auf Papier. Eine solche Suchfunktionalität könnte dann z.B. die Bewertung neuer Musterideen (MP2) sowie die Suche nach anwendbaren Mustern (MK2) unterstützen.

Die Nutzung einer Auszeichnungssprache wie SGML oder XML ergibt sich zunächst aus der Idee der formalen Musterbeschreibung. Dabei wird der Aufbau der Beschreibung von Mustern in eine SGML/XML-Gliederung überführt. Für strukturorientierte objektorientierte Entwurfsmuster lässt sich deren Struktur sowie eine Instanziierungsvorschrift für diese Struktur, sogar mittels SGML/XML abbilden (vgl. [Oht+99]). Dabei ist es denkbar, dass sich durch die Aufnahme eines Klassifikationsschemas für Musterbeziehungen in die SGML/XML Typbeschreibung (DTD, XSD), eine typisierte Vernetzung von Mustern aufbauen lässt.

Ein weiterer Ansatz ist die Entwicklung spezieller Sprachen, die Muster beschreiben. Diese sind nicht zu verwechseln mit den Mustersprachen als Mittel zur Organisation von Mustern und deren Beziehungen zueinander (vgl. 3.1.2). Praktisch alle Ansätze beschränken sich dabei auf die Beschreibung der Struktur und/oder des Verhaltens von objektorientierten Entwurfsmustern. Da sich zumindest Analysemuster in ähnlicher Weise beschreiben lassen wie Entwurfsmuster, ist es denkbar, die maschinelle Beschreibung auch auf andere Musterarten auszudehnen. Die Beschreibung kann dabei entweder deklarativ oder imperativ erfolgen.

Die deklarative Beschreibung der Struktur von Mustern wird beispielsweise in [KrPr96] genutzt, wo die Beschreibung eines Musters in OMT-Notation in Prolog-Regeln übersetzt wird. Die meisten automatisierten Beschreibungssprachen sind jedoch imperativ. So wird in [Saek00] unter Nutzung der Sprache LOTOS (vgl. [BoBr01]) beispielsweise das Verhalten von Entwurfsmustern formal beschrieben. Daneben wurden zahlreiche imperative Sprachen

für die Beschreibung der Struktur von Entwurfsmustern entwickelt (vgl. [Meij96], [Bünn99]). Die Sprache PaL beispielsweise basiert auf der Programmiersprache Eiffel und stellt u.a. zusätzliche Konstrukte für die Komposition und die Verfeinerung von Mustern zur Verfügung (vgl. [Bünn99]).

Au2. Darstellung von Mustern in entwickelten Modellen

Die Darstellung von Mustern in entwickelten Modellen bezieht sich zumeist auf die Struktur von objektorientierten Analyse- und Entwurfsmustern, die möglichst in entwickelten Modellen wiedererkennbar sein sollen. Dabei ist beispielsweise eine farbige Kennzeichnung von Klassen möglich, die an ein und demselben Muster beteiligt sind (vgl. [Wins96], S. 50). Auch eine Umrahmung, ähnlich der Subjektschicht der OOA, wäre denkbar (vgl. [CoYo94], S. 131ff).

Nachdem Muster in der UML zunächst durch die Nutzung von Stereotypes beschrieben werden mussten, definiert die UML seit der Version 1.3 eine gestrichelte Ellipse als Darstellungsform für Muster (vgl. [OMG99]). Dabei werden die am Muster beteiligten Klassen und Objekte durch gestrichelte Kanten bzw. Linien mit der Ellipse verbunden. Die Rolle, die eine Klasse im Rahmen eines Musters einnimmt, wird dabei durch die Bezeichnung der Kante beschrieben. Dieses Vorgehen ermöglicht die Darstellung von Überlagerungen von Mustern, die entstehen, wenn ein Objekt bzw. eine Klasse Bestandteil mehrerer Muster ist.

Au3. Automatisches Erkennen von Mustern in Modellen

Ausgehend von einer maschinellen Musterbeschreibung ist es insbesondere für strukturorientierte Entwurfsmuster möglich, diese in Entwurfsmodellen zu entdecken und durch eine der Techniken zur Darstellung von Mustern in Modellen grafisch darzustellen. Eine Entdeckung neuer Muster, deren Beschreibung nicht in maschineller Form vorliegt, wird beispielsweise in [Ede+97] als nicht automatisierbar abgelehnt. Letztlich beruhen alle bekannten Ansätze der automatisierten Mustererkennung auf maschinellen Beschreibungsformen von Mustern. Neben vollautomatisierten Verfahren sind auch halbautomatisierte Verfahren denkbar. Dabei besteht ein halbautomatisiertes Verfahren beispielsweise aus einer ersten automatisierten Phase und einer zweiten manuellen Phase, in der die Ergebnisse der ersten Phase beurteilt werden und falsch erkannte Musterstrukturen

verworfen werden (vgl. [Kel+99] S. 228; [Brow01]). Um die Verbesserung eines Modells zu unterstützen, ist es beispielsweise auch denkbar, gezielt nach Strukturen falsch implementierter Muster oder nach Antimustern zu suchen, mit dem Ziel, Schwachstellen in einem bestehenden Modell aufzufinden und zu beseitigen.

Ausgehend von einem existierenden Modell wird entweder durch eine statische Analyse des Modells oder im Falle von Programmkode teilweise auch durch eine dynamische Analyse zur Laufzeit (vgl. [Brow01]) versucht, Muster aus einem Pattern Repository wiederzufinden. Die meisten Ansätze beschränken sich auf das Erkennen von objektorientierten Entwurfsmustern in existierendem Programmkode im Rahmen von Reverse Engineering Projekten.[1] Allerdings ist es auch denkbar, weitere Arten von Mustern, z.B. Analysemuster, zu erkennen. Letztlich ist das Erkennen von Mustern nicht auf Quellkode beschränkt, sondern auch für andere Modellarten, z.B. Entwurfsmodelle denkbar.

Für die Analyse von Quellkode wird dabei davon ausgegangen, dass sich Entwurfsmuster sowohl in bestimmten charakterisierenden Vererbungs- und Kompositionsbeziehungen als auch in bestimmten Kommunikationsbeziehungen im Quellkode niederschlagen. Durch den Vergleich dieser Beziehungen mit denen der Muster eines Pattern Repositories lassen sich dann Muster wiederfinden. Vor der konkreten Suche nach Musterinstanzen wird der Quellkode zumeist zunächst in eine Zwischenform übersetzt, die sich durch ein Klassendiagramm darstellen lässt.

Au4. Automatische Anwendung von Mustern

In der Literatur werden vor allem Vorgehensweisen beschrieben, die zur automatischen Anwendung von Entwurfsmustern dienen. Jedoch ist es auch denkbar, analoge Methoden für weitere Musterarten zu entwickeln. Dabei lassen sich prinzipiell alle Arten modellorientierter Muster automatisch implementieren. Unter einem modellorientierten Muster wird im Rahmen dieser Arbeit ein Muster verstanden, das sich in seinen Artefakten in einem Modell

[1] Für eine Abgrenzung des Begriffs Reverse Engineerung siehe [ChCr90]. „Reverse Engineering is the process of analyzing a subject system to identify the system´s components and their relationships and create representations of the system in another form or at a higher level of abstraction“ ([ChCr90], S. 15).

wiedererkennen lässt. Die Mehrzahl von Entwurfsmustern ist modellorientiert. Prozessmuster sind generell nicht modellorientiert.

Das größte Potenzial liegt in der automatisierten Anwendung von Analysemuster, Entwurfsmustern und Idiomen. Idiome lassen sich beispielsweise in Programmierumgebungen oft relativ leicht durch die Anwendung von Makros bzw. Templates implementieren (vgl. [Copl01]). Für die automatisierte Anwendung von Entwurfsmustern lassen sich zwei Ansätze unterscheiden: die automatische Generierung von Entwurfsmodellen und die automatische Generierung von Quellkode.

Bosch unterscheidet in [Bosc01] drei Ansätze für die automatisierte Anwendung von Entwurfsmustern. Dabei untergliedert er die automatische Generierung von Quellkode noch einmal in einen generativen Ansatz, bei dem für jedes angewandte Muster Teile des Quellkodes automatisch generiert werden, um dann im Nachhinein manuell nachbearbeitet werden zu können und einen Ansatz mit speziellen Programmiersprachenkonstrukten, bei dem die Programmiersprache selbst die Möglichkeit bietet, Muster zu repräsentieren (vgl. [Bosc01], S. 3). Da sowohl der generative Ansatz als auch der Ansatz mit Programmiersprachenkonstrukten die beiden Phasen Detailentwurf und Implementierung betreffen, werden sie zusammenfassend in Abschnitt 3.2 analysiert.

Die automatische Generierung von Entwurfsmodellen beginnt mit der Zuordnung von Modellklassen zu den Rollen, die in einem Entwurfsmuster beschrieben werden. Rollen, zu denen keine Klasse gefunden werden kann, können durch neu generierte Klassen belegt werden. Im Anschluss daran wählt der Modellierer bestimmte Optionen, die in der Beschreibung des Entwurfsmusters dargestellt sind. Anhand der zugeordneten Rollen und der gewählten Optionen wird dann durch eine Abfolge von Minitransformationen, wie z.B. die Generierung von Methoden, die Umbenennung von Parametern etc. das gewählte Muster in das bestehende Modell integriert (vgl. [Toge01]).

3.2 Das Zusammenwirken von Mustern im gesamten Prozess der Systementwicklung

Während die isolierte Anwendung von Mustern innerhalb des Mikroprozesses für Muster im vorherigen Abschnitt analysiert wurde, werden nun die Zusammenhänge zwischen den

unterschiedlichen Aktivitäten der Systementwicklung in Bezug auf die Anwendung von Mustern untersucht. Dabei geschieht dies zunächst anhand des Wasserfallmodells. Gegebenenfalls wird auf die Ansätze zur Überwindung der Schwächen des Wasserfallmodells (K1 bis K5) verwiesen.

Eine zeitliche Abfolge von Aktivitäten bedingt, dass sich der Einsatz von Mustern neben der aktuellen Aktivität nur auf zeitlich nachgelagerte Aktivitäten auswirken kann. Dies führt zur Unterscheidung in die beiden Phasen Musteranwendung und Musternutzung.

Bei Anwendung des Wasserfallmodells bedeutet dies, dass sich beispielsweise die Anwendung von Entwurfsmustern (= Musteranwendung) nur auf Implementierung, Test, Einführung und Wartung auswirken kann (= Musternutzen). Diese Beschränkung wird durch die Einführung inkrementeller Entwicklungsprozesse (K1) oder die Entwicklung mittels evolutionärer Prototypen (K2) wieder abgeschwächt, da hier die zeitliche Abfolge von Aktivitäten des Wasserfallmodells aufgeweicht wird.

Im Falle inkrementeller Entwicklungsprozesse wirken sich beispielsweise die Ergebnisse der Anwendung von Entwurfsmustern in vorhergehenden Entwurfsphasen auf nachgelagerte Entwurfsphasen aus. Im Falle von Idiomen kann deren Anwendung in zeitlich vorgelagerten Iterationen die Anwendung weiterer Entwurfsmuster in den Phasen des Detailentwurfs nachgelagerter Iterationen vereinfachen, insofern der Einsatz von Idiomen den Übergang von Entwurfsmodellen zu Programmkode flexibler gestaltet und damit erleichtert.

Bei der Anwendung evolutionärer Prototypen erweitert sich die Musternutzung auf sämtliche nachgelagerten Transformationszyklen. Dies unterstützt die Forderung nach einer Dokumentation von Mustern in entwickelten Modellen. In [Pre+97a] weisen PRECHELT et al. anhand eines Experiments nach, dass die Dokumentation von Mustern die Wartung beschleunigt sowie die Qualität von Wartungsaktivitäten erhöht (vgl. [Pre+97a]).

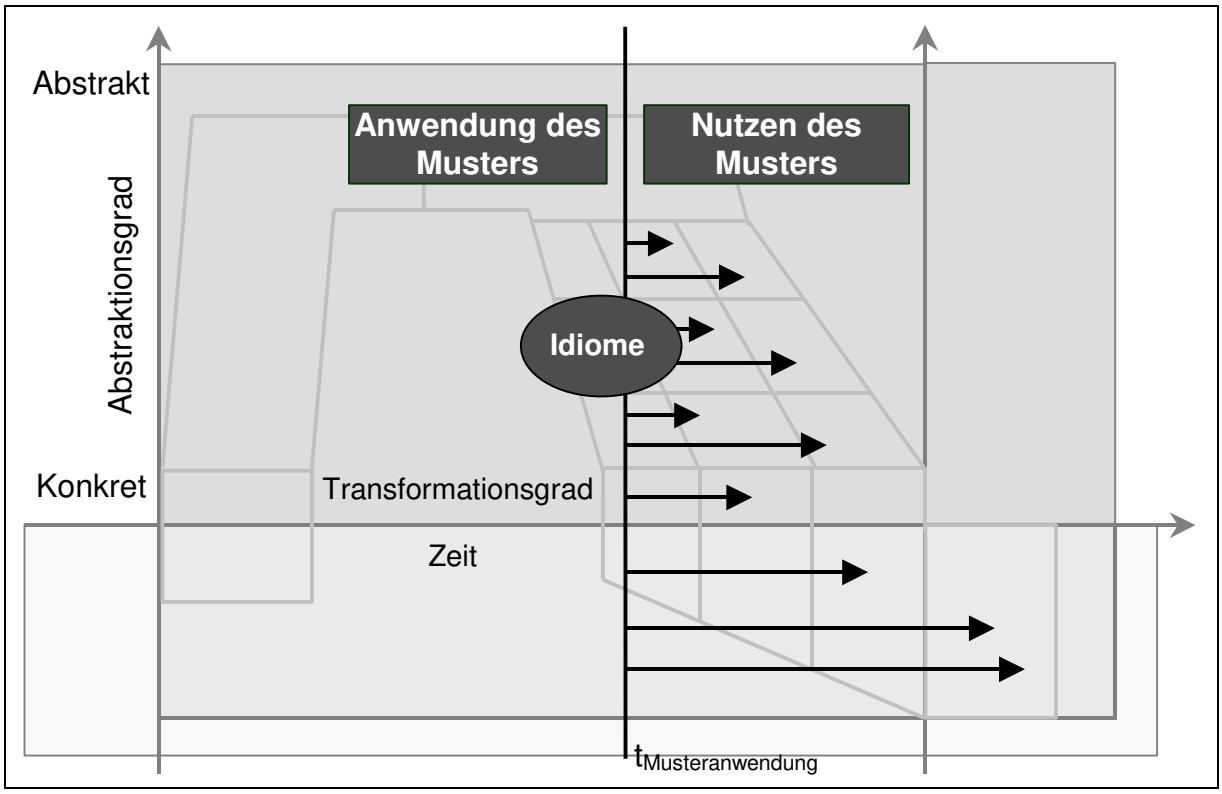

Abbildung 13: Beispielhafte Darstellung der Auswirkungen der Anwendung von Idiomen auf zeitlich nachgelagerte Aktivitäten im Rahmen einer inkrementellen Entwicklung (vgl. Abb. 6)

Die Verwendung des Wiederverwendungsansatzes Muster lässt erwarten, dass auch andere Konzepte der Wiederverwendung (K4) wie z.B. die Nutzung von Komponenten oder Rahmenwerken angewendet werden. Der Zusammenhang zwischen Mustern und Rahmenwerken wurde schon in Abschnitt 2.2.2 analysiert. Sowohl die Nutzung explorativer Prototypen (K3) und objektorientierter Konzepte (K5) haben keinen Einfluss auf die Nutzung von Mustern. Ein Indiz dafür könnte eine geringe Anwendung dieser Konzepte unter den Teilnehmern der Studie sein. Ein konkreten Zusammenhang kann jedoch nur eine Studie nachweisen, die auch Nichtnutzer von Mustern einschließt.

Automatische Quellkodegenerierung

Zunächst wird der schon im vorherigen Abschnitt erwähnte Ansatz zur automatischen Generierung von Quellkode auf der Basis von Entwurfsmustern analysiert. Dieser Ansatz betrifft sowohl Detailentwurf (Ak4) als auch Implementierung (Ak5). In der Annahme, dass automatisch erstellter Quellkode aufgrund besserer Qualität weniger Testaufwand bedeutet, würde sich dies auch noch auf den Test (Ak6) auswirken. Entwurfsmuster werden häufig

durch die Angabe von Quellkodefragmenten illustriert (vgl. [Gam+94], S. 6f). Durch eine Verallgemeinerung dieser Fragmente ist es beispielsweise denkbar, Quellkode aus Templates zu erzeugen. Dabei ist die Forderung, dass der generierte Quellkode nicht mehr manuell verändert werden muss, zu beachten (vgl. [VlAl01]). Modellorientierte Entwurfsmuster werden dabei in Form von Templates beschrieben. Diese Templates werden dann durch die Auswahl konkreter Namen für Klassen und Methoden und durch die Auswahl bestimmter Optionen, die eine konkrete Variante des Muster beschreiben, instanziiert (vgl. [Bud+96]). Dabei führt die Auswahl einer bestimmten Mustervariante dazu, dass gewisse Teile des Templates in den generierten Quellkode übernommen werden und andere ausgeschlossen werden.

Neben der Anwendung im Haupttransformationszyklus wird die automatische Quellkodegenerierung auch im Rahmen des Reengineering[1] als eine der Tätigkeiten der progressiven Wartung genutzt. Dabei wird der Quellkode zunächst anhand automatischer Mustererkennung nach Ansatzpunkten für den Einsatz von Entwurfsmustern untersucht. Die so wiedergewonnenen Teile des Entwurfsmodells werden dann durch automatische Anwendung von Entwurfsmustern in ein besseres Entwurfsmodell überführt und dann durch die Anwendung automatischer Quellkodegenerierung in den existierenden Quellkode eingepasst (vgl. [CiNi99]). Die schrittweise Anpassung von Quellkode wird dabei als Refactoring bezeichnet.

Musterverkettung

Einer der Vorteile objektorientierter Methoden wie OMT ist es, dass sie sich der gleichen objektorientierten Konzepte (K5) in unterschiedlichen Aktivitäten bedienen. Dadurch lassen sich z.B. Objekte bzw. Klassen, die während der Analyse eingeführt wurden, teilweise in der Implementierung wiederfinden. Die Verwendung eines einheitlichen Konzepts ermöglicht es auch, die Trennung der im Wasserfallmodell noch strikt unterschiedenen Aktivitäten aufzulösen. "Für die objektorientierte Systementwicklung ist der fließende Phasenübergang sogar typisch" ([StHa97], S. 256). Ein ähnliches Konzept gilt für die Anwendung von Mustern. Wie schon im Rahmen automatischer Quellkodegenerierung erläutert, resultiert die

[1] „Reengineering ... is the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form“ ([ChCr90], S. 15).

Anwendung eines bestimmten Entwurfsmusters in bestimmten charakteristischen Implementierungsmustern (= Idiome) im Quellkode. Erweitert man diese Idee auf vorhergehende Phasen, so heißt dies, dass Architekturmuster zu bestimmten Entwurfsmustern führen und Analysemuster zu bestimmten Architekturmustern. Wie schon bei der Definition von Testmustern dargestellt, führt die Anwendung bestimmter Entwurfsmuster zur Anwendung analoger Testmuster. Gleiches gilt für die Anwendung von Wartungsmustern (vgl. Abschnitt 2.2.3). Die Anwendung von gleichartigen Mustern, z.B. von Entwurfsmustern, wurde schon im Rahmen des Mikroprozesses in Abschnitt 3.1.1 diskutiert.

In [RaSr00] werden die Beziehungen von objektorientierten Analysemustern, Entwurfsmustern und Idiomen untersucht. Die Beziehungen zwischen unterschiedlichen Musterarten werden dabei in vier Stufen charakterisiert (vgl. [RaSr00], S. 365f):

- Komplette vs. partielle Überführung: Sind sämtliche Klassen des Analysemusters im Entwurfsmuster wiederzufinden?
- Replizierende vs. nichtreplizierende Überführung: Wird ein Analysemuster in mehr als einer Instanz des gleichen Entwurfsmusters abgebildet?
- 1:1 Überführung vs. 1:* Überführung: Wird ein Analysemuster in mehr als einem Entwurfsmuster abgebildet?
    - 1:* überlappende Überführung vs. 1:* nichtüberlappende Überführung: Die 1:* Überführung wird noch einmal darin unterschieden, ob sich die anzuwendenden Entwurfsmuster teilweise in ihrer Zuordnung zu Klassen überlappen.

Eine ähnliche Klassifizierung lässt sich auch für die Überführung von Entwurfsmustern in Idiome anwenden.[1] Die Anwendung von Mustern folgt dabei den Phasen des Wasserfallmodells. So ist zu bezweifeln, dass Muster nachgelagerter Phasen Muster vorgelagerter Phasen bedingen. Dies bedingt auch, dass selbst bei einer Abkehr von der

[1] Ob und wie sich eine durchgehende Kette von Analysemustern bis hin zu Implementierungsmustern finden lässt, könnte Inhalt weiterer Forschung sein. Dabei wären auch Aspekte, wie z.B. die Vollständigkeit einer solchen Abbildung (vgl. [Jac+98], S. 137) oder aber, ob solche Abbildungen auch für andere Musterarten sinnvoll erscheinen, zu diskutieren.

strikten sequenziellen Abfolge des Wasserfallmodells, z.B. durch die Anwendung inkrementeller Techniken (K1) oder evolutionärer Prototypen (K2), die konsequente Anwendung von Mustern (inklusive ihrer Abhängigkeiten) wieder dazu führen würde, dass letztlich die einzelnen Aktivitäten wieder ähnlich dem Wasserfallmodell aufeinander aufbauen. Die aus der Verwendung objektorientierter Konzepte (K5) bekannte Möglichkeit, durch eine einheitliche Notation in unterschiedliche Aktivitäten die strikte Trennung der Aktivitäten aufzuheben (vgl. [StHa97], S. 256), könnte dieses Problem umgehen. Daraus leitet sich die Forderung ab, Muster in einer einheitlichen Notation darzustellen und bei einem Wechsel zwischen unterschiedlichen Aktivitäten ein Musterteilmodell bzw. eine Mustersichtweise beizubehalten und zu pflegen (vgl. [Yac+00], S. 164f).

Beispielhaft seien hier einige einzelne Verkettungen zwischen Mustern unterschiedlicher Phasen angegeben:

- Analysemuster zu Entwurfsmuster: Die Anwendung des Analysemusters Organisation Hierarchie führt zur Anwendung des Entwurfsmusters Composite (vgl. [Fowl97], S. 19ff ; [Gam+94], S. 163ff).

- Architekturmuster und Entwurfsmuster: Die Anwendung des Architekturmusters Model-View-Controller führt u.a. zur Anwendung des Entwurfsmusters Observer (vgl. [Bus+98], S. 124ff; [Gam+94], S. 293ff).

- Entwurfsmuster zu Testmuster: Die Anwendung des Entwurfsmusters Observer führt zur Anwendung des Testmusters Observer Test (vgl. [McGr99], S. 17ff; [Gam+94], S. 293ff).

- Entwurfsmuster und Idiome: Die Anwendung des Entwurfsmusters Abstract Factory führt zur Anwendung des Idioms Virtual Constructor (vgl. [Copl01]; [Gam+94], S. 87ff).

Standardisierung

Durch die Nutzung von Mustern in zahlreichen Systementwicklungsprojekten ergibt es sich, dass diese Muster in Software wieder und wiederzufinden sind (E4). Daraus resultiert, dass die breite Nutzung von Mustern einen Quasi-Standard setzt, der praktisch von der Mehrzahl

der Systementwicklungsprojekte befolgt werden muss. Die Eigenschaft von Mustern, bewährte praktische Vorgehensweisen zu beschreiben und dabei explizit von der Beschreibung innovativer Lösungen abzusehen (E11), unterstützt den Standardisierungscharakter von Mustern. Ob neben der weiten Verbreitung von Entwurfsmustern, u.a. aufgrund des Erfolges von [Gam+94], auch andere Musterarten wie Prozessmuster und Organisationsmuster zu Standards werden, ist zu untersuchen (vgl. [Clin96], S. 47).

Kommunikation

Aus dem Standardisierungscharakter von Mustern leitet sich direkt deren Nutzung im Rahmen der Kommunikation innerhalb des Systementwicklungsprozesses ab. Am Systementwicklungsprozess sind im Normalfall neben Programmierern und Spezialisten für Analyse, Entwurf, Test etc. auch Kunden, Endbenutzer und Experten aus dem Anwendungsgebiet beteiligt (vgl. [RaGo01]). Die erste Gruppe wird im Folgenden als Softwarespezialisten bezeichnet, während die zweite Gruppe als Anwendungsspezialisten bezeichnet wird. Die Nutzung von Mustern kann die Kommunikation zwischen allen Beteiligten verbessern.

Im Allgemeinen ist zu erwarten, dass Anwendungsspezialisten kein Wissen über Muster im Prozess der Systementwicklung haben. Selbst unter Softwarespezialisten wird der Ausbildungsgrad, also die Vertrautheit mit Mustern zumeist stark variieren.

Ein Vorteil von Mustern ist, dass sie die Kommunikation innerhalb von Teams von Softwarespezialisten, die mit den Mustern vertraut sind, verbessert. Dabei stellen sie ein gemeinsames Vokabular zur Verfügung (vgl. [Gam+94], S. 352). Anstatt ein Problem-Lösungs-Paar bei jeder Nutzung erneut zu beschreiben, wird auf das entsprechende Muster verwiesen. Daher spielt die Bezeichnung eine wichtige Rolle in der Beschreibung des Musters (vgl. Abschnitt 3.1.2). Die Eigenschaften des Musters, z.B. seine Vor- und Nachteile, sind jedem Beteiligten sofort klar und müssen nicht jedes Mal erneut erläutert werden. Unterschiedliche Grade der Vertrautheit werden z.B. dadurch aufgefangen, dass ein schnelles Nachschlagen der Musterbeschreibung aufgrund isolierter Beschreibungsformen (E14) möglich ist.

Neben der Nutzung von Mustern in Teams von Softwarespezialisten, können Muster auch in der Kommunikation mit Anwendungsspezialisten benutzt werden. Obwohl Anwendungsspezialisten im Allgemeinen nicht mit Mustern vertraut sind, so bieten Muster doch die Möglichkeit, sie in der Kommunikation als Vorgehensrahmen zu verwenden (vgl. [Fowl97] S. 12). So unterstützt beispielsweise die Anwendung von Analysemustern eine gezielte Analyse von typischen Sachverhalten. Durch die Abarbeitung einer Anzahl typischer Muster eines Anwendungsgebiets können beispielsweise Probleme schneller erkannt werden. Diese Probleme können dann anhand der mit dem Muster verbundenen Lösung, einschließlich ihrer Vor- und Nachteile, mit dem Anwendungsspezialisten diskutiert werden. Fowler warnt vor dem psychologischen Effekt, dass die Anwendung von Mustern in der Anforderungsanalyse oder in der Analyse Kunden verschreckt, da Muster letztlich Erfahrungen aus anderen Projekten enthalten (E11). „Some clients don't like to think of themselves as similar to other clients ([Fowl97], S.12).

Die Nutzung von Mustern in der Kommunikation hilft auch, den Umgang mit Kunden, Marketing- und Verkaufsexperten zu verbessern. Muster sind im Vergleich zu Klassenmodellen schneller von Laien zu erfassen. Sie bieten aufgrund der Beschreibung von Konsequenzen ihrer Anwendung auch für Laien die Möglichkeit, diese einzuschätzen und gegeneinander abzuwägen (vgl. [Schm01], S. 6).

Dokumentation

Gerade in Systementwicklungsprojekten mit mehreren Beteiligten kommt der Dokumentation der Ergebnisse der einzelnen Phasen bzw. Aktivitäten eine entscheidende Rolle zu. Dabei dient die Dokumentation als Grundlage nachgelagerter Aktivitäten. Die Dokumentation von Mustern kommuniziert dabei die „Vision bei dem Entwurf des Software-Systems“ ([Bus+98], S. 6) an weitere Beteiligte. Dies ermöglicht anderen, ihre Aktivitäten dieser Vision entsprechend zu gestalten, und hilft somit, eine konsistente Anwendung von Architektur- und Entwurfsprinzipien zu garantieren.

Da Muster die Kommunikation zwischen Softwarespezialisten verbessern, ist auch die Nutzung von Mustern in der Dokumentation empfehlenswert. Ein durch Anwendung von Mustern erstelltes Modell sollte dokumentieren, welche Muster in Erstellung des Modells

eingeflossen sind. Dies erhöht die Nachvollziehbarkeit der Aktivität. Die Darstellung von Mustern in objektorientierten Modellen (Au2) wurde in Abschnitt 3.1.3 diskutiert.

Eine besondere Bedeutung kommt der Dokumentation von Mustern im Quellkode zu. Neben der Dokumentation von Mustern in Kommentaren erleichtert die Benennung von Variablen, Funktionen, Modulen etc. das Verständnis von Quellkode und damit eine manuelle Veränderung von Quellkode im Rahmen korrektiver und progressiver Wartungsaktivitäten. Anhand eines Experiments bestätigen PRECHELT et al., dass die Dokumentation von Entwurfsmustern im Quellkode die Qualität von Änderungen, die im Rahmen der Wartung vorgenommen wurden, verbessert. Daneben wurde auch bestätigt, dass diese Änderungen schneller durchzuführen waren (vgl. [Pre+97a]). In [Broo95] schlägt Brooks vor, die Dokumentation in den Quellkode zu integrieren, da so Dokumentation und Quellkode auf dem gleichen Stand gehalten werden und im Falle von Änderungen die Dokumentation sofort zur Verfügung steht. Ein solches Vorgehen würde die Forderung nach einer Dokumentation von Mustern im Quellkode nur verstärken.

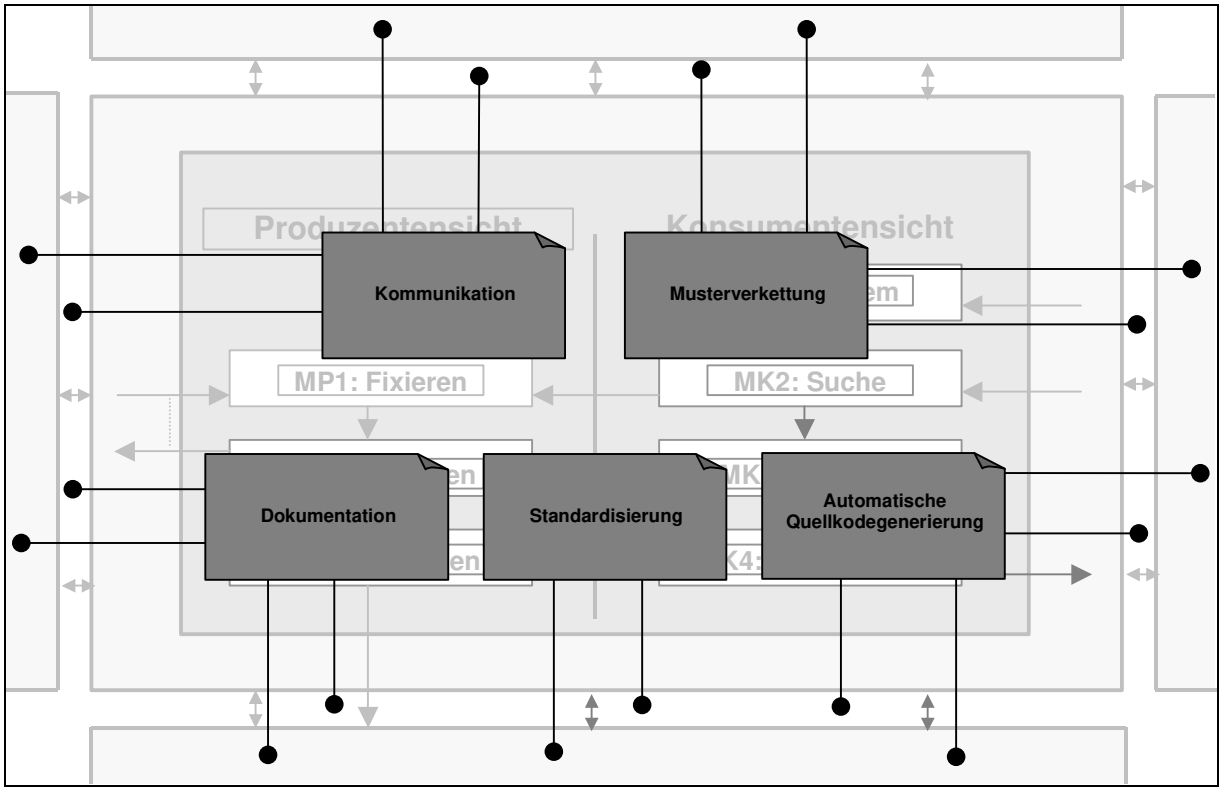

Abbildung 14: Die fünf Ansätze für das Zusammenwirken von Mustern zwischen verschiedenen Aktivitäten des Makroprozesses (vgl. Abbildung 11)

3.3 Implikationen für die empirische Studie

Ausgehend von den in Abschnitt 3 diskutierten Themen werden hier sich daraus ergebene Analysegebiete für die empirische Studie zusammengefasst.

In Verbindung mit den sich aus Abschnitt 1 und 2 ergebenen Fragen nach den Aktivitäten der Systementwicklung, mit denen sich die Befragten beschäftigen, sowie deren generelle Unterstützung durch Werkzeuge ergibt sich die Frage nach dem Grad der Automatisierung der Musteranwendung in der Praxis. Dabei wird dies noch einmal nach den vier unterschiedlichen Automatisierungsansätzen (Au1 bis Au4) unterschieden (Frage 3C).

Neben der automatisierten Anwendung von Mustern können Muster auch manuell, also aus Büchern oder als Teil der Erfahrung und Ausbildung, angewandt werden. Inwieweit dies in der Praxis durchgeführt wird, wird anhand des Fragebogens überprüft (Frage 3D).

Das Verhältnis zwischen der Anzahl von Musterproduzenten und Musterkonsumenten wird empirisch ermittelt (Frage 2A, 2B). Wie schon in Abschnitt 1 erwähnt, wird analysiert, inwiefern die Nutzung der Konzepte zur Überwindung der Schwächen des Wasserfallmodells mit der Aufnahme von Musterkonsum und Musterproduktion in das genutzte Vorgehensmodell korrespondiert (Frage 4C, 4D).

Welche der in 3.1.2 aufgeführten Musterbeschreibungsformen und Musterorganisationsschemata in der Praxis genutzt werden, wird anhand der Umfrage ermittelt (Frage 5A, 5B). Da die Nutzung geeigneter Musterorganisationsschemata, insbesondere mit Blick auf die problemgebundene Suche nach geeigneten Mustern in der Praxis scheinbar problematisch erscheint, wird versucht dies zunächst anhand einer entsprechende Frage zu bestätigen. Welcher der existierenden Ansätze sich am besten für diese Suche eignet, wird anhand der Umfrage ermittelt. Eine Frage danach, welche Beziehungen zwischen Mustern ein eventuell zu entwickelndes Musterorganisationsschema abbilden sollte, ergibt sich aus der Diskussion in Abschnitt 3.1.2 (Frage 5C, 5D, 5E).

4. Eine Bewertung des Einsatzes von Mustern

4.1 Eine Bewertung anhand der Grundsätze ordnungsgemäßer Modellierung

Im Folgenden wird der Einsatz von Mustern im Rahmen des Haupttransformationsprozesses bewertet. Dabei wird vor allem auf modellorientierte Muster, also auf Muster, deren Artefakte sich in einem Modell wiederkennen lassen, eingegangen. Dazu zählen die Mehrzahl von Architekturmustern, Analysemustern und Entwurfsmustern. Sieht man Programmkode als die Beschreibung eines im Rahmen der Implementierung entwickelten Modells an, so zählen auch Idiome dazu.

Ausgehend von den in [Schü98] aufgeführten Grundsätzen ordnungsgemäßer Modellierung (GoM) wird eine Bewertung des Einsatzes von Mustern vorgenommen (vgl. [Schü98], S. 119ff).

Grundsatz der Konstruktionsadäquanz

„Es wird der Nutzen des Modells für Zwecke seiner praktischen Anwendung betont“ ([Schü98], S. 119). Dabei besteht die Forderung an ein Modell, dass es sowohl den Konsens über ein zu repräsentierendes Problem als auch den Konsens über die Modelldarstellung unterstützt.

Wie schon im Abschnitt 3.2 erläutert, unterstützt die Anwendung von Mustern die Kommunikation zwischen den am Systementwicklungsprozess Beteiligten. Dies hat zweifellos einen positiven Einfluss auf die Erreichung eines Konsenses und eines gemeinsamen Verständnisses. Die Benutzung von einheitlichen Namenskonventionen, wie sie durch die Beschreibung von Mustern vorgegeben ist und beispielsweise in der Dokumentation von Mustern in Entwurfsmodellen oder im Quellkode zu finden ist, erhöht die im Rahmen der Konstruktionsadäquanz geforderte „einheitliche und eindeutige Benennung von Sachverhalten“ ([Schü98], S. 122).

Grundsatz der Sprachadäquanz

Die Sprachadäquanz wird anhand von Spracheignung und Sprachrichtigkeit bewertet.

Durch die Darstellung von Mustern wird eine zusätzlich Sicht auf ein Modell beschrieben. Die semantische Mächtigkeit wird durch die Aufnahme des zusätzlichen Sprachkonstruktes Muster erhöht. Die Nutzung von Mustern erhöht die Verständlichkeit, insofern mit Mustern Expertenwissen über isolierte Probleme (E8, E14) relativ leicht vermittelt werden kann. Dies ermöglicht selbst Laien ein verbessertes Verständnis der Zusammenhänge in einem Modell.

Grundsatz der Wirtschaftlichkeit

Das Ziel der Wirtschaftlichkeit steht oft in Konflikt mit anderen Zielen. Teilziele sind unter anderem Flexibilität und Robustheit.

Ein Muster beschreibt die Kräfte bzw. Umstände, durch die es in seiner Lösung beschränkt ist. Damit geht die Beschreibung von Mustern zumeist auf Aspekte wie Änderbarkeit, Interoperabilität, Effizienz, Zuverlässigkeit, Testbarkeit und Wiederverwendbarkeit ein (vgl. [Bus+98], S. 400f). Während sich die GoM auf die Bewertung eines Modells beschränken, gehen Muster meist explizit auf Effekte in nachgelagerten Phasen des Systementwicklungsprozesses ein. Muster unterstützen jedoch die Eigenschaften guter Modelle, indem sie unter anderem auf die Eigenschaften Flexibilität und Robustheit optimiert sind. Eine Bewertung der Wirtschaftlichkeit der Anwendung von Mustern kann nur aus der Erfahrung konkreter Projekte abgeleitet werden.

Grundsatz des systematischen Aufbaus

Der systematische Aufbau eines Modells wird durch eine einheitliche Verwendung von Komponenten in unterschiedlichen Sichten unterstützt. Die Nutzung von Mustern hat keinen Einfluss auf diesen Grundsatz.

Grundsatz der Klarheit

Neben einer adressatengerechte Hierarchisierung von Modellen wird hier auch die grafische Gestaltung von Modellen und adressatengerechte Filterung bewertet.

Die Verwendung von Mustern in der Darstellung von Modellen erlaubt eine zusätzliche Sichtweise auf ein Modell. Die grafische Gestaltung von Mustern in der UML erlaubt es, zusätzliche Information über das Modell zu vermitteln. Ein adressatengerechtes Filtern, z.B.

durch ein Konzentration auf die Mustersicht eines Modells, kann in der Kommunikation mit Anwendungsexperten helfen (vgl. Abschnitt 3.2).

Grundsatz der Vergleichbarkeit

Dieser Grundsatz hebt die Möglichkeit des semantischen Vergleichs zweier Modelle hervor, d.h. „ ... es sollen die mit zwei Modellen beschriebenen Inhalte hinsichtlich ihrer Deckungsgleichheit untersucht werden“ (vgl. [Schü98], S. 133).

Musterbeschreibungen beinhalten zumeist die mit ihrer Nutzung einhergehenden Vor- und Nachteile (E13). Damit erleichtert die Darstellung von Mustern in Modellen die Vergleichbarkeit von Modellen.

4.2 Vorteile und Nachteile bei der Nutzung von Mustern

Während im vorhergehenden Abschnitt vor allem die Vorteile, die sich aus der Nutzung von Mustern im Systementwicklungsprozess ergeben anhand der Grundsätze ordnungsgemäßer Modellierung analysiert wurden, werden im Folgenden neben diesen auch Nachteile, die bei der Nutzung von Mustern auftreten, zusammengefasst.

Alle Eigenschaften der Nutzung von Mustern, die im Folgenden zusammengefasst werden, basieren auf in der Literatur zu findenden Erfahrungsberichten. Im Rahmen der empirischen Untersuchung wird versucht, diese Erfahrungen zumeist einzelner Organisationen zu bestätigen.

Teilweise beeinflussen sich die Eigenschaften gegenseitig oder sind wie z.B. im Falle der Einführung einer zusätzlichen Ebene der Abstraktion (N8) durchaus als Vor- und Nachteil gleichzeitig zu interpretieren.

Auswirkungen auf die Eigenschaften der mit Mustern entwickelten Softwaresysteme

N1. (+) „Muster unterstützen die Entwicklung von Software mit definierten Eigenschaften“ ([Bus+98], S. 7). Dabei zielen Muster explizit auf nichtfunktionale Eigenschaften von Software ab.

N2. (+) „Muster helfen bei der Entwicklung komplexer, heterogener Softwarearchitekturen“ ([Bus+98], S. 7). Muster sind zwar nicht immer besser als eigene Lösungen, letztlich vereinfachen sie jedoch in vielen Fällen die Anwendung von bewährten Vorgehensweisen.

N3. (+) Muster verbessern die Dokumentation des Prozesses der Systementwicklung (vgl. [Bus+98], S. 6; [ScSt95], S. 12).

N4. (+) Muster verbessern die Erweiterbarkeit, die Portierbarkeit und die Wiederverwendbarkeit von Software (vgl. [ScCl99], S. 18; [Helm95], S. 338f; [Clin96], S. 47).

N5. (+) Muster verbessern die Klarheit eines Entwurfs (vgl. [ScCl99], S.18).

N6. (+) Muster geben Programmierern in der Wartungsphase einen Rahmen für vorzunehmende Änderungen, der garantiert, dass die ursprüngliche Entwurfsvision beibehalten wird (vgl. [Clin96], S. 48).

N7. (-) Muster sind mitunter im Quellkode nur schwer zu erkennen, obwohl der Quellkode unter Nutzung von Mustern entwickelt wurde (vgl. [Mann00], S. 15; [Bosc98], S. 19).

N8. (-) Muster erhöhen den Aufwand zur Laufzeit durch die Einführung einer zusätzliche Ebene der Abstraktion (vgl. [ScCl99], S.18).

Individuelle Erfahrungen mit Mustern

N9. (+) Muster verbessern die Kommunikation zwischen allen am Systementwicklungsprozess beteiligten Personen (vgl. [ScSt95], S. 12; [Bec+01], S. 10; [Helm95], S. 339).

N10. (+) Muster unterstützen die Integration neuer Entwickler (vgl. [Helm95], S. 339; [Schm01], S. 8).

N11. (-) Mitunter ist es schwierig zu entscheiden, welches Muster ein gegebenes Problem am besten löst (vgl. [Shaw94], S. 4).

N12. (-) Muster sind für eine direkte Anwendung nicht ausreichend spezifiziert (vgl. [ScSt95], S. 12).

N13. (-) Die Suche nach Mustern ist ein personalintensiver Vorgang (vgl. [Schm01], S.7).

N14. (-) Einige Muster sind nur schwer zu erlernen (vgl. [Clin96], S. 49).

N15. (-) Muster führen Entwickler dazu zu glauben, mehr zu wissen, als es tatsächlich der Fall ist (vgl. [Helm95], S. 339; [Schm01], S. 6).

Muster im Prozess der Softwareentwicklung

N16. (+) Muster helfen beim Übergang zur Nutzung objektorientierter Technologien (vgl. [Schm01], S. 7).

N17. (+) Muster erhöhen die Vorhersagbarkeit von Entscheidungen in Analyse, Entwurf und Programmierung und tragen somit zu einer Standardisierung von Modellen bei.

N18. (-) Muster erfordern im Vergleich zu anderen Wiederverwendungsansätzen eine wiederholte Implementierung (vgl. [Mann00], S. 15).

N19. (-) Muster verlassen sich nur auf die Erfahrung aus vorangegangenen Projekten. Sie generieren keine neuen Lösungen (vgl. [Pesc97], S. 130).

Kosten und Nutzen von Mustern

N20. (+) Der Einsatz von Mustern ist nicht mit hohen Kosten verbunden (vgl. [See+00], S. 447).

N21. (+) Muster vermeiden Kosten, die mit der Neuerfindung von Lösungen verbunden wären (vgl. [See+00], S. 447).

N22. (+) Muster verringern das Risiko in Systementwicklungsprojekten (vgl. [Schm01], S. 5).

4.3 Muster als Wettbewerbsfaktor

Muster fixieren Expertenwissen (E8). Ausgehend von dieser Mustereigenschaft werden im Folgenden Muster im Rahmen des Wissensmanagements einer Organisation untersucht.

Ähnlich dem Begriff Muster ist der Wissensbegriff der Umgangssprache entliehen. Dies resultiert in einer vielseitigen Nutzung des Begriffs Wissen mit einer teilweise kontextabhängigen Semantik. Folgende Definitionen grenzen den Begriff **Wissen**, wie er im Rahmen dieser Arbeit verstanden wird, ein:

„Gesicherter Bestand an Modellen über Sachverhalte und Objekte bzw. Objektbereiche, die partiell bei einem Menschen in Form seines Gedächtnisses, in einer gesellschaftlichen Gruppe, aber auch in einer Organisation, in einem ganzen Kulturkreis oder in der Menschheit insgesamt als kognitive Struktur vorhanden ist“ ([GéSz95], S. 607).

„Die Gesamtheit der Kenntnisse auf einem bestimmten Gebiet“ ([Hein95], S. 564).

Muster sind eine Beschreibungsform für Wissen. Dabei kann ein einzelnes Muster aufgrund seiner relativ kurzen Form (E14) nur Ausschnitte aus der Gesamtheit des Wissens über ein bestimmtes Sachgebiet beschreiben.

In [Nona91] schlägt NONAKA eine Untergliederung von Wissen in verborgenes[1] und explizites Wissen vor. Dabei ist verborgenes Wissen das in Individuen vorhandene Wissen, z.B. die Erfahrung von Experten auf einem Fachgebiet. „Explicit knowledge is formal and systematic. For this reason, it can be easily communicated and shared ... “ ([Nona91], S. 98). Eines der Ziele des Wissensmanagements ist es, das in einer Organisation vorhandene explizite und verborgene Wissen möglichst jedem Mitglied der Organisation zugänglich zu machen. Muster sind eine Möglichkeit, dies für den Bereich der Systementwicklung zu erreichen.

Die Verbreitung von Wissen innerhalb einer Organisation lässt sich dabei anhand einer Klassifizierung in vier Gruppen beschreiben (vgl. [Nona91], S. 98f). Im Folgenden wird kurz

[1] Der im englischen Original verwendete Begriff *tacit knowledge* wird hier mit dem Begriff *verborgenes Wissen* übersetzt.

die Nutzung von Mustern innerhalb dieser vier Klassen beschrieben, um den Charakter, den Muster im Rahmen des Wissensmanagements einnehmen, einzugrenzen.

Von verborgenem Wissen zu explizitem Wissen (Externalisierung)

Das in Individuen vorhandene Wissen wird dokumentiert. Muster beschreiben das Wissen von Experten auf einem Fachgebiet (E8). Daher ist der Prozess der Produktion von Mustern bzw. Musterideen, beispielsweise unterstützt durch den Mikroprozess für Muster (MP), als Prozess der Externalisierung zu sehen. Neben der Suche nach neuen Mustern im Rahmen des Mikroprozesses wird die Erstellung von Mustern in der Praxis häufig in Aktivitäten losgelöst vom konkreten Transformationsprozess durchgeführt. Neben der Integration in Form des Mikroprozesses lassen sich dabei folgende in der Praxis anzutreffende Ansätze unterscheiden (vgl. [Risi01b]; [Henn91], S. 270ff):

- Experteninterviews.
- Musterseminare, Musterversammlungen (intern, extern) und
- Die Durchsicht von Programmkode durch Experten (System Review).

Unabhängig davon können Muster auch als Teil der täglichen Arbeit, also parallel zu anderen Aktivitäten der Softwareentwicklung individuell entwickelt werden.

Von explizitem Wissen zu verborgenem Wissen (Internalisierung)

In Umkehrung des Prozesses der Externalisierung wird durch die Anwendung von Mustern das Wissen, das in Mustern fixiert wurde, zu verborgenem Wissen, das dann wiederum implizit angewandt wird. Muster unterstützen diesen Prozess, insofern sie das Wissen über ein bestimmtes Wissensgebiet in relativ kleine, isoliert beschreibbare Einheiten unterteilen (E14), die dann schrittweise in implizit angewandtes Wissen umgewandelt werden können. Folgende Verfahren zur Aneignung von Mustern sind in der Praxis anzutreffen (vgl. [Risi01c]; [Keri99]):

- Selbststudien mit Büchern und Internet,
- Musterlerngruppen und

- Der Besuch von Kursen.

Von verborgenem Wissen zu verborgenem Wissen (Sozialisation)

Dieser Prozess wird durch Muster nicht unterstützt. Grundlage für die Nutzung von Mustern ist deren Dokumentation in expliziter Form.

Von explizitem Wissen zu explizitem Wissen (Kombination)

Dieser Prozess ist durch den Austausch von Wissen gekennzeichnet. Musterkataloge, Mustersysteme und Mustersprachen unterstützten den Prozess der Kombination von Wissen. Die Organisation von Mustern in Musterorganisationsschemata (vgl. 3.1.2) schafft zusätzliches Wissen über den Zusammenhang von Mustern.

Muster unterstützen somit den Prozess des Wissensmanagements in einer Organisation. Da für Unternehmen der Softwareindustrie das Wissen und die Erfahrung der Mitarbeiter letztlich einer der entscheidenden Wettbewerbsfaktoren ist, kommt Mustern hier als Mittel des Wissensmanagements eine besondere Rolle zu. Hinsichtlich der Betrachtung als Wettbewerbsfaktor kommt Mustern dabei eine zweifache Rolle als Technologie zu. Muster selbst stellen eine Technologie dar. Gleichzeitig ist es Ziel von Mustern, Technologie innerhalb einer Organisation zu verbreiten.

Technologie ist „ … die Gesamtheit der anwendbaren und tatsächlich angewendeten Arbeits-, Entwicklungs-, Produktions- und Implementierungsverfahren in der Technik“ ([Hein95], S. 514). Technologien können dabei anhand ihres Wirkungspotenzials in Schlüsseltechnologien, Schrittmachertechnologien und Basistechnologien unterteilt werden. Inwiefern sich die Anwendung von Mustern hier einordnet, könnte anhand einer Studie untersucht werden.[1]

Die in Mustern beschriebenen Technologien können eine Wettbewerbsvorteil darstellen, wenn sie „ ... für die relative Kostenposition oder Differenzierung eines Unternehmens eine wichtige Funktion ... “ ([Port89], S. 225) einnehmen. Muster sind ein Ansatz für die

[1] Nicht zuletzt hängt eine solche Einordnung vom Einsatzgrad einer Technologie ab. Der Einsatzgrad von Mustern kann jedoch dadurch, dass die durchzuführende Studie Nichtnutzer von Mustern ausschließt, nicht ermittelt werden.

Einsparung von Kosten und können dadurch, dass sie die Entwicklung qualitativ hochwertiger Software unterstützen auch als Mittel zur Differenzierung gegenüber Wettbewerbern dienen (vgl. Abschnitt 4.2). Obwohl ein Reservoir an Mustern sowie die generelle Nutzung von Mustern durchaus als Wettbewerbsvorteil gesehen werden kann, werden Muster auf zahlreichen Konferenzen (PLoP, EuroPLoP, u.Ä.) und in zahlreichen Artikeln und Büchern publiziert (vgl. [Mar+98], S. X).

Die in Mustern beschriebene Technologie stellt einen Wert für ein Unternehmen dar. Neben dem Schutz dieses Wertes vor der Diffusion zu potenziellen Konkurrenten sollten Muster auch unter dem Aspekt der Vermarktung analysiert werden. Der Wert, den Unternehmen diesen Aspekten zuordnen, wird anhand der empirischen Studie untersucht.

4.4 Implikationen für die empirische Studie

Ausgehend von den in Abschnitt 4 behandelten Themen werden hier die sich daraus ergebenden Untersuchungsgebiete der empirischen Studie aufgestellt.

Aus der Betrachtung von Mustern im Rahmen des Wissenszyklus nach NONAKA ergibt sich die Frage, wie Muster in einer Organisation eingeführt werden und wie Muster innerhalb einer Organisation entdeckt werden (Frage 2C, 2D). Daneben werden Fragen nach der organisationalen Unterstützung der individuellen Nutzung von Mustern sowie der Entdeckung neuer Muster, beispielsweise denkbar in Form monetärer Vorteile, in den Fragebogen aufgenommen (Frage 2E, 2F).

Um einen deskriptiven Überblick über die Erfahrungen mit der Anwendung von Mustern zu gewinnen, wird versucht, die in der Literatur gefundenen Vor- und Nachteile der Nutzung von Mustern zu bestätigen. Es besteht die Möglichkeit, zusätzliche Vor- und Nachteile anzugeben. Eine weitere Frage dient dazu, potenziell problematische Gebiete bei der Anwendung von Mustern ausfindig zu machen (Frage 6A, 6D).

Eine häufig angeführtes Problem neuer Technologien ist das Problem, das die an diese Technologie geknüpften Erwartungen nicht erfüllt werden. Deshalb wird in der Literatur über Muster häufig davor gewarnt, zu hohe Erwartungen an die Nutzung von Mustern aufzubauen (vgl. [ScSt95], S. 12; [Paul99]; [Bec+01]). Den Erfolg dessen wird eine der Fragen des Fragebogens klären (Frage 6B).

Inwiefern die generelle Anwendung von Mustern bzw. die Nutzung eines organisationsspezifischen Reservoirs an Mustern als Wettbewerbsfaktor gesehen wird, wird anhand der Umfrage untersucht (Frage 7A bis 7E).

Letztlich wird analysiert, ob und in welcher Form sich ein Markt für Muster, beispielsweise ähnlich einem Markt für Softwarekomponenten entwickeln könnte (Frage 7F, 7G).

5. Studie über den Einsatz musterbasierter Systementwicklung in der Praxis

5.1 Zusammenfassung der Analysegebiete

Die jeweils am Ende der Kapitel angegebenen Implikationen für die empirische Untersuchung lassen sich in folgenden sieben Analysegebieten zusammenfassen. Jedes dieser Analysegebiete lässt sich dann innerhalb des Fragebogens für die Befragten nachvollziehbar wiederfinden.

1. Das Verständnis von Mustern

Ziel dieses Analysegebietes ist es, möglichst ein einheitliches Verständnis von Mustern zu erreichen, das dann in Zukunft als Grundlage für eine Abgrenzung des Musterbegriffs gegenüber anderen verwandten Konzepten dienen kann.

2. Arten der Verbreitung von Mustern in der Organisation

Dieses Analysegebiet beschreibt zunächst das zahlenmäßige Verhältnis zwischen Musternutzern und Musterproduzenten. Zusätzlich wird beschrieben, wie Muster in eine Organisation eingeführt werden, wie sie in einer Organisation entdeckt werden und wie dies durch die Organisation, z.B. in Form von Belohnungssystemen, unterstützt wird. Da dies eventuell von der Organisationsform beeinflusst wird, schließt dieses Analysegebiet die Frage nach der Organisationsform ein.

3. Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung

Neben der Darstellung, inwiefern die Nutzung von Mustern innerhalb der unterschiedlichen Aktivitäten verbreitet ist, lässt sich aus den gestellten Fragen auch ableiten, inwiefern dies

durch die Nutzung unterschiedlicher Arten automatisierter Musterwerkzeuge unterstützt wird. Daneben lässt sich der Bekanntheitsgrad von Mustern verschiedener Musterarten vergleichen.

4. Musterproduktion und Musternutzung im Vorgehensmodell

Hier wird untersucht, ob die Erstellung bzw. Nutzung von Mustern Bestandteil des Vorgehensmodells der Organisation ist. Zusätzlich beschreibt die Untersuchung, inwiefern Ansätze zur Überwindung der Schwächen des Wasserfallmodells mit der Nutzung von Mustern korrespondieren.

5. Die Charakterisierung von Mustern – Musterbeschreibungsformen und Musterorganisationsschemata

Neben einer Analyse der in der Praxis genutzten Musterbeschreibungsformen und Musterorganisationsschemata wird hier untersucht, ob die existierenden Musterorganisationsformen ausreichend sind. Die Wichtigkeit der unterschiedlichen Beziehungsarten zwischen Mustern für ein zu entwickelndes Musterorganisationsschema wird analysiert.

6. Gesammelte Erfahrungen mit Mustern

Um die in der Literatur zu findenden positiven und negativen Erfahrungen zu bestätigen, werden diese in den Fragebogen aufgenommen. Daneben wird die Ausprägung potenzieller Probleme bei der Anwendung von Mustern untersucht. Die Notwendigkeit von drei Musterarten, die sich zwar aus den Aktivitäten der Systementwicklung ergeben würden, jedoch in der Literatur nicht wiederzufinden sind, wird analysiert. Der Erfolg einer in der Literatur zu findenden Aussage, die Erwartungen an den Nutzen der Anwendung von Mustern möglichst gering zu halten, wird überprüft.

7. Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil

Ob Faktoren wie das Verhältnis der Nutzung eigenerstellter Muster im Vergleich zur Nutzung fremder Muster sowie die Bereitschaft, Muster mit Personen außerhalb der eigenen Organisation zu diskutieren und zu teilen, einen Einfluss darauf haben, dass die generelle Nutzung von Mustern oder das Reservoir von organisationsspezifischen Mustern als

Wettbewerbsvorteil gesehen wird, wird hier analysiert. Daneben wird auch untersucht, ob Studien über Einsparmöglichkeiten durch die Nutzung von Mustern im Unternehmen durchgeführt wurden und ob dies eventuell dazu führt, dass ein potenzieller Markt für Muster gesehen wird.

5.2 Zusammenhang zu anderen Studien

Neben verschiedenen Erfahrungsberichten, zumeist aus einzelnen Unternehmen, sind in der Fachliteratur nur wenige Studien, die sich mit der Anwendung von Mustern beschäftigen, zu finden.

Eine der Studien, deren Ergebnisse unter anderem in die Aufstellung der Vor- und Nachteile der Nutzung sowie der Eigenschaften von Mustern innerhalb dieser Arbeit einfließen, fasst die Erfahrungen von sechs Unternehmen zusammen. Eine der dargestellten Erfahrungen in der Nutzung von Mustern, die von allen Unternehmen geteilt wurde, ist, dass Muster ein gutes Kommunikationsmittel sind. Dies korrespondiert mit der in dieser Arbeit aufgeführten Eigenschaft N9 bzw. E9.

Daneben führt die Studie noch zwei weitere Eigenschaften von Mustern an, die allen Unternehmen gemeinsam sind – (1) Muster werden aus funktionierenden Entwürfen extrahiert und (2) Muster fassen die Essenz einer Entwurfsentscheidung zusammen. Während die erste Eigenschaft mit der Eigenschaft E11 korrespondiert, lässt sich die Essenzeigenschaft am ehesten in der Eigenschaft E7 wiederfinden (vgl. [Bec+01]).

Ziel der Studie im Rahmen dieser Arbeit ist unter anderem, diese Erkenntnisse auf eine breitere Untersuchungsbasis zu stellen.

Eine Studie in Form eines Experiments, durchgeführt an der Universität Karlsruhe, beschäftigt sich mit der Hypothese, dass die Dokumentation von Mustern im Programmkode die Wartung beschleunigen und die Qualität der Wartung erhöhen. Beide Hypothesen konnten bestätigt werden (vgl. [Pre+97a]).

Eine ähnliche Studie an der Washington University, St. Louis, die versuchte, in einem gleichgearteten Experiment beide Hypothesen nachzuweisen, konnte nur die erste belegen. Die zweite Hypothese konnte weder bestätigt noch widerlegt werden (vgl. [Pre+97b]).

Eine weitere Studie versucht die Hypothese zu beweisen, dass die Kenntnis von Entwurfsmustern die Kommunikation innerhalb eines Teams effektiver gestaltet. Diese Eigenschaft von Mustern korrespondiert mit der Eigenschaft der Nutzung von Mustern N6 sowie der Eigenschaft E9. Die Hypothese konnte bestätigt werden (vgl. [Unge00]).

Zu beachten ist, dass sich sämtliche oben aufgeführten Studien ausschließlich mit der Anwendung von Entwurfsmustern beschäftigen, wohingegen die Studie im Rahmen dieser Arbeit sich explizit auch auf andere Arten von Mustern bezieht.

5.3 Aufbau und Durchführung der Studie

Ziel der Untersuchung ist es, einen deskriptiven Überblick über die Nutzung und das Verständnis von Mustern in der Praxis zu geben. Die Grundgesamtheit besteht dabei aus individuellen Softwareentwicklern, die sich erkennbar mit Mustern beschäftigen. Dies schließt sowohl Musterkonsumenten als auch Musterproduzenten ein. Explizit ausgeschlossen sind Softwareentwickler, die sich in keiner Weise mit Mustern beschäftigen.

Eine derartige Definition der Grundgesamtheit garantiert, dass ausgehend von den gewonnenen Erkenntnissen Aussagen zu den in Abschnitt 5.1 aufgeführten Analysegebieten gemacht werden können. Aussagen, die Musternutzer mit Nichtnutzern vergleichen, können nicht getroffen werden. Dazu zählen Erkenntnisse zum Verbreitungsgrad der Nutzung von Mustern unter Softwareentwicklern oder aber auch zu charakterisierenden Eigenschaften, die Musternutzer von Nichtnutzern unterscheiden, z.B. zum Grad der Nutzung objektorientierter Konzepte.

Die empirische Untersuchung wurde in Form eines durch die Teilnehmer auszufüllenden Fragebogens durchgeführt. Der Fragebogen wurde dabei als eine Applikation im WWW umgesetzt. Dies garantiert die weltweite Erreichbarkeit von Teilnehmern. Der Fragebogen ist in Englisch verfasst.

Die Größe der Grundgesamtheit lässt sich nur schwer bestimmen. Jedoch liegt sie hinsichtlich Menge und Kosten außerhalb der Grenzen, die eine Vollerhebung für sinnvoll erscheinen lässt. Damit ergibt sich die Notwendigkeit, ausgehend von der Analyse einer Stichprobe Schlussfolgerungen auf die Eigenschaften der Grundgesamtheit zu ziehen. Eine solche Stichprobe sollte aus zufällig ermittelten Personen der Grundgesamtheit bestehen. Potenzielle

Teilnehmer wurden dabei per Email angeschrieben. Die Namen und Emailadressen der Teilnehmer wurde dabei aus öffentlich zugänglichen Teilnehmerlisten von Konferenzen, Vorlesungen und Seminaren, die sich mit der Anwendung von Mustern beschäftigen, zusammengestellt. Zusätzlich wurden Autoren von Artikeln und Büchern zum Thema Muster angeschrieben. Als weitere Kontaktform wurden Mailinglisten gewählt, die sich eindeutig dem Thema Muster widmen.

Der Fragebogen wurde in zwei Phasen getestet. In einer ersten Phase haben drei Personen aus unterschiedlichen Firmen den Fragebogen selbstständig ausgefüllt, um in einem anschließenden Interview mögliche Probleme zu identifizieren. Im Anschluss wurde die WWW-Version des Fragebogens erstellt. Diese Version wurde mit drei weiteren Probanden unter realistischen Bedingungen getestet. Während des Ausfüllens standen die Befragten unter Beobachtung und waren aufgefordert, mögliche Probleme laut zu äußern. In einem anschließenden Interview wurden diese Probleme näher spezifiziert und in den Fragebogen eingearbeitet.

Die im Fragebogen verwendete englische Sprache war für keinen der Befragten die Muttersprache. Alle Probanden gaben an, keine Probleme mit dem Verständnis der Fragen zu haben. Alle Probanden waren mit Mustern vertraute Softwareentwickler. Aus den durchgeführten Tests ergab sich eine durchschnittliche Bearbeitungszeit von 20 Minuten. Zu bemerken ist, dass diese stark von der Motivation der Teilnehmer abhängt.

Unter dem Gesichtspunkt der Distanz zwischen Interviewer und befragter Person ähneln WWW-basierte Umfragen am ehesten schriftlichen Befragungen, insbesondere im Vergleich mit Telefoninterviews. Geringe Kosten sowie die Möglichkeit, Teilnehmer aus geografisch weit verbreiteten Gebieten zu befragen, sprechen damit für WWW-Umfragen (vgl. [CzBl95], S. 31ff). Im direkten Vergleich mit Fragebögen, die per Post versendet werden, erweisen sich WWW-Umfragen als ideale Technologien, um internationale Studien durchzuführen (vgl. [Dill98], S. 18). Diese Gründe waren ausschlaggebend für die Wahl des oben beschriebenen Vorgehens. Daraus ergeben sich jedoch auch einige potenzielle Probleme hinsichtlich folgender vier von DILLMAN unterschiedenen Fehlerkategorien: Abdeckung der Grundgesamtheit, Auswahl der Stichprobe, Nichtantworten und Messfehler (vgl. [Dil+98], S.2).

Abdeckung der Grundgesamtheit

"The Internet user population is clearly not representative of the popluation at large" ([Forr98], Kapitel 8). Dies kann möglicherweise zu einer Vorselektion von Teilnehmern führen, die die definierte Grundgesamtheit nicht repräsentativ abdeckt und damit die Ergebnisse einer Befragung verfälscht oder sogar unbrauchbar macht. Im Falle der durchgeführten Studie ist dieses Problem weniger akut, insofern davon auszugehen ist, dass nahezu 100 Prozent der Softwareentwickler einen Internetzugang haben. Zwei zusätzliche Kriterien, bei denen jedoch im Falle der durchgeführten Studie auch von einer annähernd 100-prozentigen Abdeckung durch die Grundgesamtheit ausgegangen werden kann, werden von BALLANTYNE aufgeführt: das Ausmaß der Vertrautheit im Umgang mit Computern und der Grad der Akzeptanz Aufgaben am Computer zu bearbeiten (vgl. [Ball00]).

Auswahl der Stichprobe

Wie auch bei Fragebögen, die per Post versendet werden, spielt die Auswahl der Stichprobe eine besondere Rolle für die Signifikanz der Ergebnisse. Die Stichprobe sollte eine zufällige Auswahl von Individuen der Grundgesamtheit entsprechen. Dies wurde versucht, durch die oben beschriebene Auswahl von Emailadressen und Mailinglisten zu erreichen. "It is noteworthy that newsgroups and discussion lists can assist researchers in generating a screened sample. Internet users that participate in newsgroups and discussion mailing lists do have similar interests …" ([Forr98], Kapitel 8). Die Qualität der Stichprobe wird jedoch zusätzlich von der Anzahl der Nichtantworten bestimmt.

Nichtantworten

Ein hohe Anzahl von Nichtantworten schränkt die Interpretierbarkeit der Ergebnisse insofern ein, als dass die Charakteristika der Befragten, die sich entschieden haben, den Fragebogen nicht auszufüllen, komplett von den Eigenschaften der tatsächlich Befragten unterscheiden kann. Dies wird im Allgemeinen als Self-Selection bezeichnet (vgl. [Med+99], S. 4; [Forr98], Kapitel 8).

Messfehler

Die Darstellung des Fragebogens auf dem Bildschirm der Befragten hängt sowohl von Bildschirmgröße als auch von Einstellungen des benutzten WWW-Browsers ab. Hinweise für die Gestaltung von Fragebögen speziell für das Internet geben DILLMAN et al. beispielsweise in [Dil+98] und [DiBo01].

Ein weitere Möglichkeit für Messfehler besteht darin, dass für einen Großteil der Befragten die im Fragebogen verwendete englische Sprache nicht die Muttersprache ist. Der unterschiedliche kulturelle Hintergrund der Teilnehmer beeinflusst das Verständnis der im Fragebogen verwendeten Begriffe. "The Internet is global; hence differences in language, slang and culture may lead to the misinterpretation of questions or even offend the respondent" ([Forr98], Kapitel 8). Um mögliche Beeinflussung durch unterschiedliche kulturelle Hintergründe zu vermeiden, wurde der Fragebogen mit Softwareentwicklern unterschiedlicher Abstammung getestet.

6. Auswertung der Studie

6.1 Allgemeine Angaben

Die ersten Anschreiben in Form von Emails wurden am 5. und 6. September 2001 versendet. Das Anschreiben ist in Anhang 1 abgedruckt. Ein Erinnerungsschreiben wurde am 20. September 2001 verschickt. Insgesamt wurden 759 Personen angeschrieben. Die Anzahl der Ausfälle beträgt 184 und ist hauptsächlich auf ungültige Emailadressen zurückzuführen. Die Anzahl der erreichten Personen ist damit 575. Ähnlich dem Versenden per Post ist auch bei der Versendung per Email nicht zu garantieren, dass die angeschriebene Person auch tatsächlich erreicht wurde. Aus dem Quotienten der erreichten Personen und der Anzahl angeschriebener Personen errechnet sich die Ausschöpfungsquote. Im Falle dieser Studie beträgt diese 76 Prozent.

Parallel zu den persönlichen Anschreiben per Email wurde ein Link zum Fragebogen in Mailinglisten, die sich mit der Anwendung von Mustern beschäftigen, publiziert. Eine erste Email an 15 Mailinglisten wurde am 18. September 2001 versendet. Ein weiteres Erinnerungsschreiben an die Mailinglisten wurde am 9. Oktober 2001 versendet. Die Umfrage wird am 21. Oktober 2001 abgeschlossen. Eine Übersicht über die ausgewählten Mailinglisten

ist in Anhang 15 zu finden. Aufgrund des Charakters von Mailinglisten lässt sich für diesen Kontaktweg keine Ausschöpfungsquote angeben.

Die im Folgenden dargestellten Kennzahlen und Analysen basieren auf dem zum 28. September 2001 zur Verfügung stehenden Datenmaterial.

Insgesamt wurde der Fragebogen von 151 Personen vollständig ausgefüllt. Dabei erfolgte der erste Kontakt mit der Studie für 103 Befragte mittels persönlichen Anschreibens. Daraus lässt sich eine Rücklaufquote, als Quotient der Rückmeldungen und der Anzahl der erreichten Personen, von 18 Prozent berechnen. Aufgrund dessen, dass eine Vielzahl von Abonnenten mehrere Mailinglisten abonniert hat, lässt sich für die per Mailinglisten angeschriebenen Personen keine Rücklaufquote berechnen.

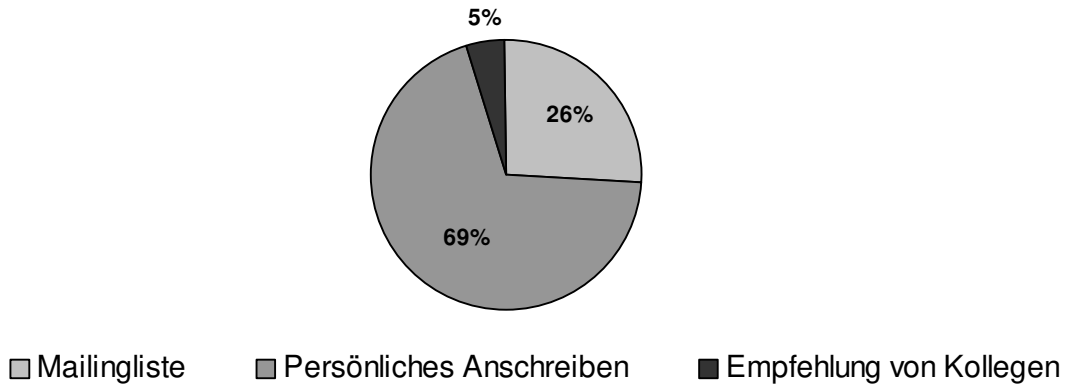

Abbildung 15: Wie erfolgte der erste Kontakt mit den Befragten (Frage 8D)?

Diejenigen, die den Fragebogen nicht ausgefüllt haben, gaben folgende Gründe dafür an:

- Keine Zeit (Sieben Nennungen)
- Nur unregelmäßig Nutzung von Mustern (Eine Nennung)
- Befragter beschäftigt sich nicht mehr mit Mustern (Eine Nennung)
- Fehlendes Wissen, um Fragen zu beantworten (Eine Nennung)
- Generelle Abneigung gegen Fragebögen (Zwei Nennungen)

- Das Ausfüllen des Fragebogens ist zu langweilig (Eine Nennung).
- Komplizierter Einstieg in den Fragebogen, insbesondere Erstellung eines Logins (Eine Nennung)
- Link aus Email funktionierte nicht, Timeoutprobleme (Fünf Nennungen)
- Lange Ladezeiten der Internetseiten (Eine Nennung)

Die beiden zuletzt aufgeführten Punkte sind auf technische Probleme hinsichtlich Internetanbindung und Ausfällen der Server zurückzuführen.

Teilweise gaben direkt angeschriebene Personen an, den Fragebogen nicht selbst ausfüllen zu können, jedoch die Einladung zum Ausfüllen an entsprechende Personen weitergeleitet zu haben. Insgesamt sieben Befragte gaben an, den Fragebogen aufgrund der Empfehlung eines Kollegen ausgefüllt zu haben.

Weit mehr als die Hälfte der Befragten arbeiten in einer Firma. In etwa ein Drittel derjenigen, die den Fragebogen ausgefüllt haben, arbeiten bzw. studieren an einer Universität. Die meistgenannte Antwort in der Kategorie Sonstiges war die Arbeit in einem Forschungszentrum (4 Nennungen). Daneben wurde diese Kategorie auch von Personen gewählt, die sich nicht eindeutig einer der drei anderen Kategorien zuordnen ließen, beispielsweise von Studenten, die parallel zu ihrem Studium in einer Firma arbeiten.

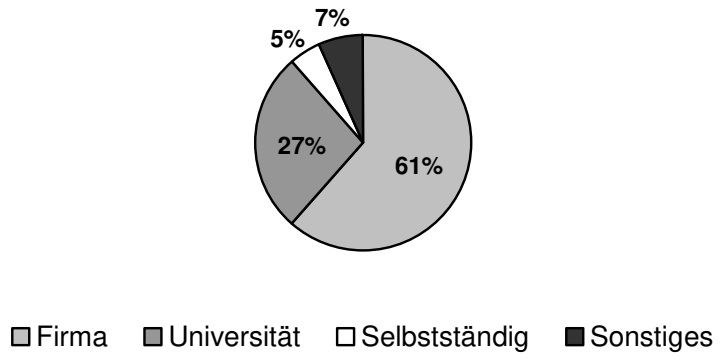

Abbildung 16: Art der Organisation, in der die Befragten arbeiten (Frage 8C)

6.2 Signifikanz der abgeleiteten Aussagen

Die Mehrheit der folgenden Betrachtungen basiert auf der Analyse dichotomer Merkmale, die im Fragebogen erfragt wurden. Beispielhaft sei hier die Signifikanz der folgenden Aussagen anhand einer statistischen Analyse der Frage 1B des Fragebogens dargestellt. Während sich diese Frage beispielsweise auf eine einzige Eigenschaft bezieht, nämlich die Sichtweise auf Algorithmen, werden in anderen Fragen mehrere Merkmale gleichzeitig abgefragt. So ergeben sich aus Frage 1A beispielsweise 15 Merkmale – ein Merkmal pro abgefragter Mustereigenschaft.

Inwiefern lassen sich die Aussagen, die sich aus der Analyse der Stichprobe ergeben, auf die Grundgesamtheit ausdehnen? Die Statistik bedient sich dazu des Signifikanzniveaukonzeptes. Die im Folgenden gemachten Aussagen sollen anhand eines Signifikanzniveaus von 0,95 analysiert werden (α = 0,05). Dies bedeutet, dass die gemachten Aussagen mit einer Wahrscheinlichkeit von 95 Prozent auf die Grundgesamtheit zutreffen. Die Größe des Konfidenzintervalls errechnet sich gemäß folgender Formel:

$$d = \frac{c \bullet \sqrt{\hat{p} \bullet (1 - \hat{p})}}{\sqrt{n}}$$

$$[\hat{p} - d ; \hat{p} + d] = Konfidenzintervall\ für\ \hat{p}$$
$$c = (1 - a/2) - Fraktilswert\ der\ Standardnormalverteilung$$
$$n = Größe\ der\ Stichprobe$$
$$\hat{p} = Anteilswert\ in\ der\ Stichprobe$$

Im Falle der Frage 1B gaben 22 Prozent der Befragten an, Algorithmen als Muster anzusehen. In zwei der 151 ausgefüllten Fragebögen wurde diese Frage nicht ausgefüllt. Damit reduziert sich die Größe der Stichprobe auf 149. Für große Stichproben lässt sich der verwendete Fraktilswert aus der Standardnormalverteilung ableiten. Für ein Signifikanzniveau von 0,95 ergibt sich c zu 1,96. Nach Einsetzen der Werte ergibt sich damit folgendes Konfidenzintervall:

$$d = \frac{1{,}96 \bullet \sqrt{0{,}22 \bullet (1 - 0{,}22)}}{\sqrt{149}} = 0{,}067$$

$$Konfidenzintervall\ für\ Anteilswert\ \hat{p}: [0{,}22 - 0{,}067 ; 0{,}22 + 0{,}067]$$

Damit lässt sich anhand des gesammelten Datenmaterials folgende Aussage ableiten: Mit einer Wahrscheinlichkeit von 95 Prozent sehen zwischen 15 und 29 Prozent der Softwareentwickler, die sich mit Mustern beschäftigen, Algorithmen als Muster an.

Wie anhand der Formel zu sehen ist, hängt die Größe des Konfidenzintervalls vom gemessenen Anteilswert innerhalb der Stichprobe sowie von der Größe der Stichprobe ab. Im ungünstigsten Fall, einem Anteilswert von 50 Prozent, würde sich das Konfidenzintervall auf +/- 0,08 vergrößern.

Die Größe der Stichprobe wird durch die Anzahl der Nichtantworten reduziert. Dabei bewegt sich die Anzahl der Ausfälle für einzelne Fragen innerhalb des Fragebogens zwischen 1 und 17. Rein rechnerisch hat dies einen zu vernachlässigenden Einfluss auf die Größe des Konfidenzintervalls.

Wie schon in Abschnitt 5.3 aufgeführt, beeinflusst eine hohe Anzahl von Ausfällen jedoch die Interpretierbarkeit der Ergebnisse, insofern Nichtantworten auf tieferen Gründen als dem puren Überlesen der Frage basieren können. Dazu zählen beispielsweise das Missverstehen der Frage, das Verweigern der Antwort oder fehlendes Wissen, um die Frage zu beantworten. Im Falle einer großen Anzahl von Nichtantworten (größer fünf Prozent, bzw. acht Nichtantworten) auf eine spezifische Frage wird im Folgenden darauf hingewiesen (vgl. [CzBl95], S. 181f, S. 229).

6.3 Betrachtungen zu den einzelnen Analysegebieten

6.3.1 Das Verständnis von Mustern

Ziel dieses Abschnittes ist es, ein einheitliches Verständnis von Mustern zu erreichen. Während in Abschnitt 2.2.2 Muster anhand der theoretisch erarbeiteten Mustereigenschaften E1 bis E15 von anderen Ansätzen der Wiederverwendung abgegrenzt wurden, bilden diese Eigenschaften im Folgenden die Grundlage für eine pragmatische Abgrenzung von Mustern anhand des Verständnisses in der Praxis.

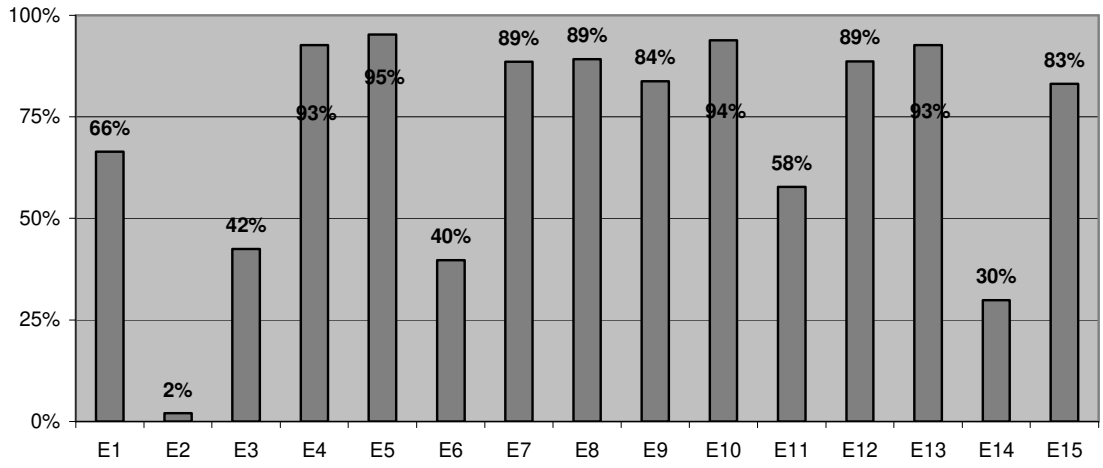

Abbildung 17: Welche Eigenschaften charakterisieren den Begriff Muster (Frage 1A)?[1]

Neun der 15 erfragten Mustereigenschaften werden von mehr als 80 Prozent der Befragten als charakterisierend für Muster gesehen. Anhand dieser Eigenschaften lässt sich folgende pragmatische Abgrenzung von Mustern ableiten:

Ursprung von Mustern

Ein Muster erscheint regelmäßig immer wieder (E4) und wurde schon mehr als einmal erfolgreich angewandt (E10). Es fixiert gesammeltes Expertenwissen (E8). Muster sind unabhängig von objektorientierten Konzepten (E15).

Musterbeschreibung

Ein Muster beschreibt ein Problem-Lösungs-Paar (E12) sowie die Kräfte bzw. Umstände, durch die es in seiner Lösung beschränkt ist (E13).

Musteranwendung

Ein Muster kann wieder und wieder angewandt werden (E5) und verändert sich bei jeder Anwendung, das generelle Schema wird jedoch beibehalten (E7). Muster vereinheitlichen die Kommunikation von Experten (E9).

[1] Eine zusammenhängende Auflistung der Eigenschaften E1 bis E15 ist in Anhang 16 zu finden.

Interessanterweise wird die Eigenschaft der Vollkommenheit als Charakteristikum von Mustern von fast allen Befragten abgelehnt. Dies widerspricht der in dieser Arbeit verwendeten Charakterisierung von Mustern. Für die Eigenschaften E1, E3, E6, E11 und E14 liegt in der Praxis keine klar mehrheitliche Sichtweise vor. Insofern eignen sich diese Eigenschaften nur schwer für eine Abgrenzung von Mustern.

Mit 22 Prozent sieht nur ein geringer, aber nicht zu vernachlässigender Teil der Befragten Algorithmen als Muster an. Dies spricht dafür, dass sich der Musterbegriff nur schwer von anderen Ansätzen der Wiederverwendung abgrenzen lässt.

In Abschnitt 2.2.2 wurden die Verwendung des Musterbegriffs innerhalb dieser Arbeit gegen Algorithmen abgegrenzt. Bei einer Interpretation der folgenden Ergebnisse ist zu berücksichtigen, dass der Musterbegriff durch den Einschluss von Algorithmen von den Befragten teilweise anders ausgelegt wird.

6.3.2 Arten der Verbreitung von Mustern in der Organisation

96 Prozent der Befragten gaben an, Muster zu benutzen. Dieser hohe Anteil liegt hauptsächlich daran, dass die Grundgesamtheit als Softwareentwickler, die sich mit Mustern beschäftigen, definiert wurde. Ein weiterer Grund für diesen hohen Anteil könnte die Allgemeinheit des Begriffs Musternutzung sein. Im Gegensatz dazu benutzen nur 81 Prozent der Softwareentwickler Muster über die individuelle Nutzung hinaus, beispielsweise in der Kommunikation innerhalb des Teams oder in der Dokumentation der durchgeführten Aktivitäten, obwohl gerade dies immer wieder als Hauptvorteil der Nutzung von Mustern angeführt wird (vgl. [ScSt95], S.12; [Bec+01], S. 10; [Helm95], S. 339). 60 Prozent der Befragten gaben an, neue Muster zu schreiben. Diese Zahl erscheint hoch und ist möglicherweise durch die Art der Auswahl der Stichprobe beeinflusst.

Weiterhin ist zu beachten, dass die Grundgesamtheit der Umfrage aus individuellen Softwareentwicklern besteht. Dies hat zur Folge, dass, obwohl sich die in diesem Abschnitt analysierten Fragen auf die Anwendung von Mustern innerhalb der Organisation beziehen, Schlussfolgerungen immer auf der Basis der Sichtweise einzelner Mitglieder der Organisation gezogen werden.

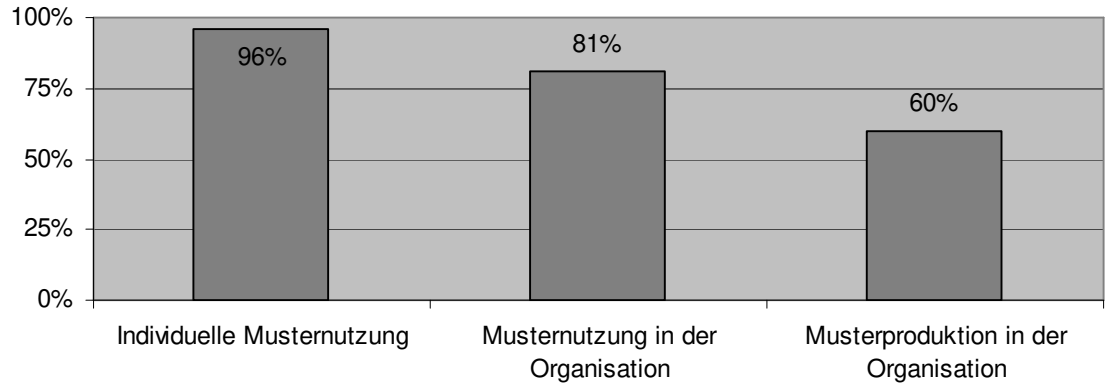

Abbildung 18: Anteil der Softwareentwickler, die Muster nutzen/entwickeln (Fragen 2A, 2B)

Welcher Methoden bedienen sich Softwareentwickler innerhalb ihrer Organisation, um bekannte Muster zu erlernen und neue Muster aufzufinden? Nahezu alle Befragten gaben an, in der einen oder anderen Form Muster zu erlernen. Dies deckt sich mit dem hohen Grad an Musternutzern. Die meistgenutzte Methode ist dabei das Selbststudium, welche von 94 Prozent der Befragten benutzt wird.

Auffällig ist die häufige Angabe informeller Kommunikationswege unter Sonstiges (24 Nennungen). Genannt werden die individuelle Beratung durch einen Mentor, spontane Diskussionen über Muster sowie Learning-by-Doing. Letzteres wird durch die konkrete Forderung nach dem Einsatz bestimmter Muster innerhalb eines Projektteams ausgelöst. Diese informellen Methoden werden teilweise durch interne Mailinglisten unterstützt. Weiterhin wurden angeführt: der Besuch von Konferenzen über Muster (sechs Nennungen) sowie das Lernen neuer Muster in System Reviews (vier Nennungen).

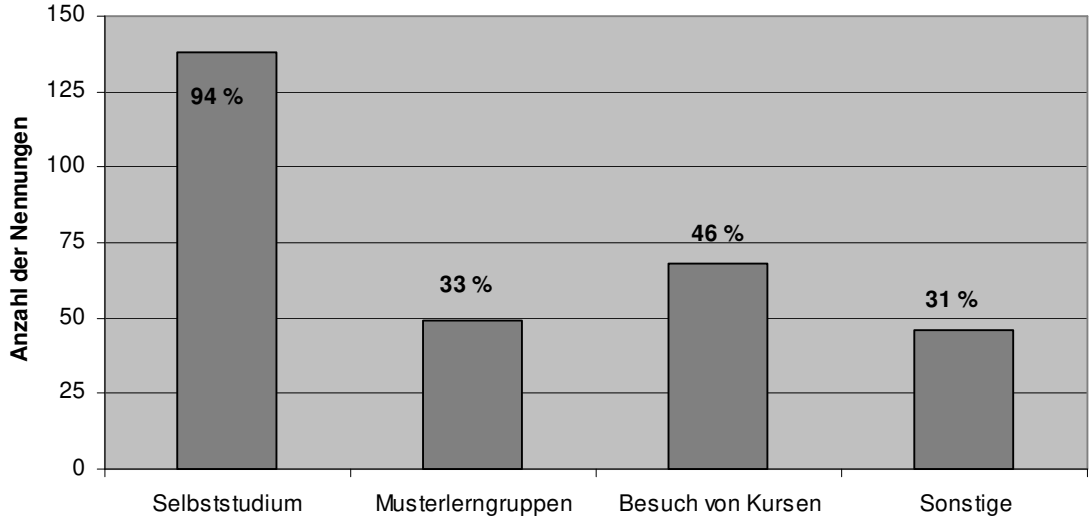

Abbildung 19: Einführung neuer Muster in die Organisation (Frage 2C, Mehrfachantworten möglich)[1]

Die Produktion von Muster, also die Suche nach neuen Musterideen, deren Bewertung sowie deren Umsetzung werden anhand der Frage 2D analysiert. Diese Frage wurde von acht der Befragten nicht beantwortet. 85 Prozent gaben an, mindestens eine Form der Suche nach neuen Mustern innerhalb ihrer Organisation zu nutzen. Dieser Wert liegt über dem in Frage 2B ermittelten Wert von 60 Prozent. Selbst unter der Annahme, dass diejenigen, die diese Frage unbeantwortet ließen, keine Form der Musterproduktion nutzen, reduziert sich der Wert von 85 Prozent nur auf 80 Prozent. Zu beachten ist, dass Frage 2B danach fragt, ob Muster innerhalb der Organisation geschrieben werden, während 2D nach Methoden, mit denen neue Muster in einer Organisation entdeckt werden, fragt. Zu erwarten wäre ein in etwa gleicher Anteil in beiden Fragen. Worin der zu beobachtende Unterschied begründet liegt, bleibt offen.

Der Großteil der Softwareentwicklern, die sich mit Mustern beschäftigen, entdeckt Muster im Verlauf der täglichen Arbeit. Während diese Methode ein eher informelles Vorgehen nahe legt, werden, wenn auch in einem geringeren Ausmaß, innerhalb der Organisationen der befragten Softwareentwickler auch formale Methoden wie Experteninterviews, externe Musterseminare sowie System Reviews genutzt. Wie schon bei der Analyse der Arten der

[1] Angaben in Prozent berechnen sich auf Basis der Anzahl der Befragten, die mindestens eine der genannten Arten benutzen (147 Nennungen).

Einführung neuer Muster in eine Organisation besteht der Hauptteil der freien Antworten unter Sonstiges aus informellen Vorgehensweisen (14 Nennungen). Dazu zählen die Diskussion mit anderen Autoren von Mustern, gelegentliche Meetings sowie einfach nur die Beschäftigung mit fertigen Entwürfen bzw. Programmkode. Drei Nennungen, alle von Befragten aus einer Universität, erwähnen explizit die Forschung nach neuen Mustern.

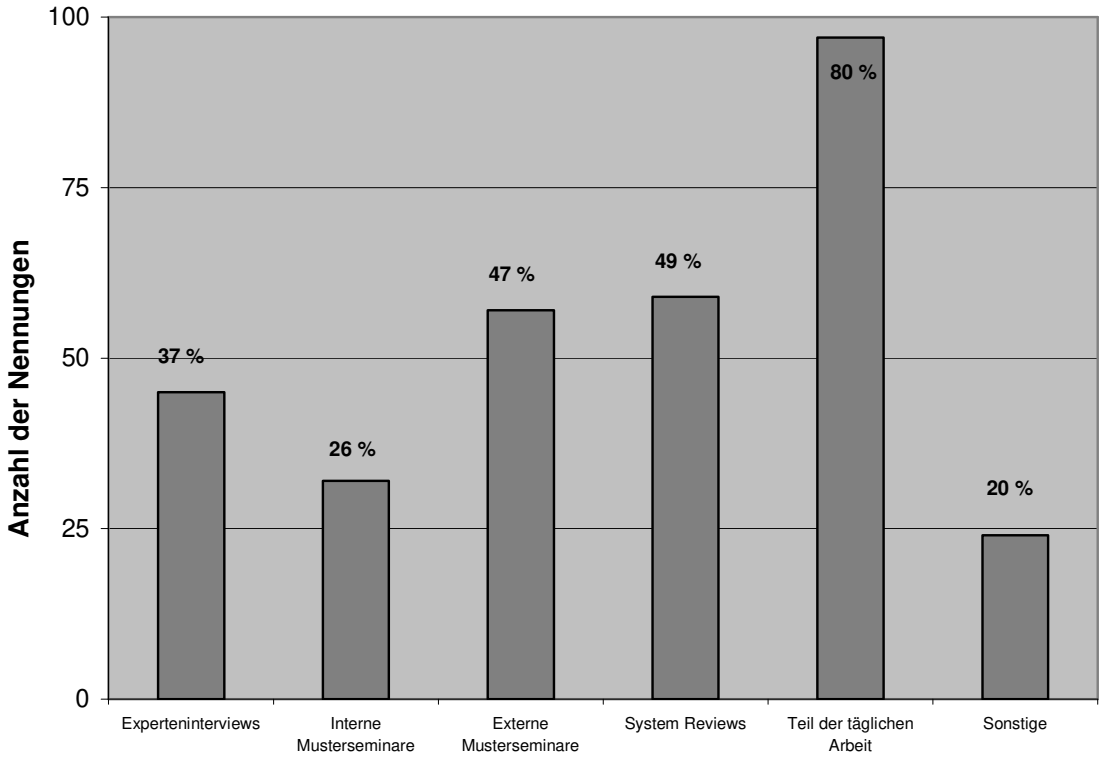

Abbildung 20: Arten der Suche nach neuen Musterideen (Frage 2D, Mehrfachantworten möglich)[1]

Die Fragen 2E und 2F beziehen sich auf Formen der organisatorischen Unterstützung von Musternutzung und Musterproduktion. 48 Prozent der Befragten gaben an, dass die Nutzung von Mustern in ihrer Organisation unterstützt wird. Werden nur diejenigen Softwareentwickler betrachtet, die angaben, Muster individuell zu nutzen, ändert sich dieser Wert nur geringfügig. Dies ist darin begründet, dass fast alle Befragten Muster individuell benutzen. Wie zu erwarten erhöht sich der Anteil organisatorischer Unterstützung, wenn nur

[1] Angaben in Prozent berechnen sich auf Basis der Anzahl der Befragten, die mindestens eine der genannten Arten benutzen (121 Nennungen).

Softwareentwickler betrachtet werden, die Muster auch über die individuelle Nutzung hinaus, beispielsweise in der Dokumentation, anwenden. Allerdings liegt der beobachtete Anteil mit 56 Prozent immer noch weit unter dem Maximum. Das heißt, dass 44 Prozent der Entwickler Muster offenbar freiwillig über die individuelle Nutzung hinaus anwenden, obwohl dies innerhalb der Organisation nicht gefördert wird.

Hinsichtlich der Produktion von Mustern, also dem Entdecken, Bewerten und Schreiben neuer Muster, gaben nur 27 Prozent der Befragten an, dass dies innerhalb ihrer Organisation unterstützt wird. Werden nur diejenigen Softwareentwickler betrachtet, in deren Organisation auch tatsächlich Muster geschrieben werden, erhöht sich dieser Wert auf 41 Prozent.

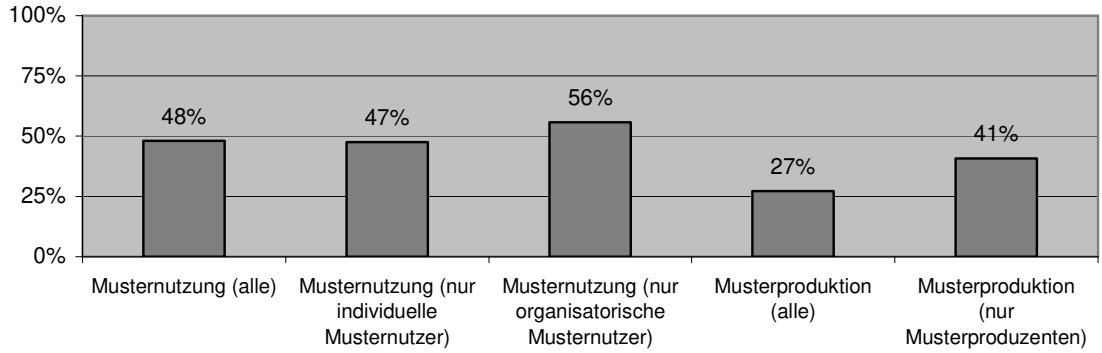

Abbildung 21: Organisatorische Unterstützung von Musternutzung und Musterproduktion (Fragen 2E, 2F)[1]

Von besonderem Interesse sind die freien Darlegungen, da beide Fragen als offene Fragen konzipiert wurden und keine Kategorisierung möglicher Antworten vorgenommen wurde. Die Nutzung von Mustern innerhalb der Organisation wurde insgesamt sechs mal als absolut notwendig angesehen. D.h., dass ein gewisser inoffizieller Druck besteht, Muster anzuwenden. 14 Befragte gaben an, dass in ihrer Organisation der Gebrauch von Mustern offiziell oder inoffiziell empfohlen wird. Dies geschieht beispielsweise in Form von Empfehlungen in System Reviews, oder aber in informellen Diskussionen zwischen Softwareentwicklern. 16 Befragte gaben an, dass die Nutzung von Mustern durch die

[1] Aufgrund der Vorauswahl, basierend auf den Antworten zu den Fragen 2A und 2B, gehen in die Berechnung der Anteilswerte, nach Ausschluss der Nichtantworten, für Säule zwei nur 139 Antworten, für Säule drei nur 115 Antworten und für Säule fünf nur 86 Antworten ein. Für Säule drei ergaben sich, kombiniert aus den Fragen 2A und 2E, insgesamt 8 Nichtantworten.

Förderung des Erlernens von Mustern begünstigt wird. Dazu zählen die schon oben beschriebenen Methoden (vgl. Abbildung 19). Die formelle Unterstützung, beispielsweise innerhalb des Softwareprozesses, wurde viermal angeführt. Ebenfalls viermal wurde die Unterstützung durch separate (Projekt-)Teams genannt. Diese sind unter anderem dafür verantwortlich, die Nutzung von Mustern innerhalb der Organisation zu fördern. Drei der Befragten führten die Existenz einer Leitfigur an: „The big boss loves patterns." Auch die Publikation von Mustern im Intranet wird als Förderung der Nutzung von Mustern angesehen (vier Nennungen). Die denkbare Unterstützung in Form monetärer Vorteile wurde nicht genannt. Die Vergabe von Auszeichnungen wurde einmal angeführt: „Awards have recognised early successes in Patterns usage."

Auch die Produktion von Mustern wird nicht durch monetäre Vorteile unterstützt. Zu bemerken ist jedoch, dass die Möglichkeit, nach neuen Mustern zu suchen und diese zu publizieren, auch als Belohnung, wenn auch in nicht monetärer Form, angesehen werden kann. Die Publikation der Ergebnisse von Forschungsaktivitäten in Fachzeitschriften wurde von fünf Befragten angegeben. Drei der fünf sind in Universitäten beschäftigt, wobei es sich einmal um ein akademisches Forschungsprojekt in Kooperation mit einer Firma handelt. Als Anreiz für die Suche nach neuen Mustern wurden weiterhin die Publikation bzw. Diskussion im Intranet (vier Nennungen), die informelle Unterstützung der Suche nach neuen Muster (vier Nennungen) sowie die Unterstützung durch spezieller Trainer bzw. Teams (zwei Nennungen) angeführt. Sieben Antworten bezogen sich auf die schon in der Analyse der Frage 2D diskutierten Arten des Entdeckens neuer Muster.

6.3.3 Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung

Wie schon in Abschnitt 6.3.2 bei der Analyse der Antworten zu Frage 2A wird bei der Auswertung des Nutzungsgrades der einzelnen Musterarten die Anwendung von Mustern auf individueller und organisatorischer Basis unterschieden. Während für die individuelle Nutzung von Mustern die Nichtantworten im verträglichen Bereich liegen, lag die Anzahl der Nichtantworten für die Musternutzung in der Organisation je nach Musterart zwischen 10 und 17. Die naheliegende Vermutung, dass diejenigen, die schon bei Frage 2A die organisatorische Nutzung von Mustern verneint haben, den entsprechenden Teil der Frage 3A übersprungen haben, lässt sich nicht bestätigen.

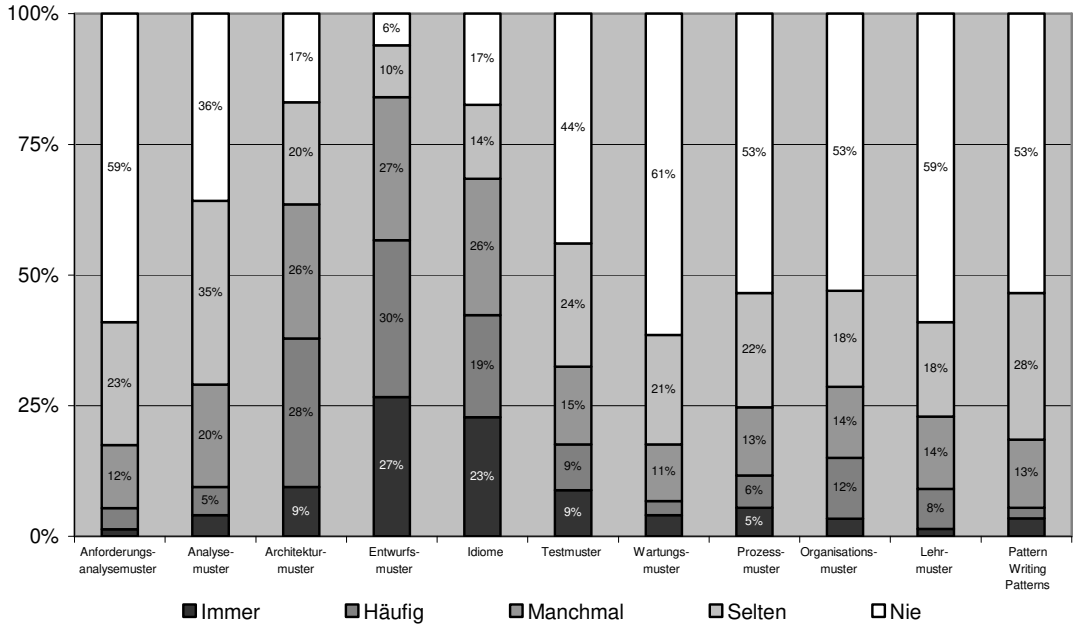

Abbildung 22: Häufigkeit der individuellen Nutzung von Mustern (Frage 3A, alle Befragten)

Aufgrund der Kategorisierung der Häufigkeit der Nutzung der einzelnen Musterarten lässt sich keine Rangordnung zwischen den einzelnen Musterarten erstellen. Aus Abbildung 22 ist jedoch zu entnehmen, dass sich die Nutzung von Mustern auf Analysemuster, Architekturmuster, Entwurfsmuster, Idiome und Testmuster konzentriert. Für diese Musterarten gab mehr als die Hälfte der Befragten an, diese zumindest gelegentlich anzuwenden. Am häufigsten werden Entwurfsmuster benutzt. Dies lässt sich wohl damit erklären, dass die Ursprünge von Mustern in der Entwicklung von Entwurfsmustern liegen (vgl. [Copl96b]). Dies könnte jedoch auch daran liegen, dass unter den Befragten die Softwareentwickler dominieren, die sich mit dem Entwurf von Softwaresystemen beschäftigen. Dieser Gedanke wird weiter unten in diesem Abschnitt noch einmal aufgegriffen.

Die Häufigkeit der Nutzung von Anforderungsanalysemustern, Wartungsmustern, Prozessmustern, Organisationsmustern, Lehrmustern und Pattern Writing Patterns ist in etwa gleich und bei weitem geringer als die der anderen fünf Musterarten.

Die Verteilung der Häufigkeit der Anwendung der einzelnen Musterarten in der Organisation für Zwecke der Dokumentation oder der Kommunikation mit anderen Entwicklern zeigt in etwa das gleiche Bild wie die individuelle Nutzung. Zu bemerken ist eine für alle Musterarten

abnehmende Häufigkeit. Dies deckt sich mit der in Abschnitt 6.3.2 bei der Analyse der Frage 2A gefundenen gleichgearteten Beobachtung für die Nutzung von Mustern im Allgemeinen.

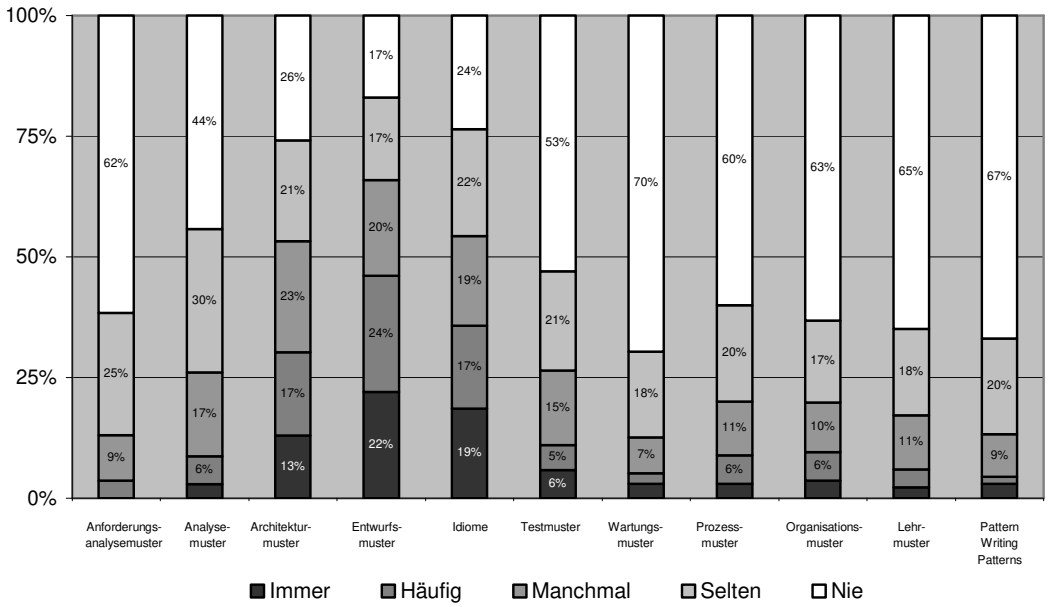

Abbildung 23: Häufigkeit der Nutzung von Mustern in der Organisation (Frage 3A, alle Befragten)

Eine mögliche Dominanz von Softwareentwicklern, die sich mit einer spezifischen Aktivität, z.B. dem Entwurf, beschäftigen, könnte einen ungewollten Effekt auf die Nutzung spezifischer Musterarten haben. Für phasenabhängige Muster lässt sich dies anhand eines Vergleiches, basierend auf den Antworten zu Frage 8A, untersuchen. Während gemäß Abbildung 22 nur 6 Prozent der Befragten angaben, Entwurfsmuster nie zu benutzen, reduziert sich diese Zahl auf 2 Prozent, bezieht man nur Softwareentwickler ein, die sich auch tatsächlich mit dem Entwurf beschäftigen (vgl. Abbildung 24). Für Analysemuster, Architekturmuster, Entwurfsmuster und Idiome bewegen sich die Abweichungen in einem ähnlichen Rahmen. Festzustellen ist, dass für Anforderungsanalyse-, Test- und Wartungsmuster durch die Beschränkung auf Softwareentwickler, die sich auch mit der jeweiligen Aktivität beschäftigen, die Anzahl derjenigen, die zumindest gelegentlich die entsprechenden Muster nutzen, um zehn bis 14 Prozentpunkte ansteigt.

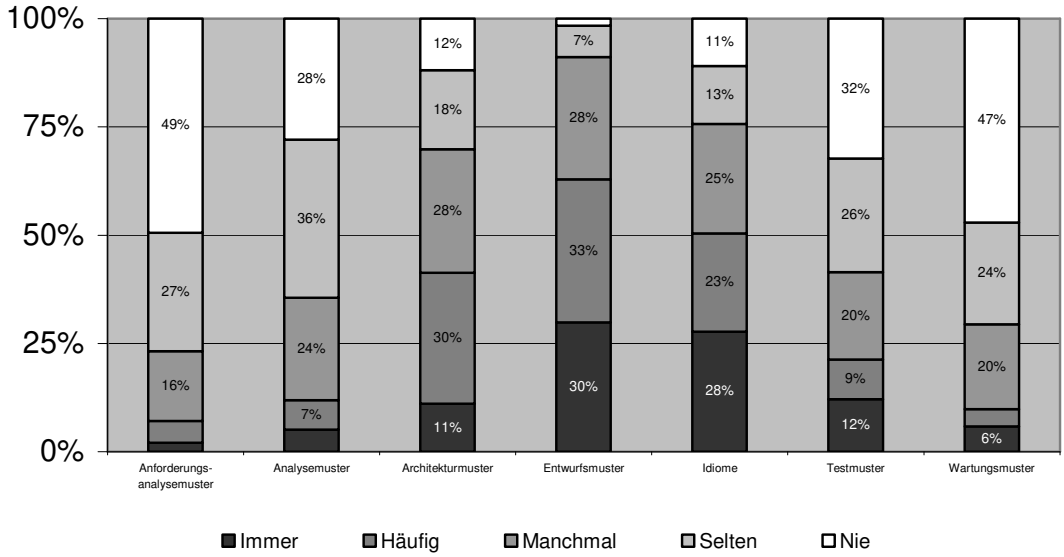

Abbildung 24: Häufigkeit der individuellen Nutzung von Mustern (Frage 3A, 8A, Auswahl der Befragten)[1]

Die Vertrautheit mit den Mustern der verschiedenen Musterarten wurde anhand der Anzahl der bekannten Muster der jeweiligen Kategorie gemessen. Es ist anzumerken, dass die Anzahl verfügbarer Muster innerhalb der einzelnen Musterarten variiert. Da die Kenntnis eines Musters nichts über den individuellen Grad der Vertrautheit mit einem Muster aussagt, wurde innerhalb der Frage die Kenntnis des Problems und seiner generellen Lösung als Bedingung dafür aufgeführt, ein Muster als vertraut anzusehen.

Wie schon bei der Häufigkeit der Nutzung lässt sich für phasenabhängige Muster eine Dominanz von Analysemustern, Architekturmustern, Entwurfsmustern, Idiomen und Testmustern feststellen (vgl. Abbildung 25). Für diese fünf Arten gaben mehr als 50 Prozent der Befragten an, mindestens ein Muster zu kennen. Obwohl Prozessmuster, Organisationsmuster und Pattern Writing Patterns von weniger als der Hälfte der Befragten überhaupt benutzt werden, so kennen doch ca. 60 Prozent derjenigen, die den Fragebogen ausgefüllt haben, mindestens ein Muster der jeweiligen Musterart.

[1] Für die Darstellung in diesem Diagramm wurden für die jeweiligen Musterarten nur diejenigen Befragten berücksichtigt, die sich auch tatsächlich mit der entsprechenden Aktivität beschäftigen. Daraus ergibt sich eine Beschränkung auf phasenabhängige Muster. Aufgrund dieser Vorselektion reduziert sich die Anzahl der auswertbaren Antworten. Von links nach rechts beträgt diese Anzahl, unter Abzug der Nichtantworten: 99, 118, 126, 124, 119, 99 und 51

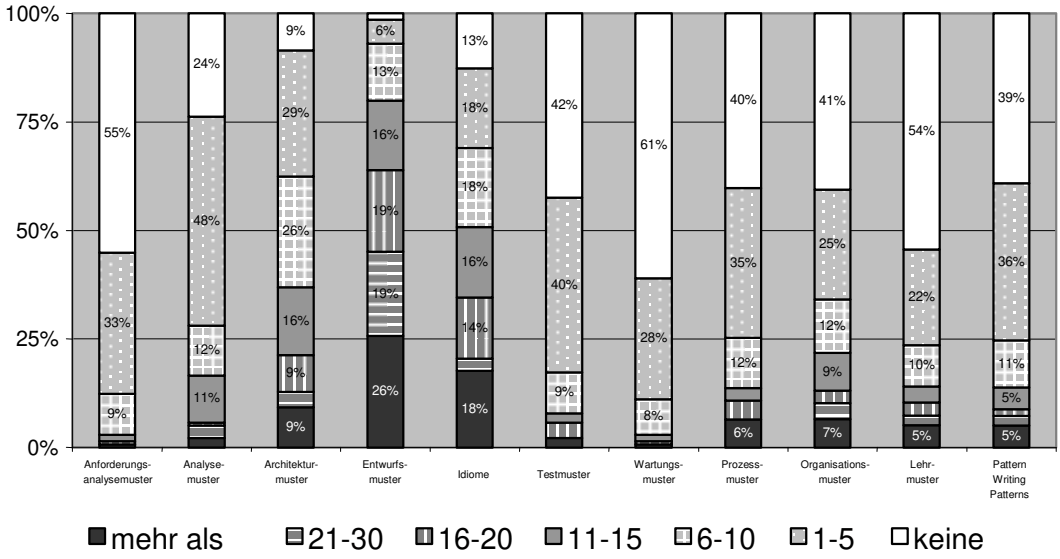

Abbildung 25: Vertrautheit mit Mustern (Frage 3B, alle Befragten)

Wie schon bei der Betrachtung der Häufigkeit der Nutzung der einzelnen Musterarten, lässt sich für phasenbasierte Muster die Analyse der Vertrautheit auf diejenigen beschränken, die sich mit den entsprechenden Aktivitäten beschäftigen.

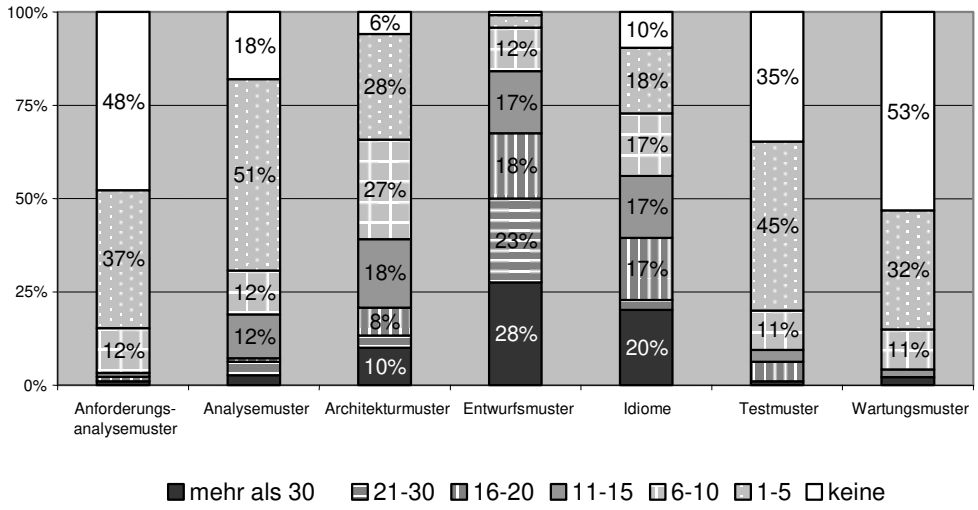

Abbildung 26: Vertrautheit mit Mustern (Frage 3B, 8A, Auswahl der Befragten)[1]

[1] Für jede Musterart wurde ein Vorselektion der Befragten vorgenommen. Nur diejenigen, die die entsprechende Aktivität auch durchführen werden betrachtet. Damit reduziert sich die Anzahl auswertbarer Antworten. Von links nach rechts beträgt diese Anzahl, nach Abzug der Nichtantworten: 92, 111, 120, 120, 114, 95 und 47. Für Anforderungsanalysemuster, Analysemuster und Architekturmuster ergeben sich jeweils acht Nichtantworten.

Im Gegensatz zum Ergebnis der entsprechenden Analyse für die Nutzung von Mustern, lässt sich durch die Beschränkung auf diejenigen, die die entsprechende Aktivität auch durchführen, kein erheblicher Unterschied herbeiführen. Allenfalls für Anforderungsanalysemuster und Wartungsmuster ergibt sich eine Differenz von sieben bzw. acht Prozentpunkten für die Anzahl derjenigen, die kein Muster der entsprechenden Art kennen.

Wie stark werden die in Abschnitt 3.1.3 vorgestellten Automatisierungsansätze in der Praxis genutzt? Zunächst ist festzustellen, dass musterspezifische Werkzeuge von den Befragten in allen Aktivitäten, für die Muster bekannt sind, mehr oder weniger stark angewendet werden.

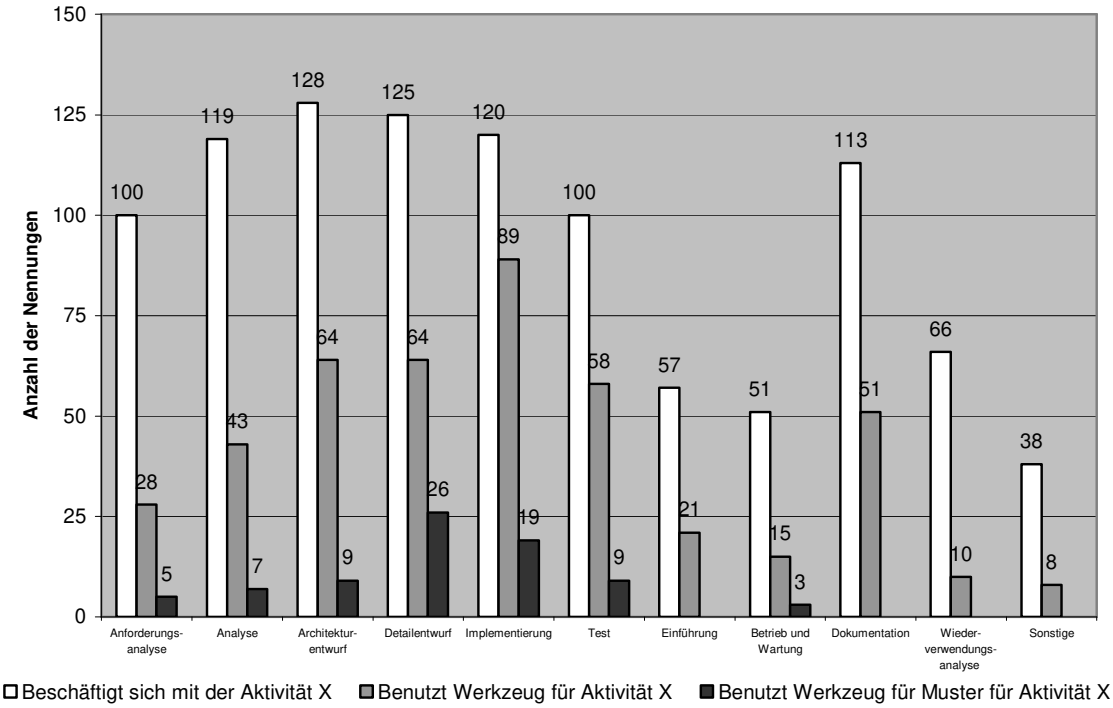

Abbildung 27: Anzahl der Nutzer von Werkzeugen für Mustern (Frage 3C)[1,2]

[1] Für die Aktivitäten Einführung, Dokumentation und Wiederverwendungsanalyse sind keine Muster und demzufolge auch keine derartigen Automatisierungsansätze bekannt. Der Platz an entsprechender Stelle im Diagramm ist nicht belegt (= 0). Das gleiche gilt für die Kategorie Sonstige, obwohl beispielsweise im Falle von Managementaktivitäten Prozess- bzw. Organisationsmuster hier einzuordnen wären.

[2] Für die Analyse der generellen Werkzeugunterstützung wurden nur die Antworten der Befragten einbezogen, die auch angaben die entsprechende Aktivität durchzuführen. Für die Berechnung der Anzahl derjenigen, die musterspezifische Werkzeuge nutzen, wurden wiederum nur die Befragten einbezogen, die auch angaben überhaupt Werkzeuge für diese Aktivität zu nutzen.

Am weitesten verbreitet ist die Nutzung von automatisierten Hilfsmitteln im Detailentwurf und in der Implementierung mit 22 bzw. 16 Prozent. Automatisierungsansätze für die Anwendung von Mustern spielen in der Softwareentwicklung, mit Ausnahme von Detailentwurf und Implementierung, nur eine geringe Rolle. 40 Prozent derjenigen, die ein generelles Werkzeug innerhalb des Detailentwurfs benutzen, wenden auch einen der Automatisierungsansätze für Muster an. Für andere Aktivitäten ist dieser Prozentsatz weit geringer und liegt zwischen 14 und 21 Prozent. Mit Ausnahme der generellen Werkzeugunterstützung für Detailentwurf und Dokumentation (neun bzw. elf fehlende Antworten) sowie der musterspezifischen Werkzeugunterstützung für die Implementierung (neun fehlende Antworten) liegt die Anzahl der Nichtantworten unter dem Wert von acht.

In Abbildung 28 wird die Nutzung von Werkzeugen anhand der in Abschnitt 3.1.3 vorgestellten Automatisierungsansätze (Au1 bis Au4) näher beschrieben. Die Anzahl der Nichtantworten für die zugrundeliegenden Teilfragen lag zwischen neun und 14.

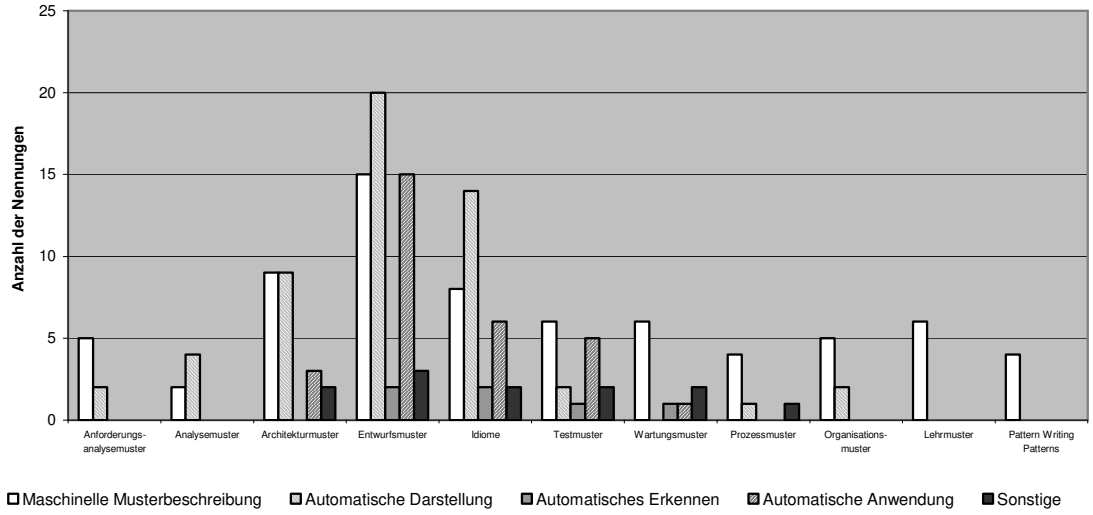

Abbildung 28:Anzahl der Nennungen der verschiedenen Arten der Automatisierung (Frage 3C, vergleiche Abbildung 27, Mehrfachnennungen möglich,)

Zu verzeichnen ist, dass das automatische Erkennen (Au3) von Mustern, beispielsweise in Quellkode, nur von geringer Bedeutung ist (maximal zwei Nennungen). Die automatisierte Anwendung von Mustern (Au4) konzentriert sich auf Entwurfsmuster (15 Nennungen). Hingegen findet die Nutzung maschineller Beschreibungsformen (Au1) auch über Entwurfsmuster hinaus Anwendung. Die meistgenannte Form der Automatisierung ist die

Darstellung von Mustern in Modellen, insbesondere gilt dies für Architekturmuster, Entwurfsmuster und Idiome.

Folgende Ansätze ließen sich von den Befragten nicht in die verwendete Klassifikation einordnen. Für Architekturmuster wurden die Unterstützung durch die verwendete Sprache sowie CVOPS genannt. Inwiefern CVOPS, ein virtuelles Betriebssystem für die Entwicklung von Protokollen, die Anwendung von Mustern unterstützt, bleibt offen (vgl. [VTT01]). Nennungen in der Kategorie Entwurfsmuster waren erneut die Unterstützung durch eine Sprache, CVOPS sowie Together Control Center (vgl. [Toge01]). Bei Idiomen wurden CVOPS sowie die Unterstützung durch die Programmierumgebung genannt. Für Testmuster wurden die Unterstützung durch ein spezielles Werkzeug für den Test sowie xUnit angeführt. XUnit ist ein von Beck entwickeltes Rahmenwerk für den Test von Software. In der Dokumentation des Rahmenwerkes wird von Beck ein musterbasierter Ansatz verfolgt (vgl. [Beck01]). Für Wartungsmuster wurden zwei Ansätze genannt: Sprachunterstützung, sowie spezielle Wartungssoftware. Eine Angabe bei Prozessmustern bezog sich auf eine interne Entwicklung.

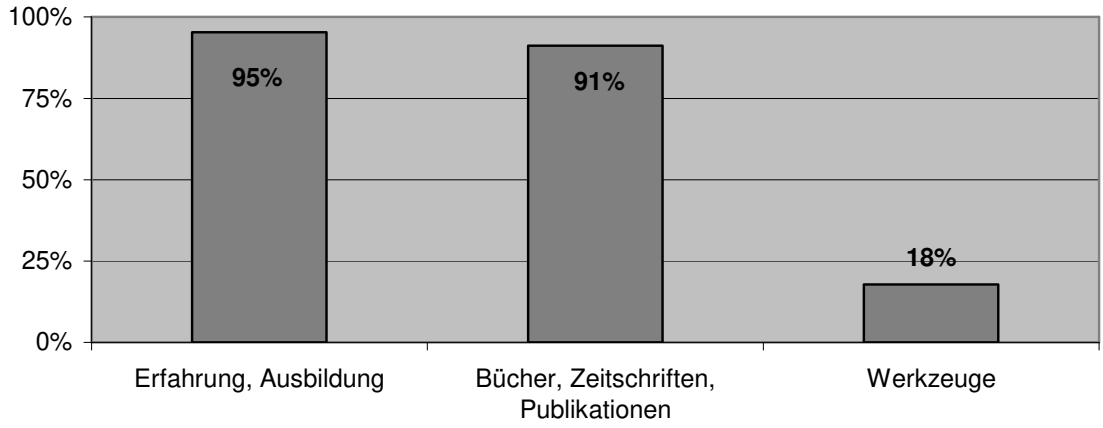

Abbildung 29: Wie werden Muster angewendet (Frage 3D)?

Die geringe Nutzung von Werkzeugen für die Anwendung von Mustern wird durch die Analyse der Frage 3D noch einmal bestärkt (vgl. Abbildung 29). Stattdessen werden Muster hauptsächlich auf Grundlage der gesammelten Erfahrungen bzw. direkt aus Büchern, Zeitschriften bzw. sonstigen Publikationen angewendet.

6.3.4 Musterproduktion und Musternutzung im Vorgehensmodell

Wie in Abbildung 30 zu sehen ist, wird nur in 63 Prozent der Organisationen, in denen Muster genutzt werden, auch ein Vorgehensmodell benutzt. Werden nur diejenigen Befragten berücksichtigt, die in einer Firma arbeiten, erhöht sich dieser Anteil auf 76 Prozent (72 auswertbare Antworten).

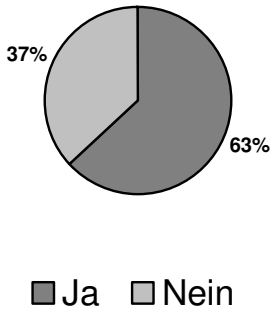

Abbildung 30: Wird ein Vorgehensmodell innerhalb der Organisation benutzt (Frage 4A, 2A)[1]

Die Frage 4A diente der Vorselektion der Befragten für die folgenden Auswertungen der Fragen 4B, 4C und 4D. In die Analysen werden nur diejenigen einbezogen, die angaben, für die Softwareentwicklung innerhalb ihrer Organisation ein Vorgehensmodell zu benutzen. Die Anzahl der auswertbaren Antworten reduziert sich damit auf 72.

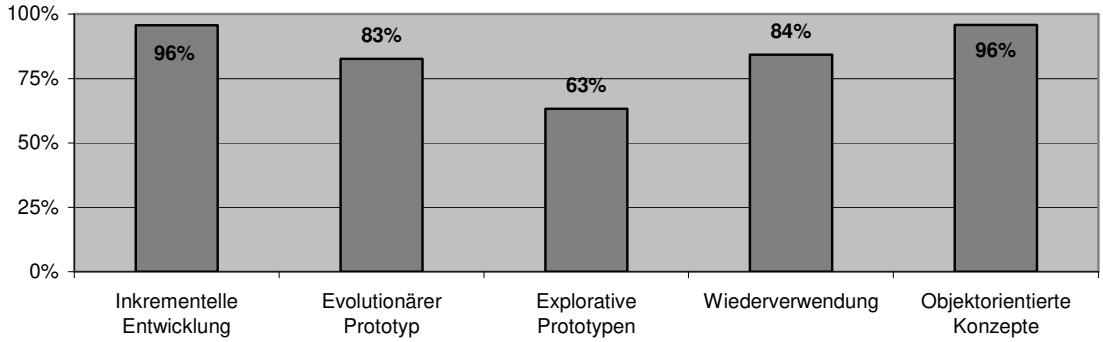

Abbildung 31: Anteil der Nutzung der Konzepte K1 bis K5 (Frage 4B, 4A, 2A, Mehrfachantworten möglich)

[1] Für diese Auswertung wurden nur diejenigen berücksichtigt, die Muster auch innerhalb ihrer Organisation anwenden. Die Anzahl der auswertbaren Antworten reduziert sich damit, nach Abzug der Nichtantworten auf 114.

Aus Abbildung 31 ist zu erkennen, dass sowohl die inkrementeller Entwicklung als auch objektorientierte Konzepte extrem häufig angewendet werden. Es ist zu vermuten, dass dies in Zusammenhang mit der Anwendung von Mustern steht. Die in der inkrementellen Entwicklung notwendige Nutzung von Modellen bzw. Quellkode vorheriger Inkremente könnte durch die Verwendung von Mustern vereinfacht werden (vgl. Abschnitt 3.2 sowie Abschnitt 4.2). Die häufige Nennung objektorientierter Konzepte könnte in der Dominanz objektorientierter Muster liegen. Die Nutzung explorativer Prototypen ist mit einem Anteil von 63 Prozent geringer ausgeprägt als der Anteil der anderen Konzepte. Wie schon in Abschnitt 3.2 angeführt, können die in Abbildung 31 dargestellten Sachverhalte nur als Indiz für die tatsächlichen Zusammenhänge dienen.

Ist die Anwendung von Mustern bzw. die Suche nach neuen Muster Bestandteil des Vorgehensmodells? Zu erkennen ist, dass dies nur selten der Fall ist. 34 Prozent der Befragten gaben an, dass die Nutzung von Mustern ein expliziter Teil des Vorgehensmodells ist. Im Vergleich dazu ist die Suche nach neuen Mustern, z.B. in Form von System Reviews, nur in 17 Prozent der Fälle Bestandteil des Vorgehensmodells.

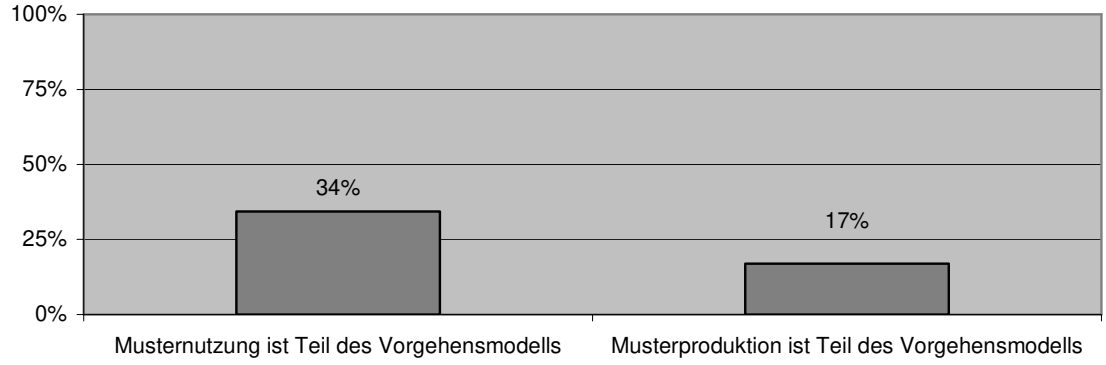

Abbildung 32: Anteil der Befragten, für die Musternutzung und Musterproduktion Bestandteil der Vorgehensmodells sind (Frage 4C, 4D, 4A, 2A)

6.3.5 Die Charakterisierung von Mustern – Musterbeschreibungsformen und Musterorganisationsschemata

Abbildung 33 zeigt den Anteil der Nutzung der verschiedenen Musterbeschreibungsformen. Zu bemerken ist zunächst, dass 95 Prozent der Befragten zumindest eine spezifizierte Musterbeschreibungsform benutzen. Dies liegt wohl darin begründet, dass die Darstellung

von Mustern in einer bestimmten, einheitlichen Form das Erlernen der Muster bzw. die Arbeit mit Mustern erleichtert. Die beiden aufgeführten formellen Beschreibungsformen werden dabei am häufigsten verwendet. Dahingegen spielen die Portlandform und die Beschreibungsform nach ALEXANDER, beides sind informelle Beschreibungsformen, nur eine untergeordnete Rolle. Die häufigste Nennung unter Sonstiges war die POSA-Form (sieben Nennungen). Diese formelle Musterbeschreibungsform wurde von BUSCHMANN et al. in [Bus+98] vorgestellt. In sechs Fällen wurden Adaptionen bestehender Musterbeschreibungsformen benutzt. Fünf Befragte gaben an, eine firmen- bzw. projektspezifische Form zu benutzen. In drei Antworten wurde explizit erwähnt, dass jeweils die am besten passende Musterbeschreibungsform benutzt wird. Als spezifische Formen wurden weiterhin aufgeführt (jeweils eine Nennung): BINDERS Test Design Pattern Template (vgl. [Bind00], S. 334ff) und die Musterbeschreibungsform nach MOWBRAY und MALVEAUX (vgl. [MoMa97], S. 68ff). Beides sind formelle Beschreibungsformen.

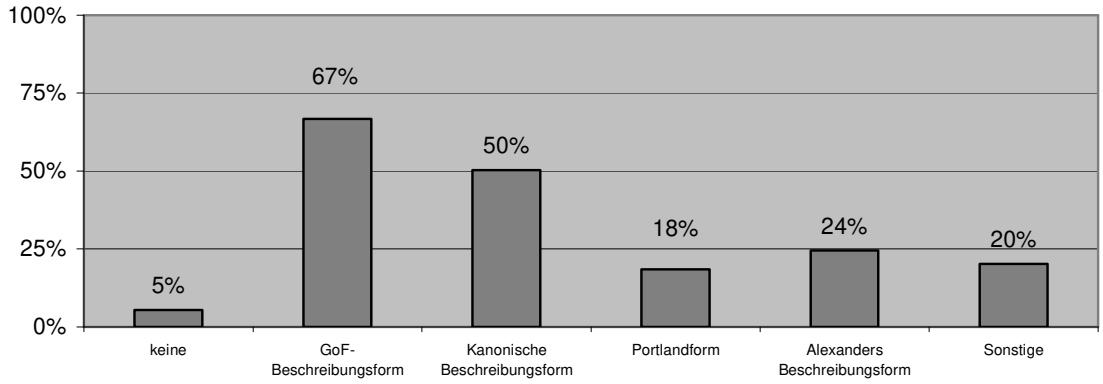

Abbildung 33: Anteil der Nutzung verschiedener Musterbeschreibungsformen (Frage 5A)

Während Frage 5A auf die Nutzung der verschiedenen Musterbeschreibungsformen abzielte, richtet sich Frage 5B auf die Nutzung von Musterorganisationsschemata. 32 Prozent der Befragten gaben an, keine spezifischen Musterorganisationsschemata zu verwenden (vgl. Abbildung 34). Die beiden von GAMMA et al. und BUSCHMANN et al. vorgeschlagenen Organisationsschemata werden von 44 bzw. 35 Prozent der Befragten angewendet. Andere Schemata spielen nur eine untergeordnete Rolle. Die häufigste Nennung in dieser Kategorie war die von Mustersprachen, einem Ansatz, bei dem Muster eines bestimmten Anwendungsgebietes zueinander in Beziehung gesetzt werden (sieben Antworten). Weiterhin

wurden genannt: eine hierarchiebasierte Klassifikation mit mehreren beschreibenden Parametern, die Klassifikation nach MOWBRAY und MALVEAUX (vgl. [MoMa97], S.46ff), RISINGS Musteralmanach sowie eine firmeninterne Lösung.

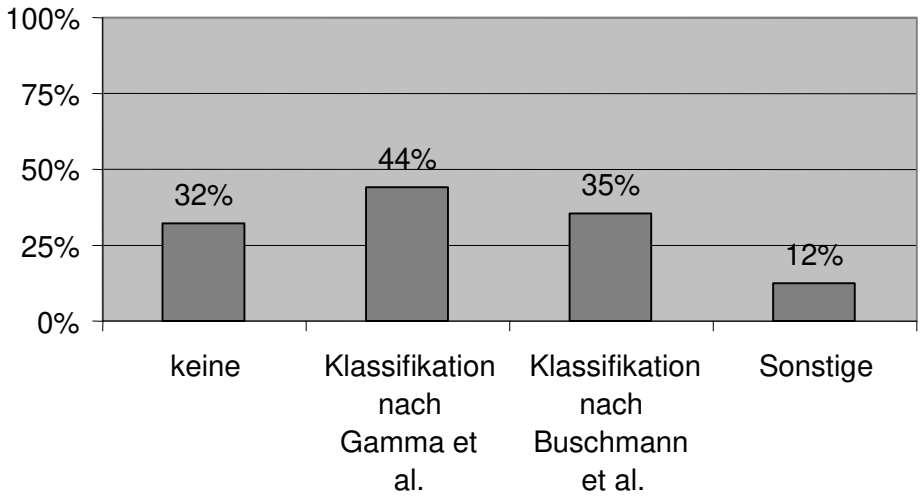

Abbildung 34: Anteil der Nutzung von Musterorganisationsschemata (Frage 5B)

Zwei Drittel der Befragten sind mit dem genutzten Musterorganisationsschema zufrieden. Dies bedeutet gleichzeitig, dass ein Drittel unzufrieden ist, was Spielraum für die Entwicklung besserer Organisationsschemata lässt.

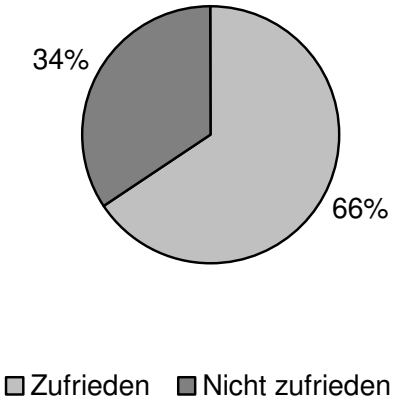

Abbildung 35: Zufriedenheit mit den genutzten Musterorganisationsschemata (Frage 5C)[1]

[1] Für die Auswertung dieser Frage wurden nur die Antworten ausgewertet, die auch tatsächlich ein Musterorganisationsschema anwenden. Dadurch reduziert sich die Anzahl der Antworten auf 93.

Interessant für die Entwicklung neuer Organisationsschemata für Muster ist, welcher Ansatz darin verfolgt werden sollte. Dies wird durch Frage 5D und 5E geklärt. Mit jeweils rund 40 Prozent sehen die meisten Befragten die Ansätze Mustersprache und Musterkatalog als geeignet an. Angaben unter Sonstiges waren: Ein Ansatz mit Metavariablen, ein Klassifikationshierarchie, ein multidimensionales Netz von Mustern, einfache textbasierte Suche, eine Kombination aus Musterkatalog und Mustersprachen sowie ein Klassifikationsschema. Drei Antworten gingen darauf ein, dass jeder Ansatz bestimmte Vor- und Nachteile hat und deshalb kein spezifischer Ansatz der Beste ist.

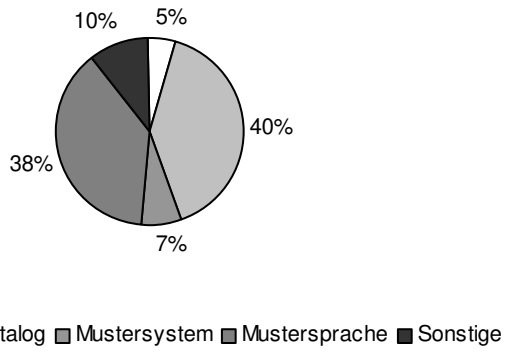

Abbildung 36: Welcher Ansatz für Musterorganisationsschemata wird bevorzugt (Frage 5D)?[1]

Um ein detaillierteres Bild darüber zu bekommen, welche Beziehungen zwischen Mustern in einem Musterorganisationsschema abgebildet werden sollten, wurde nach der Wichtigkeit verschiedener Musterbeziehungen gefragt. Aus der Wichtigkeit der einzelnen Beziehungen lässt sich nur schwer ein Rangordnung ableiten (vgl. Abbildung 37). Gemessen am Anteil derjenigen, die die jeweiligen Beziehung mit sehr wichtig bzw. ziemlich wichtig beurteilt haben, lässt sich feststellen, dass die Beziehungen, die die Anwendung von Musterketten thematisieren, als wichtiger angesehen werden. Ein zu entwickelndes Musterorganisationsschema sollte demzufolge unbedingt die beiden Beziehungen:

- Muster X benutzt Muster Y in seiner Lösung und
- Muster X verursacht Muster Y in einer anderen Aktivität

[1] Diese Frage wurde von acht Befragten unbeantwortet gelassen.

abbilden. Nichtsdestotrotz werden allen fünf aufgeführten Beziehungen von mindestens 86 Prozent der Befragten als zumindest „ein bisschen wichtig“ eingestuft.

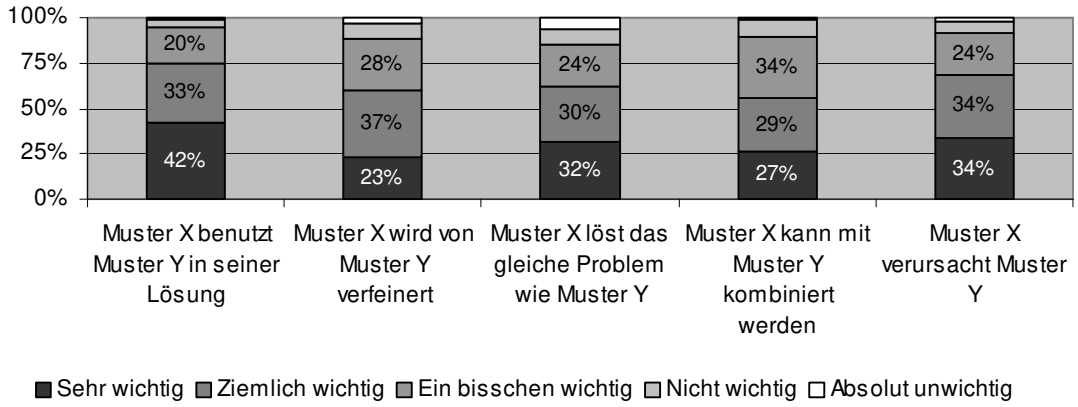

Abbildung 37: Wichtigkeit einzelner Beziehungen von Mustern für die Entwicklung eines Musterorganisationsschemas (Frage 5E)

6.3.6 Gesammelte Erfahrungen mit Mustern

Für die Analyse ordinalskalierter Daten bieten sich die Verwendung von Quartilswerten, wie z.B. dem Median, aber auch die Nutzung des Modalwertes an. Im Folgenden wird der Modalwert angewendet. Dabei ist zu beachten, dass dieser bei Ordinalskalen durch die Abgrenzung der Kategorien beeinflusst wird. Weiterhin kann der Modalwert nicht genutzt werden, um die Verteilung der Antworten zu beschreiben. Dies scheint mit Blick auf die tatsächlichen Streuungen der Antworten jedoch nicht problematisch, insofern keine bipolaren Verteilungen vorzufinden sind, bei denen im Extrem die Hälfte der Befragten einer Aussage voll zustimmt, während die andere Hälfte dem entschieden widerspricht.

Die bei weitem größte Zustimmung erhielt die Aussage, dass Muster die Kommunikation zwischen allen am Systementwicklungsprozess beteiligten Personen verbessern (N9). 57 Prozent der Befragten stimmten dem nachhaltig zu. Der zu Beginn von Abschnitt 6.3.5 festgestellte hohe Anteil derjenigen, die zumindest eine der verschiedenen Musterbeschreibungsformen anwenden (95 Prozent), steht damit im Einklang, insofern die Nutzung einheitlicher Beschreibungsformen ebenfalls die Kommunikation unterstützt. Eine wohlüberlegte Namensgebung für Muster, wie sie in Abschnitt 3.1.2 analysiert wurde, verbessert die Kommunikation zwischen Softwareentwicklern. Unter den freien Antworten zu Frage 6A war eine Bemerkung hinsichtlich der negativen Auswirkungen auf die Kommunikation zu finden:

„Patterns may even cause more harm than good, if some members of the team are unfamiliar with the patterns being used and talked about by others."

Für die Aussagen N1 bis N5, N10, N13, N20 und N21 fällt der Modalwert in die Kategorie „Ich stimme zu." Damit lässt sich für diese ebenfalls eine breite Zustimmung ableiten. Schwer zu beurteilen ist die Bedeutung der Eigenschaften N7, N11, N16 bis N18 sowie N22, da für diese der Modalwert in der Kategorie „Ich tendiere dazu zuzustimmen" liegt. Gleiches gilt für die Aussagen N6, N8, N12, N14 und N15. Der Modalwert für diese Aussagen lag in der Kategorie „Ich tendiere dazu zu widersprechen." Am ehesten lässt sich dies als eine schwache Unterstützung bzw. Ablehnung der entsprechenden Aussagen interpretieren. Einzig die Aussage, dass Muster den Aufwand zur Laufzeit durch die Einführung einer zusätzlichen Ebene der Abstraktion erhöhen (N8), wurde von mehr als zwei Dritteln der Befragten mehr oder weniger stark abgelehnt.

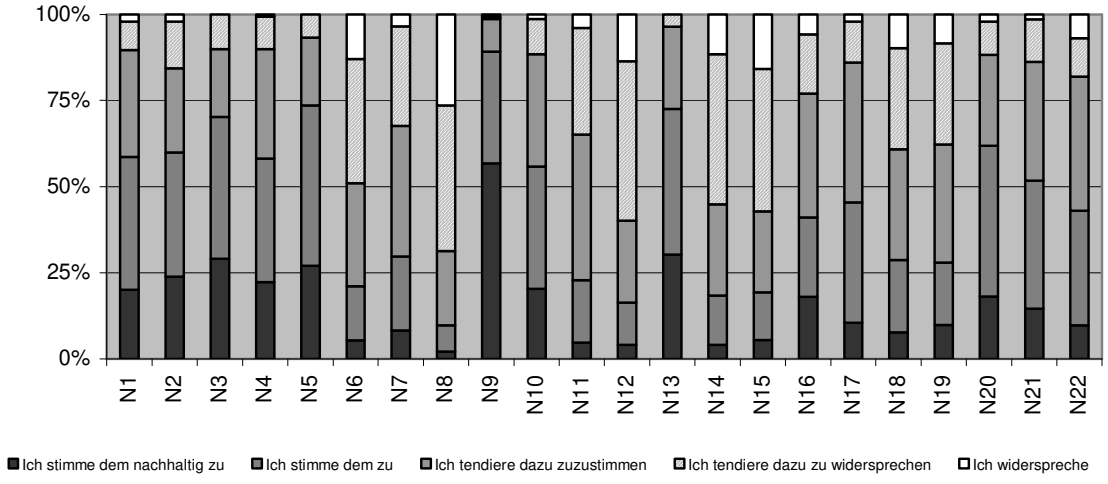

Abbildung 38: Vorteile und Nachteile bei der Nutzung von Mustern (Frage 6A)[1]

Von der Möglichkeit, freie Antworten zu den Vor- und Nachteilen der Nutzung von Mustern zu geben, wurde rege Gebrauch gemacht. Eine Übersicht über diese Angaben ist in Anhang 18 zu finden.

[1] Die exakten Prozentangaben sind Anhang 17 zu entnehmen. Eine Auflistung der freien Antworten ist in Anhang 18 zu finden. Die Anzahl der Nichtantworten für diese Frage lag zwischen 2 und 12.

In etwa die Hälfte der Befragten war mit der Nutzung von Mustern sehr zufrieden. Die Erwartungen an Muster wurden übertroffen. Keiner der Befragten gab an, unzufrieden mit der Nutzung von Mustern zu sein. Letzteres ist nicht weiter überraschend, insofern die Grundgesamtheit aus an Mustern interessierten Softwareentwicklern besteht.

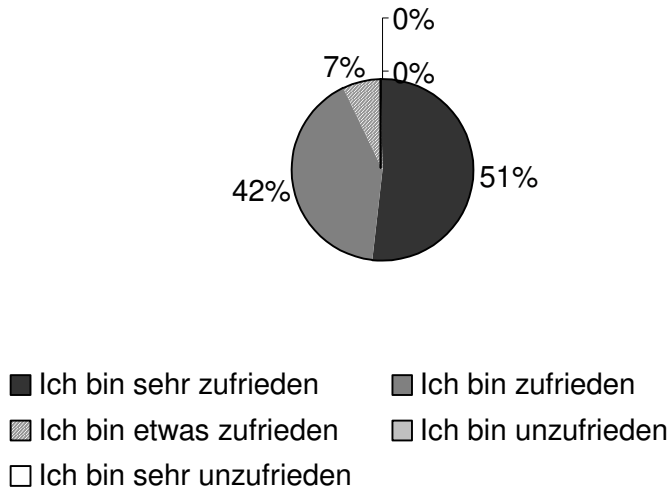

Abbildung 39: Zufriedenheit mit Mustern (Frage 6B)

Wenn überhaupt, wo werden Probleme in der Anwendung von Mustern gesehen? Frage 6D zielt auf die Beantwortung dieser Frage ab. Einzig die Schwierigkeit, passende Muster zum aktuellen Problem zu finden, sticht hervor. 50 Prozent der Befragten gaben an, dies als sehr bzw. ziemlich problematisch anzusehen. Dies unterstützt die Forderung, ein Musterorganisationsschema zu entwickeln, welches die Suche nach passenden Mustern erleichtert.

Als relativ unproblematisch sehen 56 Prozent der Befragten die Anzahl verfügbarer Muster an. Dies könnte als Indiz für eine Beantwortung der in Abschnitt 4.3 offen gelassenen Frage nach der Einordnung von Mustern als Schlüssel-, Schrittmacher oder Basistechnologie benutzt werden. Eine ausreichende Anzahl an Mustern unterstützt die Einordnung von Mustern als Basistechnologie. Die vier verbleibenden Problemkategorien lassen sich nur schwer analysieren. Zu sehen ist jedoch, dass jeweils mindestens 66 Prozent die dargestellten Problembereiche als mehr oder weniger diffizil ansehen.

Weitere von den Befragten genannte Probleme sind:

- Einige der veröffentlichten Muster sind nicht wirklich Muster;
- Musterbeschreibungen verschweigen negative Konsequenzen;
- Die Wartung von Musterkatalogen; fehlende Hierarchie von Mustern in der Art ALEXANDERS; Dokumentation von Mustersprachen, anstatt lose gekoppelter Muster;
- Die Anwendung von Mustern, obwohl diese gar nicht benötigt werden;
- Fehlendes Verständnis für die Beziehungen zwischen Mustern innerhalb einer Mustersprache; zu viele verschiedene Muster, die nicht miteinander integriert sind;
- Die Existenz von Musterduplikaten; die weit verstreute Veröffentlichung von Mustern in vielen verschiedenen Publikationen; eine Explosion verfügbarer Muster macht eine mögliche Integration unmöglich;
- Die Nutzung anwendungsbereichsspezifischer Terminologie in der Beschreibung von Mustern verhindert deren Verwendung auf anderen Gebieten.

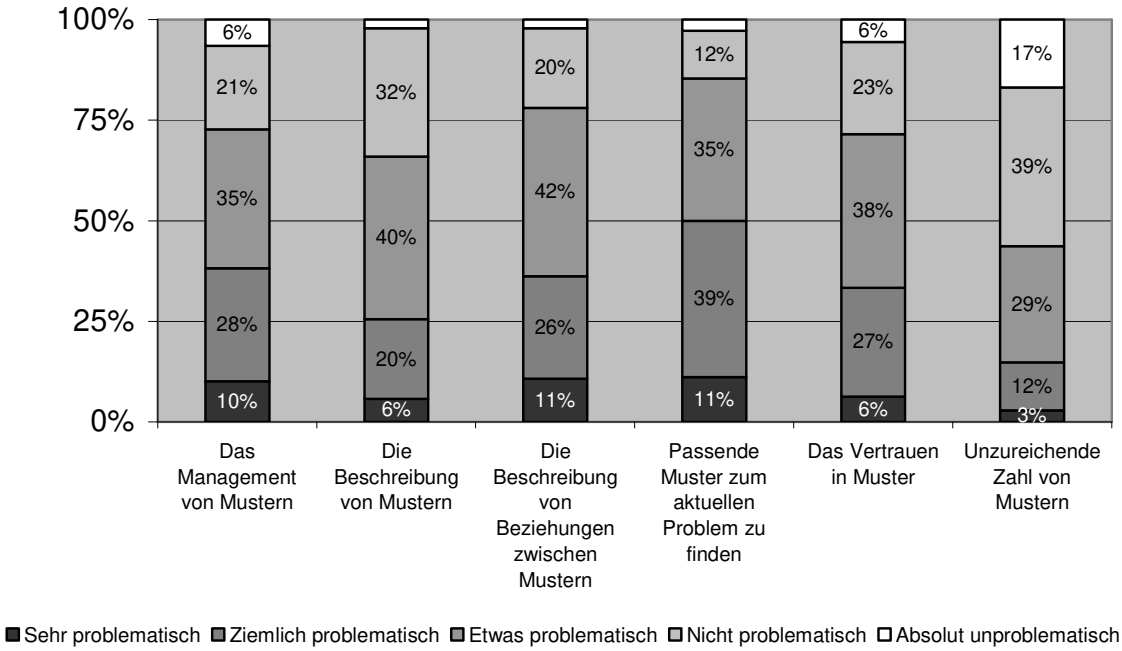

Abbildung 40: Problembereiche bei der Nutzung von Mustern (Frage 6D)

Auf die Frage, ob Muster für die Bereiche Wiederverwendung, Dokumentation und Einführung entwickelt werden könnten, antworteten ca. 90 Prozent der Befragten mit „Ja“. Dies erschien jedoch nur 80 Prozent als sinnvoll. Trotzdem lässt dies darauf schließen, dass auch für diese Aktivitäten der Systementwicklung in Zukunft Muster entwickelt werden. Die Anzahl der Nichtantworten für diese Frage lag zwischen acht und 14.

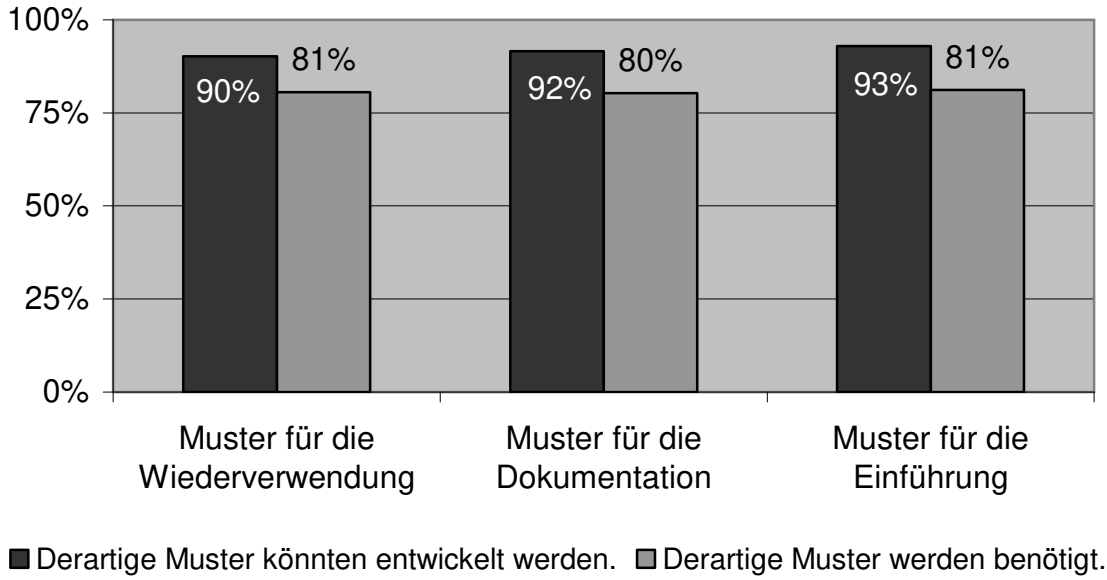

Abbildung 41: Entwicklung weiterer Musterarten (Frage 6C)

6.3.7 Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil

Aus der Analyse von Frage 7A lässt sich ableiten, dass die Mehrheit der verwendeten Muster außerhalb der Organisation der Befragten entwickelt wurden (vgl. Abbildung 42). Rund ein Drittel verwendet ausschließlich Muster, die außerhalb der Organisation entwickelt wurden. Nur zehn Prozent schätzen ein, dass die Mehrheit der verwendeten Muster eigene Muster sind. Dies lässt darauf schließen, dass ein Grossteil der Befragten keine Probleme damit haben, fremde Muster zu verwenden.

Dies ließe darauf schließen, dass sich ein Markt für Muster entwickeln könnte. Die Fragen 7B bis 7F untersuchen dies näher. Aus Abbildung 43 ist zu ersehen, dass 59 Prozent über Muster ihrer eigenen Organisation mit Personen außerhalb ihrer Organisation diskutieren. Dieser hohe Anteil spricht gegen die Entwicklung eines Marktes, insofern dies die Vermutung unterstützt, dass ein Grossteil der befragten Personen die für ihre Organisation spezifischen

Muster nicht als Wettbewerbsvorteil sehen. In der Tat sehen 52 Prozent der Befragten die Menge organisationsspezifischer Muster nicht als Wettbewerbsvorteil an.

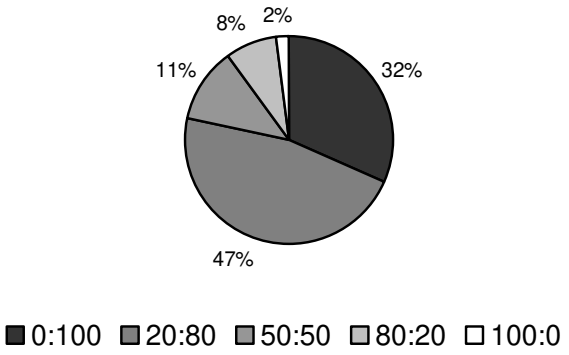

Abbildung 42: Die Verwendung fremder vs. selbstentwickelter Muster (fremd:eigen, Frage 7A)

Ebenfalls für die geringe Bedeutung von organisationsspezifischen Mustern als Wettbewerbsvorteil spricht der hohe Grad des Austausches von Mustern zwischen verschiedenen Organisationen.

Nur neun Prozent der Befragten gaben an, dass innerhalb ihrer Organisation Untersuchungen zu potentiellen Einsparmöglichkeiten durch die Nutzung von Mustern durchgeführt wurden. Dies ermöglicht den Schluss, dass der Wert des Einsatzes von Mustern innerhalb der Organisationen entweder gar nicht oder nur schwer zu quantifizieren ist. Eine rationale Begründung des Einsatzes von Mustern, basierend auf einem Kosten-/Leistungs-Vergleich ist somit kaum möglich.

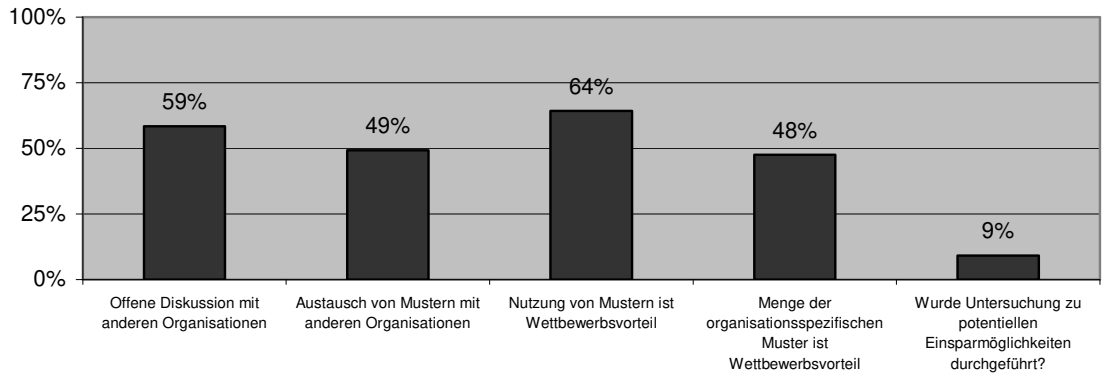

Abbildung 43: Muster als Wettbewerbsvorteil (Fragen 7B bis 7F)[1]

[1] Für die Fragen 7D, 7E und 7F ergaben sich jeweils 8, 10 bzw. 9 Nichtantworten.

Auf die direkte Frage, wie ein möglicher Markt für Muster aussehen könnte, antworteten 49 Prozent der Befragten, dass Werkzeuge, die die Anwendung von Mustern automatisieren, in Zukunft verfügbar sein werden. Einen Markt für einzelne Muster sehen nur 14 Prozent, wohingegen 36 Prozent Gruppen verwandter Muster als verkaufbar ansehen, beispielsweise in Form von Zeitschriften. Elf der Befragten sehen in der Publikation von Büchern über Muster die Bestätigung für die Möglichkeit der Vermarktung von Mustern. Weitere Angaben unter Sonstiges waren:

- Die Möglichkeit, Muster in Verbindung mit Softwarekomponenten als Mittel zur Dokumentation zu verwenden;
- Die Möglichkeit, die Verwendung von Mustern als Beratungs- bzw. Servicedienstleistung anzubieten;
- Die mögliche Patentierbarkeit von Mustern und die damit verbundene Verbesserung der Vermarktungsmöglichkeiten.

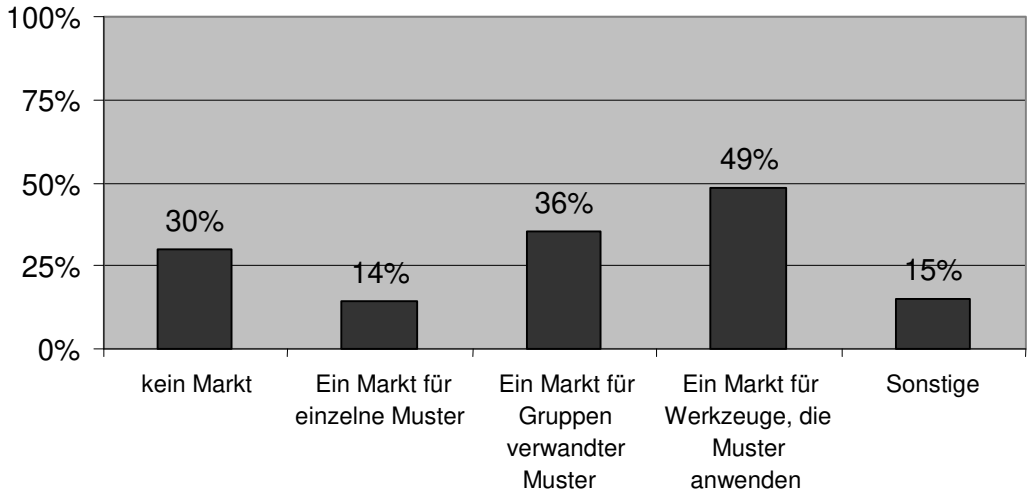

Abbildung 44: Wie könnte ein Markt für Muster aussehen (Frage 7G, Mehrfachantworten möglich)?[1]

[1] Diese Frage wurde von 34 Befragten nicht beantwortet.

7. Zusammenfassung und Ausblick

Das Ziel dieser Arbeit, eine Überblick über das Verständnis und die Nutzung von Mustern in der Praxis zu erlangen, wurde durch eine umfassende deskriptive Analyse des gesammelten Datenmaterials erreicht. Dabei erwiesen sich die im Kapitel eins bis vier erarbeiteten theoretischen Konzepte als gute Basis für eine Strukturierung der Fragen zu den einzelnen Analysegebieten.

Auf der Grundlage der definierenden Eigenschaften E1 bis E15 wurden Muster in Kapitel zwei von anderen Ansätzen der Wiederverwendung abgegrenzt. Parallel dazu wurden diese Eigenschaften benutzt, um das Verständnis von Mustern in der Praxis abzuleiten. Die daraus abgeleitete Abgrenzung von Mustern stellt einen allgemeinen Konsens über das Verständnis von Mustern dar.

Die entwickelte Klassifikation von Mustern in elf Musterarten wurde innerhalb des Fragebogens beibehalten. Die verwendete Kategorisierung von Mustern in phasenabhängige Muster im Gegensatz zu Prozessmustern ist dabei als kritisch anzusehen, insofern es in der Praxis mitunter problematisch ist, eine klare Grenze zwischen beiden zu ziehen. Weitere (empirische) Forschung könnte dies klären. Eine Analyse des Datenmaterials ergab, dass Analysemuster, Architekturmuster, Entwurfsmuster, Idiome und Testmuster dominieren. Diese Musterarten werden am häufigsten angewendet und weisen einen hohen Grad der Vertrautheit unter den Befragten auf.

Die in Kapitel drei entwickelten Ansätze für die Automatisierung der Anwendung von Mustern werden, mit Ausnahme von Entwurfsmuster und Idiomen, in der Praxis nur selten benutzt. Nichtsdestotrotz konnten die vorgestellten Automatisierungsansätze in den Mikroprozess für Muster, der ebenfalls in Kapitel drei entwickelt wurde, eingeordnet werden. Auch für die Diskussion verschiedener Formen der Charakterisierung von Mustern erwies sich der vorgestellte Mikroprozess als anwendbar. Die für eine Charakterisierung verwendeten Musterbeschreibungsformen und Musterorganisationsschemata werden in der Praxis häufig benutzt. Ein Drittel der Befragten ist unzufrieden mit den derzeitig genutzten Musterorganisationsschemata. Weiterer Forschungsbedarf besteht. Welcher Ansatz dabei verfolgt werden sollte und welche Musterbeziehungen ein neu zu entwickelndes Schema abbilden sollte, wurde anhand des Fragebogens analysiert. Ähnlich den Werken ALEXANDERS

scheint die Darstellung von Hierarchien von Mustern, wie sie schon in Kapitel 3 angeregt wurde, in der Praxis gewünscht.

Die mit Entwicklung des Mikroprozesses verfolgte Idee einer Einordnung von Mustern in verwendete Vorgehensmodelle wird von den Befragten mitunter abgelehnt. Häufig werden Muster als ein nicht formalisierbarer Ansatz angesehen. Gegenwärtig beschäftigt sich eine Studie an der University of North Carolina, Asheville, mit dem Thema der Dissemination der Nutzung von Mustern innerhalb einer Organisation. Die Ergebnisse stehen noch aus.

Einzelne, in der Literatur zu findende Erfahrungsberichte bildeten die Grundlage für eine Darstellung der Vor- und Nachteile, die durch die Nutzung von Mustern entstehen. Auf Basis dieser Beobachtungen Einzelner konnte ein umfassenderes Bild der Nutzung von Mustern gezeichnet werden. Als Hauptvorteil von Mustern in der Praxis wird die Verbesserung der Kommunikation zwischen allen am Systementwicklungsprozess beteiligten Personen gesehen. Die Frage nach den Vor- und Nachteilen der Anwendung von Mustern bot die Gelegenheit, eine offene Antwort zu geben. Von dieser Möglichkeit wurde von den Befragten rege Gebrauch gemacht. Eine Übersicht über die gegebenen Antworten ist in Anhang 18 zu finden.

Nur kurz wurde auf den Wert von Mustern als Wettbewerbsfaktor sowie die Entwicklung eines Marktes eingegangen. Die Entwicklung eines Marktes für einzelne Muster wird von den meisten Befragten als unwahrscheinlich angesehen. Mögliche Vermarktungsstrategien sollten sich auf die Publikation von Gruppen von Mustern als auch auf die Entwicklung von Werkzeugen für die Anwendung von Mustern konzentrieren. Ein diesen Bereich tangierendes Problem sind die Auswirkungen einer möglichen Patentierbarkeit von Mustern. Weitere Forschung auf diesem Gebiet erscheint notwendig.

Die Entscheidung, die Studie in Form eines Internetfragebogens durchzuführen, erwies sich als gut, insofern eine Rücklaufquote von 18 Prozent mit relativ geringen Kosten erreicht werden konnte. Da davon auszugehen ist, dass praktisch alle Personen der Grundgesamtheit mittels Email zu erreichen sind, bot dieser Weg eine gute Möglichkeit, die Individuen der Stichprobe zu kontaktieren.

Abbildungsverzeichnis

Abbildung 1: Der Übergang vom Problemsystem zum Lösungssystem (vgl. [Essw93], S. 177)..... 5

Abbildung 2: Überlagerung von Transformationszyklen und Systemlebenszyklus 6

Abbildung 3: Die Aktivitäten der Entwicklung von Softwaresystemen..... 10

Abbildung 4: Die Einordnung des Wasserfallmodells in den allgemeinen Prozess der Systementwicklung..... 11

Abbildung 5: Die Einordnung inkrementeller Entwicklung in den allgemeinen Prozess der Systementwicklung..... 14

Abbildung 6: Die Einordnung der Entwicklung mittels evolutionären Prototyps in den allgemeinen Prozess der Systementwicklung 15

Abbildung 7: Die Ansätze zur Überwindung der Schwächen des Wasserfallmodells..... 17

Abbildung 8: Eine Abgrenzung von Mustern im Verständnis dieser Arbeit gegen andere Ansätze der Systementwicklung anhand der Eigenschaften E1 bis E15 27

Abbildung 9: Eine Kategorisierung von Musterarten 29

Abbildung 10: Die Einordnung der verschiedenen Musterarten in den allgemeinen Prozess der Systementwicklung..... 34

Abbildung 11: Der Mikroprozess für Muster 40

Abbildung 12: Einordnung der vier Ansätze zur Automatisierung von Mustern in den Mikroprozess (vgl. Abbildung 11)..... 46

Abbildung 13: Beispielhafte Darstellung der Auswirkungen der Anwendung von Idiomen auf zeitlich nachgelagerte Aktivitäten im Rahmen einer inkrementellen Entwicklung (vgl. Abb. 6) 52

Abbildung 14: Die fünf Ansätze für das Zusammenwirken von Mustern zwischen verschiedenen Aktivitäten des Makroprozesses (vgl. Abbildung 11)..... 58

Abbildung 15: Wie erfolgte der erste Kontakt mit den Befragten (Frage 8D)?..... 76

Abbildung 16: Art der Organisation, in der die Befragten arbeiten (Frage 8C)..... 77

Abbildung 17: Welche Eigenschaften charakterisieren den Begriff Muster (Frage 1A)?..... 80

Abbildung 18: Anteil der Softwareentwickler, die Muster nutzen/entwickeln (Fragen 2A, 2B)..... 82

Abbildung 19: Einführung neuer Muster in die Organisation (Frage 2C, Mehrfachantworten möglich) 83

Abbildung 20: Arten der Suche nach neuen Musterideen (Frage 2D, Mehrfachantworten möglich)... 84

Abbildung 21: Organisatorische Unterstützung von Musternutzung und Musterproduktion (Fragen 2E, 2F) 85

Abbildung 22: Häufigkeit der individuellen Nutzung von Mustern (Frage 3A, alle Befragten) *87*

Abbildung 23: Häufigkeit der Nutzung von Mustern in der Organisation (Frage 3A, alle Befragten). *88*

Abbildung 24: Häufigkeit der individuellen Nutzung von Mustern (Frage 3A, 8A, Auswahl der Befragten) *89*

Abbildung 25: Vertrautheit mit Mustern (Frage 3B, alle Befragten) *90*

Abbildung 26: Vertrautheit mit Mustern (Frage 3B, 8A, Auswahl der Befragten) *90*

Abbildung 27: Anzahl der Nutzer von Werkzeugen für Mustern (Frage 3C) *91*

Abbildung 28:Anzahl der Nennungen der verschiedenen Arten der Automatisierung (Frage 3C, vergleiche Abbildung 27, Mehrfachnennungen möglich,) *92*

Abbildung 29: Wie werden Muster angewendet (Frage 3D)? *93*

Abbildung 30: Wird ein Vorgehensmodell innerhalb der Organisation benutzt (Frage 4A, 2A) *94*

Abbildung 31: Anteil der Nutzung der Konzepte K1 bis K5 (Frage 4B, 4A, 2A, Mehrfachantworten möglich) *94*

Abbildung 32: Anteil der Befragten, für die Musternutzung und Musterproduktion Bestandteil der Vorgehensmodells sind (Frage 4C, 4D, 4A, 2A) *95*

Abbildung 33: Anteil der Nutzung verschiedener Musterbeschreibungsformen (Frage 5A) *96*

Abbildung 34: Anteil der Nutzung von Musterorganisationsschemata (Frage 5B) *97*

Abbildung 35: Zufriedenheit mit den genutzten Musterorganisationsschemata (Frage 5C) *97*

Abbildung 36: Welcher Ansatz für Musterorganisationsschemata wird bevorzugt (Frage 5D)? *98*

Abbildung 37: Wichtigkeit einzelner Beziehungen von Mustern für die Entwicklung eines Musterorganisationsschemas (Frage 5E) *99*

Abbildung 38: Vorteile und Nachteile bei der Nutzung von Mustern (Frage 6A) *100*

Abbildung 39: Zufriedenheit mit Mustern (Frage 6B) *101*

Abbildung 40: Problembereiche bei der Nutzung von Mustern (Frage 6D) *102*

Abbildung 41: Entwicklung weiterer Musterarten (Frage 6C) *103*

Abbildung 42: Die Verwendung fremder vs. selbstentwickelter Muster (fremd:eigen, Frage 7A) *104*

Abbildung 43: Muster als Wettbewerbsvorteil (Fragen 7B bis 7F) *104*

Abbildung 44: Wie könnte ein Markt für Muster aussehen (Frage 7G, Mehrfachantworten möglich)? *105*

Abkürzungsverzeichnis

CORBA	Common Object Request Broker Architecture
CVOPS	C-based Virtual Operating System
DTD	Document Type Definition
EuroPLOP	European Pattern Languages of Programs (Conference)
GoM	Grundsätze ordnungsgemäßer Modellierung
IPC	Interprocess Communication
J2EE	Java 2 Enterprise Edition
LOTOS	Language of Temporal Ordering Specifications
OMT	Object Modelling Technique
OOA	Object-Oriented Analysis
PaL	Pattern-oriented Language
PLOP	Pattern Languages of Programs (Conference)
RDBMS	Relational Database Management System
SGML	Standard Generalized Markup Language
UML	Unified Modelling Language
WWW	World Wide Web
XML	Extensible Markup Language
XSD	XML Schema Definition

Literaturverzeichnis

[Aar+95] Aarsten, A.; Elia, G.; Menga, G.: G++: A Pattern Language for Computer-Integrated Manufacturing. In [CoSc95], S. 91-118

[Aho+88] Aho, A.; Sethi, R.; Ullman, J.:Compilerbau. Addison Wesley, Bonn u.a., 1988

[Anti01] o.V.: The Software Patterns Criteria. Internet: http://www.antipatterns.com/whatisapattern/, Download: 21.03.2001, 2001

[Anth96] Anthony, D.: Patterns for Classroom Education. In: [CoSc96], S. 391-406

[Ale+77] Alexander, C.; Ishikawa, S.; Silverstein, M.; Jacobson, M.; Fiksdahl-King, I.; Angel, S.: A Pattern Language: Towns, Buildings, Construction. Oxford University Press, New York, 1977

[Alex79] Alexander, C.: The Timeless Way of Building. Oxford University Press, New York, 1979

[Ambl98] Ambler, S.: An Introduction to Process Patterns. Internet: http://www.ambysoft.com/processPatterns.pdf, Download: 05.07.2001, 1998

[Ambl01] Ambler, S.: Completing the Unified Process With Process Patterns. Internet: http://www.ambysoft.com/unifiedProcess.html, Download: 02.05.2001, 2001

[Bac+95] Bach, V.; Brecht, L.; Österle, H.: Software-Tools für das Business Process Redesign. FBO-Velag, Baden-Baden, 1995

[Ball00] Ballantyne, C.: Why Survey Online? A Practical Look at Issues in the Use of the Internet for surveys in Higher Education. Internet: http://cleo.murdoch.ed u.au/evaluation/pubs/confs/aea-2000.html, Download: 05.07.2001, 2000

[Balz98] Balzert, H.: Lehrbuch der Software-Technik: Software-Management, Software-Qualitaetssicherung, Unternehmensmodellierung. Spektrum Akademischer Verlag, Heidelberg u.a., 1998

[Bec+01] Beck, K.; Coplien, J.; Crocker, R.; Dominick, L.; Meszaros, G.; Paulisch, F.; Vlissides, J.: Industrial Experience with Design Pattern. Internet: http://www1.bell-labs.com/user/cope/Patterns/ICSE96/icse.html, Download: 05.07.2001, 2001

[Beck01] Beck, K.: Simple Smalltalk Testing: With Patterns. Internet: http://www.xProgramming.com/testfram.htm, Download: 29.09.2001, 2001

[Bell01] o.V.: Organizational Patterns. Internet: http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?OrganizationalPatterns, Download: 21.03.2001, 2001

[Bind00] Binder, R.: Testing Object-Oriented Systems: Models, Patterns, and Tools. Addison Wesley Longman, Reading u.a., 2000

[BoBr01] Bolognesi, T.; Brinksma, E.: Introduction to the ISO Specification Language LOTOS. Internet: http://lotos.site.uottawa.ca/ftp/pub/Lotos/Intro/bolognesi_brinksma.pdf, Download: 21.03.2001, 2001

[Boeh76] Boehm, B.: Software Engineerung. In: IEEE Transactions on Computing, Vol. 25, No. 12, 1976. S. 1226-1241.

[Booc94] Booch, G.: Object-oriented analysis and design with applications. The Benjamin/Cummings Publishing Company, Redwood City, 1994

[Bosc98] Bosch, J: Design Pattern as Language Constructs. In Journal of Object-Oriented Programming, Mai 1998, S. 18-32

[Bosc01] Bosch, J.: Design Patterns & Frameworks: On the issue of Language Support. Internet: http://www.ipd.hk-r.se/bosch/lsdforg/bosch.ps, Download: 05.07.2001, 2001

[BrFl98] Bradac, M.; Fletcher, B.: A Pattern Language for Developing Form Style Windows. In [Mar+98], S. 347-358

[Broc97] o.V.: Der Brockhaus: In Fünfzehn Bänden. Brockhaus Verlag, Leipzig, 1997

[Broo95] Brooks, F.: The mythical man-month: essays on software engineering. Addison Wesley Longman, Wokingham u.a., 1995

[Bro+00] Brown, W.; McGormick, H.; Scott, T.: AntiPatterns in Project Management. John Wiley & Sons, New York, 2000

[Bro+98] Brown, W.; Malveau, R.; McGormick H.; Mowbry, T.: AntiPatterns; refactoring Software, Architectures, and Projects in Crisis. John Wiley & Sons, New York, 1998

[Brow01] Brown, K.: Design Reverse-Engineering and Automated Design Pattern Detection in Smalltalk. Internet: http://members.aol.com/kgb1001001/Articles/THESIS/thesis.htm, Download: 05.07.2001, 2001

[BrWh96] Brown, K.; Whitenack, B.: Crossing Chasms: A Pattern Language for Object-RDBMS. In [CoSc96], S. 227-238

[Bud+96] Budinsky, F.; Finnie, M.; Vlissides, J.; Yu. P.: Automatic Code Generation from Design Patterns. Internet: http://www.research.ibm.com/journal/sj/budin/budinsky.html, Download: 05.07.2001, 1996

[Bünn99] Bünnig, S: Entwicklung einer Sprache zur Unterstützung von Design Patterns und Implementierung eines dazugehörigen Compilers, Diplomarbeit. Internet: http://wwwswt.informatik.uni-rostock.de/deutsch/Infothek/uml/pal/DiplomBuennig.pdf, Download: 05.07.2001, 1999

[BuMe95] Buschmann, F.; Meunier, R.: A System of Patterns. In [CoSc95], S. 325-343

[Bus+98] Buschmann, F.; Meunier, R.;Rohnert, H.; Sommerlad, P.; Stal, M.: Pattern-orientierte Software-Architektur: Ein Pattern-System. Addison-Wesley-Longman, Bonn u.a., 1998

[ChCr90] Chikofsky, E.; Cross, J.: Reverse Engineering and Design Recovery: A Taxonomy. In IEEE Software Volume 7 Issue 1, 1990, S. 13-17

[Chu+99] Chu, W.; Lu, C.; Shiu, J. Xudong, H.: Pattern Based Software Re-engineering: A Case Study. In IEEE Proceedings of Software Engineering Conference Sixth Asia Pacific (APSEC 99), 1999, S.300-308

[CiNi99] Cinnéide,M.; Nixon, P.: A Methodology for the Automated Introduction of Design Patterns. In IEEE Proceedings of International Conference on Software Maintenance (ICSM 99), 1999, S. 463-472

[Clin96] Cline, M.: The Pros and Cons of Adopting and Applying Design Patterns in the Real World. In Communications of the ACM, Oktober 1996 Volume 39, No. 10, S. 47-49

[Coa+95] Coad, P.; North, D.; Mayfield, M.: Object Models: Strategies, patterns, applications. Prentice-Hall, Englewood Cliffs u.a., 1995

[CoSc95] Coplien, J. (Hrsg.); Schmidt, D. (Hrsg.): Pattern Languages of Program Design. Addison-Wesley Publishing, Wokingham u.a., 1995

[CoSc96] Coplien, J. (Hrsg.); Schmidt, D. (Hrsg.): Pattern Languages of Program Design 2. Addison-Wesley Publishing, Wokingham u.a., 1996

[Copl96a] Coplien, J.: Broadening beyond objects to patterns and other paradigms. In ACM Computing Surveys 28 (4es), Dezember 1996, Internet: http://www.acm.org/pubs/citations/journals/surveys/1996-28-4es/a152-coplien/a152-coplien.html, Download: 21.03.2001, 1996

[Copl96b] Coplien, J.: A brief history of design patterns. Internet: http://www.bell-labs.com/user/cope/Patterns/ICSE96/node3.html, Download: 11.09.2001, 1996

[Copl96c] Coplien, J.: The Human Side of Patterns. Internet: http://www1.bell-labs.com/user/cope/Patterns/C++Report/OrgPatterns-1.html, Download: 21.03.2001, 1996

[Copl01] Coplien, J.: C++ Idioms. Internet: http://www.bell-labs.com/user/cope /Patterns/C++Idioms/EuroPLoP98.html, Download: 02.07.2001, 2001

[CoYo94] Coad, P.; Yourdon, E.: Objekt-Orientierte Analyse. Prentice Hall Verlag, München u.a., 1994

[CoYo91] Coad, P.; Yourdon, E.: Object-oriented Design. Prentice Hall, Englewood Cliffs u.a., 1991

[Cree01] Creel, C.: Requirements by Patterns. Internet: http://www.sdmagazine.com/articles/1999/9912/9912d/9912d.htm, Download: 21.03.2001, 2001

[Cunn96] Cunnningham, W.: EPISODES: A Pattern Language of Competitive Development. In [CoSc96], S. 371-390

[Cunn99] Cunningham, W. (Hrsg.): Portland Pattern Repository: Pattern Forms. Internet: http://www.c2.com/cgi/wiki?CategoryPatternForm, Download: 01.06.2001, 1999

[CzBl95] Czaja, R.; Blair, J.: Designing Surveys A Guide To Decisions and Procedures. Pine Forge Press, Thousand Oaks u.a., 1995

[Czic98] Czichy, T.: Der Einsatz von Referenzmodellen bei der Einführung von Standardsoftware im Rahmen einer Beratungstätigkeit, Seminararbeit. Internet: http://www.czichy.org/seminar.pdf, Download: 21.03.2001, 1998

[Davi88] Davis, A.; Bersoff, E.; Comer, E.: A Strategy for Comparing Alternative Software Development Live Cycle Models. In: IEEE Transactions on Software Engineering, Vol. 14, No. 10, Oktober 1988, S. 1453-1461

[Dew+99] Dewar, R.; Lloyd, A.; Pooley, R.; Stevens, P.: Identifying and Communication Expertise in Systems Engineering: A Patterns Approach. In: IEEE Software, Volume 146 Issue 3, 1999, S. 145 - 152

[DiBo01] Dillman, D.; Bowker, D.: The Web Questionnaire Challenge to Survey Methodologists. Internet: http://survey.sesrc.wsu.edu/dillman/zuma_paper_dillman_bowker.pdf, Download: 05.07.2001, 2001

[Dil+98] Dillman, D.; Tortora, R.; Bowker, D.: Principles for Constructing Web Surveys. Internet: http://survey.sesrc.wsu.edu/dillman/papers/websurveyppr.pdf, Download: 05.07.2001, 1998

[Dill98] Dillman, D.: Mail and Other Self-Adminstered Surveys in the 21st Century: The Beginning of a New Era. Internet: http://survey.sesrc.wsu.edu/dillman/papers/svys21st.pdf, Download: 05.07.2001, 1998

[Ede+97] Eden, A.; Gil, Joseph; Yehudai, A.: Automating the application of design patterns. In Journal of Object-Oriented Programming, Mai 1997, S. 44-46

[Essw93] Esswein, W.: Modellierung der Verteilbarkeit von Daten. In: Fortschrittsberichte VDI Reihe 10: Informatik/Kommunikationstechnik, Nr. 251, VDI-Verlag, Düsseldorf, 1993, S. 170-182

[FeSi94] Ferstl, O.; Sinz, E: Grundlagen der Wirtschaftsinformatik. 2. Auflage, Oldenbourg Verlag, München u.a., 1994

[FoOp95] Foote, B.; Opdyke, W.: Lifecycle and Refactoring Patterns That Support Evolution and Reuse. In [CoSc95], S. 239-257

[Forr98] Forrest, E.: Internet Marketing Research: Resources and Techniques. Internet: `http://gaius.cbpp.uaa.alaska.edu/afef/`, Download: 05.07.2001, 1998

[FoYo98] Foote, B.; Yoder, J.: The Selfish Class. In [Mar+98], S. 451-470

[Fowl97] Fowler, M.: Analysis Patterns: Reuseable Object Models. Addison Wesley Longman, Menlo Park u.a., 1997

[Gam+94] Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: elements of reuseable object-oriented software. Addison-Wesley Publishing Company, Wokingham u.a., 1994

[GéSz95] Géza, M. (Hrsg.); Szabo, Z. (Hrsg.): Kleines Lexikon der Informatik und Wirtschaftsinformatik. Oldenbourg, München, 1995

[Grah94] Graham, I.: Object Oriented Methods. 2. Auflage, Addison-Wesley Publishing Company, Wokingham u.a., 1994

[Habe97] Haberfellner, R.; Daenzer, W. (Hrsg); Becker, M.: Systems Engineering: Methodik und Praxis. Verlag Industrielle Organisation, Zürich, 1997

[Harr96] Harrison, N.: Organizational Patterns for Teams. In [CoSc96], S. 345-352

[Hay95] Hay, D.: Data model patterns: conventions of thought. Dorset House Publishing, New York, 1995

[Hein95] Heinrich, L.: Wirtschaftsinformatik-Lexikon. 5. Auflage. Oldenbourg, München, 1995

[Helm95] Helm, R.: Patterns in Practice. In ACM Proceedings of the Tenth Annual Conference on Object-oriented Programming Systems, Languages, and Applications 1995, S. 337-341

[Henn91] Hennings, R.: Informations- und Wissensverarbeitung: Theoretische Grundlagen Wissensbasierter Systeme. Walter de Gruyter & Co, Berlin u.a., 1991

[Hofm00] Hofmann, H.: Requirements engineering: a situated discovery process. Gabler Verlag, Wiesbaden, 2000

[Jac+94] Jacobson, E.; Ericsson, M.; Jacobson, A.: The Object Advantage Business Process Reengineering with Object Technology. Addison-Wesley Publishing Company, Wokingham u.a., 1994

[Jac+98] Jacobsen, E.; Kristensen, B.; Nowack, P.: Characterising Patterns in Framework Development. In IEEE Proceedings of Technology of Object-Oriented Languages and Systems (Tools 97), 1998, S. 121-142

[Jac+99] Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Development Process. Addison Wesley Longman, Reading u.a., 1999

[Jac+92] Jacobson, I.; Christerson, M.; Jonsson, P.; Övergaard, G.: Object-Oriented Software Engineerung - A Use Case Driven Approach. Addison-Wesley Publishing Company, Wokingham, 1992

[Jenn95] Jenny, B.: Projektmanagement in der Wirtschaftsinformatik. vdf Hochschulverlag AG an der ETH Zürich, Zürich, 1995

[Kel+99] Keller, R.; Schauer, R.; Robitaille, S.; Pagé, P.: Pattern-Based Reverse Engineering of Design Components. In ACM Proceedings of the International Conference on Software Engineering 1999, S. 226-235

[Kent90] Kent, W.: The Evolving Role of Database in Object Systems. Internet: `http://www.hpl.hp.com/techreports/90/HPL-90-04.pdf`, Download: 02.05.2001, 1990

[Keri99] Kerievsky, J.: Knowledge Hydrant: A Pattern Language for Study Groups. Internet: `http://www.industriallogic.com/papers/khdraft.pdf`, Download: 05.07.2001, 1999

[Keri00] Kerievsky, J.: Startle & Shout: Describe the Problem. Internet: `http://www.industriallogic.com/pulse/20000130.html`, Download: 21.03.2001, 2000

[KiBe96] Kim, J.; Benner, K.: Implementation Patterns for the Observer Pattern. In: [CoSc96], S. 75-86

[KrPr96] Krämer, C.; Prechelt, L.: Design Recovery by Automated Search for Structural Design Patterns in Object-Oriented Software. In IEEE Proceedings of the Third Working Conference on Reverse Engineering, 1996, S. 208-215

[Kruc00] Kruchten, P.:The Rational Unified Process An Introduction. 2. Auflage, Addison Wesley Publishing Company, Boston u.a., 2000

[Kruc01] Kruchten, P.: A Rational Development Process. Internet: `http://www.rational.com/products/whitepapers/334.jsp`, Download: 02.05.2001, 2001

[MaJo98] Mahemoff, M.; Johnston, L.: Pattern Languages for Usability: An Investigation on Alternative Approaches. In IEEE Proceedings 3rd Asia Pacific on Computer Human Interaction, 1998, S.25-30

[Mann00] Mannhaupt, D.: Integration of Design Patterns into Object-Oriented Design using Rational Rose, Diplomarbeit. Internet: http://wwwswt.informatik.uni-rostock.de/deutsch/Infothek/uml/rational/patterns/DiplomMannhaupt.pdf, Download: 05.07.2001, 2000

[Mar+98] Martin, R. (Hrsg.); Riehle, D. (Hrsg.); Buschmann, F. (Hrsg.): Pattern Languages of Program Design 3. Addison Wesley Longman, Wokingham u.a., 1998

[McGr99] McGregor, J.: Test Patterns: Please Stand By. In: Journal of Object-Oriented Programming, Juni 1999, S. 14-19

[McGr01] McGregor, J: An Overview of Testing. Internet: http://www.cs.clemson.edu/~johnmc/joop/col1/column1.html, Download: 21.03.2001, 2001

[Med+99] Medlin, C.; Roy, Subroto; Chai, T.: World Wide Web versus Mail Surveys: A Comparision and Report. Internet: http://www.anzmac99.unsw.edu.au/anzmacfiles/M/Medlin.pdf, Download: 05.07.2001, 1999

[MeDo98] Meszaros, G.; Doble, J.: A Pattern Language for Pattern Writing. Internet: http://www.hillside.net/patterns/Writing/pattern_index.html, Download: 05.07.2001, 1998

[Meij96] Meijers, M.: Tool Support for Object-Oriented Design Patterns, Master Thesis. Internet: http://www.serc.nl/people/florijn/papers/mm-thesis.ps.gz, Download: 05.07.2001, 1996

[Meye94] Meyer, B.: Reuseable Software The Base Object-Oriented Component Libraries. Prentice Hall, Englewood Cliffs u.a., 1994

[MoMa97] Mowbray, T.; Malveau, R.: Corba Design Patterns. John Wiley & Sons, New York u.a., 1997

[Müll97] Müller, B.: Reengineering: eine Einführung. Teubner, Stuttgart, 1997

[Nobl98a] Noble, J: Towards a Pattern Language for Object Oriented Design. In IEEE Proceedings of Technology of Object-Oriented Languages (TOOLS 28), 1998, S. 2-13

[Nobl98b] Noble, J:Classifying Relationships Between Object-Oriented Design Pattern. In: IEEE Proceedings of Australian Software Engineering Conference, 1998, S. 98-107

[Nona91] Nonaka, I.: The Knowledge-Creating Company. In Harvard Business Review, November, Dezember 1991, S. 96-104

[Ocal98] O'Callaghan, A.: ADAPTOR: a pattern language for the re-engineering of systems of Object Technology. In IEEE Colloquium on Understanding Patterns and Their Application to Systems Engineering, Digest No. 1998/308, 1998, S. 1/1-1/6

[Oest97] Oesterreich, B.: Objektorientierte Softwareentwicklung: mit der Unified modelling language. R. Oldenbourg Verlag, München, 1997

[Oht+99] Ohtsuki, M.; Yoshida, N.; Makinouchi, A.: A Source Code Generation Support System Using Design Pattern Documents Based on SGML. In IEEE Proceedings of Sixth Asia Pacific Software Engineering Conference (APSEC 99), 1999, S. 292-299

[OOSE01] o.V.: Glossar www.oose.de. Internet: http://www.oose.de/glossar/, Download: 21.03.2001, 2001

[Ott91] Ott, H.: Software-Systementwicklung: praxisorientierte Verfahren und Methoden. Hanser Verlag, München u.a., 1991

[Paul99] Paulson, J.: Software Design Patterns. Internet: http://sern.ucalgary.ca/courses/SENG/693/F00/assignments/paulson/report2.htm, Download: 21.03.2001, 1999

[Pesc97] Pescio, C.: Principles Versus Patterns. In IEEE Computer Volume 30 Issue 9, 1997, S. 130-131

[Port89] Porter, M.: Wettbewerbsvorteile: Spitzenleistungen erreichen und behaupten. Campus Verlag, Frankfurt/Main, 1989

[Pre+97a] Prechelt, L.; Unger, B.; Philippsen, M.: Documenting Design Patterns in Code Eases Program Maintenance. Internet: http://www.ubka.uni-karlsruhe.de/cgi-bin/psgunzip/1997/informatik/34/34.ps, Download: 05.07.2001, 1997

[Pre+97b] Prechelt, L.; Unger, B.; Schmidt, D.: Replication of the first controlled experiment on the usefulness of design patterns: Detailed description and evaluation. Internet: http://wwwipd.ira.uka.de/~prechelt/Biblio/wustl_pattern34-1997.ps.gz, Download: 05.07.2001, 1997

[Pree94] Pree, W.: Design Pattern for Object-Oriented Software Development. Addison Wesley, Wokingham u.a., 1994

[Pres87] Pressman, R.: Software Engineering: A Practitioner's Approach. McGraw-Hill Book Company, New York u.a., 1987

[Pres00] Pressman, R.: Software Engineering: A Practitioner's Approach European Adaptation. 5. Auflage, McGraw-Hill Publishing Company, London u.a., 2000

[RaGo01] Rawsthorne, D.; Goodwin, P.: Effective Analysis for Object-Oriented Development. Internet: http://www.creport.com/html/from_pages/view_recent_articles_c.cfm?ArticleID=1568, Download: 21.03.2001, 2001

[RaSr00] Ram, D.; Skreekanth, M.: Reusable Integrated Components of Inter-related Patterns for Software Development. In IEEE Proceedings of Seventh Asia-Pacific Software Engineering Conference (APSEC 2000), 2000, S. 364-371

[Reic00] Reichardt, S.: Wiederverwendungsansätze in der Software-Entwicklung: Darstellung und Einordnung am Beispiel der „IBM SanFrancisco Application Business Components for Java“. Diplomarbeit, TU Dresden, 2000

[RiZü96] Riehle, D.; Züllighoven, H.: Understanding and Using Patterns in Software Development. Internet: http://www.riehle.org/papers/1996/tapos-1996-survey.pdf, Download: 05.07.2001, 1996

[Risi01a] Rising, L.: Patterns: A Way to Reuse Expertise. Internet: http://www.agcs.com/supportv2/techpapers/patterns/papers/expertise.htm, Download: 03.04.2001, 2001

[Risi01b] Rising, L.: Patterns Mining. Internet: http://www.agcs.com/supportv2/techpapers/patterns/papers/mining.htm, Download: 03.04.2001, 2001

[Risi01c] Rising,L.: Patterns: A Training Experience. Internet: http://www.agcs.com/supportv2/techpapers/patterns/papers/cacmpatterns.htm, Download: 03.04.2001, 2001

[Royc70] Royce, W.: Managing the development of large software systems: Concepts and techniques. In: Proceedings of IEEE WESTCON, Los Angeles, 1970, S. 1-9.

[Rumb91] Rumbaugh, J.: Object-oriented modelling and design. Prentice-Hall, Englewood Cliffs u.a., 1991

[Schm95] Schmalzl, J.: Architekturmodelle zur Plannung der Infomrationsverarbeitung von Kreditinstituten. Physica-Verlag, Heidelberg, 1995

[Schm01] Schmidt, D.: Experience Using Design Patterns to Develop Reuseable Object-Oriented Communication Software. Internet: http://www.cs.wustl.edu/~schmidt/CACM-95.ps.gz, Download: 05.07.2001, 2001

[ScCl99] Schmidt, D.; Cleeland, C.: Applying a Pattern Language to Develop Extensible ORB Middleware. Internet: http://www.cs.wustl.edu/~schmidt/PDF/ORB-patterns.pdf, Download: 05.07.2001, 1999

[ScSt95] Schmidt, D.; Stephenson, P.: Experience Using Design Patterns to Evolve Communication Software Across Diverse OS Platforms. Internet: http://www.cs.wustl.edu/~schmidt/PDF/ECOOP-95.pdf, Download: 05.07.2001, 1995

[Schü98] Schütte, R.: Grundsätze ordnungsmässiger Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modelle. Gabler Verlag, Wiesbaden, 1998

[Sedg91] Sedgwick, R.: Algorithmen. 2. Auflage, Addison Wesley, Bonn u.a., 1991

[See+00] Seen, M.; Taylor, P.; Dick, M.: Applying a Crystal Ball to Design Pattern Adoption. In IEEE Proceedings of International Conference on Technology of Object-Oriented Languages (TOOLS 33) , 2000, S. 443-454

[Shaw94] Shaw, M.: Patterns for Software Architectures. Internet: http://www.cs.cmu.edu/afs/cs/project/vit/ftp/pdf/PLoP94.pdf, Download: 05.07.2001, 1994

[Snee91] Sneed, H.: Softwarewartung. Verlagsgesellschaft Rudolf Müller, Köln, 1991

[StHa97] Stahlknecht, P.; Hasenkamp, U.: Einführung in die Wirtschaftsinformatik. Springer, Berlin u.a., 1997

[Tayl92] Taylor, D.: Object-oriented onformation systems: planning and implementation. John Wiley & Sons, New York, 1992

[Toge01] Software: Together Control Center 5.0. Internet: http:://www.together.com/, Download: 01.05.2001, 2001

[Unge00] Unger, B; Tichy, W.: Do Design Patterns Improve Communication? An Experiment with Pair Design. Internet: http://hometown.aol.com/geshome/wess200 0/unger-tichy.pdf, Download: 22.03.2001, 2000

[Vlis98] Vlissides, J.: Pattern Hatching: Design Patterns Applied. Addison Wesley Longman, Wokingham u.a., 1998

[Vlis01] Vlissides, J.: Reverse Architecture. Internet: ftp://st.cs.uiuc.edu/pub/patterns/papers/revarch.ps, Download: 02.07.2001, 2001

[VlAl01] Vlissides, J.; Alexandrescu, A.: Pattern Hatching To Code or Not to Code. Internet: http://www.creport.com/html/from_pages/view_recent_articles_c.cfm?ArticleID=317, Download: 22.03.2001, 2001

[VTT01] o.V.: CVOPS – A Tool for Portable Communication Software. Internet: http://www.vtt.fi/tte/tte22/cvops/, Download: 29.09.2001, 2001

[Whit95] Whitenack, B.: RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development. In: [CoSc95], S. 259-291

[Wins96] Winsen, P.: (Re)engineering with Object Oriented Design Patterns, Master Thesis. Internet: http://www.serc.nl/people/florijn/papers/pvw-thesis.p s.gz, Download: 05.07.2001, 1996

[Yac+00] Yacoub, S.; Xue, H.; Ammar, H.: Automating the Development of Pattern-Oriented Designs for Application Specific Software Systems. In IEEE Proceedings of 3rd Symposium on Application-Specific Systems and Software Engineering Technology 2000, S. 163-170

[YaDo00] Yau, S.; Dong, N.: Integration of Component-Based Software Development Using Design Pattern. In IEEE Proceedings of The 24th Annual International Computer Software and Applications Conference (COMPSAC 2000), 2000, S. 369-374

[Zimm95] Zimmer, Walter: Relationships Between Design Patterns. In [CoSc95], S. 345-360

Anhang

Anhang 1: Anschreiben ..... *XVI*

Anhang 2: Startseite des Internetfragebogens (erste Seite) ..... *XVIII*

Anhang 3: Login-Seite (zweite Seite) ..... *XX*

Anhang 4: Hinweise zur Benutzung des Fragebogens (dritte Seite) ..... *XXI*

Anhang 5: Das Verständnis von Mustern (vierte Seite) ..... *XXII*

Anhang 6: Arten der Verbreitung von Mustern in der Organisation (fünfte Seite) ..... *XXIV*

Anhang 7: Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung (sechste Seite) ..... *XXVI*

Anhang 8: Musterproduktion und Musternutzung im Vorgehensmodell (siebte Seite) ..... *XXX*

Anhang 9: Die Charakterisierung von Mustern – Musterbeschreibungsformen und Musterorganisationsschemata (achte Seite) ..... *XXXII*

Anhang 10: Gesammelte Erfahrungen mit Mustern (neunte Seite) ..... *XXXV*

Anhang 11: Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil (zehnte Seite) ..... *XXXIX*

Anhang 12: Allgemeine Angaben (elfte Seite) ..... *XLI*

Anhang 13: Danksagung für das Ausfüllen des Fragebogens (zwölfte Seite) ..... *XLIV*

Anhang 14: Zusammenfassung der verschiedenen Musterarten ..... *XLV*

Anhang 15: Aufstellung angeschriebener Mailinglisten ..... *L*

Anhang 16: Zusammenfassung der Mustereigenschaften E1 bis E15 ..... *LV*

Anhang 17: Exakte Prozentangaben für Frage 6A (vgl. Abbildung 37) ..... *LVI*

Anhang 18: Freie Angaben zu Frage 6A (vgl. Abschnitt 6.3.6) ..... *LX*

Anhang 1: Anschreiben

Dear Colleague,

This email is about a survey on the use of Patterns in Software Development conducted by the University of Technology, Dresden, Germany. It is part of a research project for the Systems Engineering department. You are kindly requested to participate in this survey by following this link:

`http://wiseweb.wiwi.tu-dresden.de/survey/`

You have been chosen to take part, since you showed an interest in this subject in the past (e.g. by publishing papers on Software Patterns). This survey is largely dependent on your assistance. If you know other people who might be interested in this survey please feel free to forward this email. Everyone's help is deeply appreciated.

The survey aims to clarify practical issues in seven major areas of interest:

1. Reaching a common understanding of the term Pattern as it is used in software engineering.
2. Do organisations produce/write and consume/introduce Patterns? How?
3. Usage of different pattern kinds (e.g. Design Patterns, Organisational Patterns, Architectural Patterns) in general and usage of tools for pattern automation.
4. Possible inclusion of pattern usage and pattern mining in the process model.
5. Usage of different pattern organisation schemes (e.g. a pattern catalogue).
6. Summarise gained experience with pattern usage.
7. Recognition of Patterns as a potential competitive advantage.

The questionnaire will take about 20 minutes to complete. All information will be handled anonymously and will only be published in aggregated form. If you provide your email address, a summary of the results will be sent to you as soon as it is available.

To start the survey please follow this link for a brief introduction:

http://wiseweb.wiwi.tu-dresden.de/survey/

Many thanks for your time and effort,

Sincerely,

Thoralf Czichy, Student

Andreas Dietzsch, Department of Systems Engineering, University of Technology, Dresden

Prof. Dr. Werner Esswein, Department of Systems Engineering, University of Technology, Dresden

Anhang 2: Startseite des Internetfragebogens (erste Seite)

Survey on Patterns in Software Development

Introduction

Although Patterns have their roots in practice, little is really known about the actual usage of Patterns in practice. This questionnaire is part of a research project conducted on behalf of the **University of Technology, Dresden, Germany**. As part of a master thesis on practical Pattern usage it aims to draw a big picture of different aspects **of Patterns in software development**. In addition to Design Patterns, also included are patterns for other application fields, for example Analysis Patterns and Process Patterns.

Objectives and Areas of Interest

1. Reaching a common understanding of the term Pattern as it is used in software engineering.
2. Do organisations produce/write and consume/introduce Patterns? How?
3. Usage of different pattern kinds (e.g. Design Patterns, Organisational Patterns, Architectural Patterns) in general and usage of tools for pattern automation.
4. Possible inclusion of pattern usage and pattern mining in the process model.
5. Usage of different pattern organisation schemes (e.g. a pattern catalogue).
6. Summarise gained experience with pattern usage.
7. Recognition of Patterns as a potential competitive advantage.

Structure of the Questionnaire

The questionnaire consists of 36 questions and the time needed to complete it is approximately 20 minutes. The survey is divided into eight general areas, each containing two to seven questions. Step by step you will be guided through all eight areas. All provided information will be handled anonymously and only be published in aggregated form.

Getting a Summary of the Results

If you enter your email address at the end of this questionnaire, a summary of this survey will be sent to you as soon as it is available.

Your Help

This survey greatly depends on your assistance. If you know other people who might be interested, click here to send them a link to this page (http://wiseweb.wiwi.tu-dresden.de/survey). Everyone's help is deeply appreciated.

Further Questions

If you have any further questions do not hesitate to send an email to pattern-survey@wiseweb.wiwi.tu-dresden.de

Click the button to proceed to the login section.

Many thanks for your time and effort,

Sincerely,

Thoralf Czichy, Student, University of Technology, Dresden

Andreas Dietzsch. Department of Systems Engineering, University of Technology, Dresden

Prof. Dr. Werner Esswein, Department of Systems Engineering, University of Technology, Dresden

Anhang 3: Login-Seite (zweite Seite)

Log In to the Survey

If you are new to this survey, you have to create a new login / password pair.

By using this login you may suspend a partially completed questionnaire and resume at a later date.

You can use anything for login and password. Just make sure to remember it when you come back later.

If you have visited this survey before, you can click here to continue and use your previously created login.

Login:

Password:

Confirm password:

Anhang 4: Hinweise zur Benutzung des Fragebogens (dritte Seite)

How to Use This Online Questionnaire

Try to Answer all Questions

Please try to answer all questions. If any questions do not apply to you, or you do not feel you are able to answer them, just leave them blank or unchecked.

Navigation

To navigate from one section to another, use the 'Previous' and 'Next' Buttons located at the top and bottom of each page. No additional action is required to save the data you entered, this happens automatically when you change sections.

Use the 'Cancel' button to suspend the survey. This button is located at the right top and bottom of each section. This saves all data you have entered so far and suspends the survey.

When you come back at a later time using your previously created login, all data will be reloaded, so you can continue where you left off.

Javascript

This survey can be used without JavaScript enabled. However, some actions might not work as expected:

You will not be able to close windows by clicking on a hyperlink.

In some sections, JavaScript helps you by answering some questions based on your previous answers.

It is best to enable JavaScript while using the survey.

You can now click the button to proceed to the first section.

Anhang 5: Das Verständnis von Mustern (vierte Seite)

1. Understanding of Patterns

(1a)

In your opinion, which of the following qualities characterise a pattern in the software engineering domain (e.g. Design Patterns, Analysis Patterns, Process Patterns)?

Click (Yes) to agree, (No) to disagree.

(Yes) (No)

Characteristics Based on Pattern Origins

- A pattern occurs again and again
- A pattern captures the knowledge of experts
- A pattern is a solid practical procedure and no innovative solution
- A pattern was successfully used more than once (Rule of Three)
- A pattern is perfect
- Patterns are independent of object-oriented concepts

Characteristics Based on Pattern Descriptions

- A pattern is a (shining) example
- A pattern is a combination of some components
- A pattern is a sample/template/model
- A pattern describes a problem-solution pair
- A pattern describes its context and the forces which limit its solution
- A pattern description is isolated and short

Characteristics Based on Pattern Application

- A pattern is modified when applied, but keeps its general form
- A particular pattern can be applied to many actual problems
- A pattern unifies communication of experts

(1b)

Do you consider algorithms (e.g. Quicksort, Dijkstra's shortest path algorithm) as patterns?

(Yes) (No)

Anhang 6: Arten der Verbreitung von Mustern in der Organisation (fünfte Seite)

2. Patterns in Your Organisation

(2a)

Do you use patterns in your organisation?

(Yes) (No)

- Personally
- Other people in my organisation

(2b)

Do you write new patterns in your organisation (e.g. for later internal or external use)?

(Yes) (No)

(2c)

How do you learn/introduce new patterns in your organisation?

(Yes) (No)

- No introduction
- Self study (e.g. books, internet)
- Pattern study group (a group that meets on a regular basis and discusses patterns)
- External or internal courses
- Other - If you selected Other, please describe:

(2d)

How do you discover new ideas for patterns in your organisation?

Click (Yes) to agree, (No) to disagree.

- No discovering
- By interviewing experts
- Internal pattern workshops, meetings (e.g. a monthly meeting)
- External pattern workshops, meetings (e.g. a conference workshop)
- System reviews by experts (e.g. code reviews)
- As part of my daily work (e.g. while creating a design model)
- Other - If you selected Other, please describe:

(2e)

Is the use of patterns in your organisation promoted?

(Yes) (No)

If yes, by which means:

(2f)

Is the discovery of new patterns in your organisation promoted?

(Yes) (No)

If yes, by which means:

Anhang 7: Der Grad der Nutzung unterschiedlicher Musterarten und deren Automatisierung (sechste Seite)

3. Different Pattern Kinds - Usage and Tool Support

(3a)

Which of the following pattern types do you use strictly personally? Which ones do you use in your organisation (e.g. for communication amongst software engineers or in documentation)? Please indicate the frequency of use of patterns of these types.

For a more detailed description of each pattern kind you can click here.

Please select a rating from 1 to 5 with the following meanings: 1 = Never | 2 = Rarely (less than once a month) | 3 = Sometimes (1-3 times a month) | 4 = Frequently (once or twice a week) | 5 = Always (more than twice a week)

Personally

- Patterns for Requirement Analysis
- Analysis Patterns
- Architectural Patterns
- Design Patterns
- Idioms (Implementation Patterns)
- Patterns for Testing
- Patterns for Maintenance
- Process Patterns
- Organisational Patterns
- Educational Patterns

- Patterns on Pattern Writing

In your Organisation (e.g. for communication, in documentation):

- Patterns for Requirement Analysis
- Analysis Patterns
- Architectural Patterns
- Design Patterns
- Idioms (Implementation Patterns)
- Patterns for Testing
- Patterns for Maintenance
- Process Patterns
- Organisational Patterns
- Educational Patterns
- Patterns on Pattern Writing

(3b)

Please give an estimate of the number of patterns that you are familiar with in each category. In terms of this study you are familiar with a pattern, if you know the problem it describes and have a general idea of its solution. For a more detailed description of each pattern kind you can click here.

(Number of Familiar Patterns = none, 1-5, 6-10, 11-15, 16-20, 21-30, more than 30)

- Patterns for Requirement Analysis
- Analysis Patterns

- Architectural Patterns
- Design Patterns
- Idioms (Implementation Patterns)
- Patterns for Testing
- Patterns for Maintenance
- Process Patterns
- Organisational Patterns
- Educational Patterns
- Patterns on Pattern Writing

(3c)

Please indicate whether you use tools in relation with each pattern type. If you use a tool please select its type(s) from the five options. For a more detailed description of each pattern kind you can click here.

A. Computerised pattern descriptions (e.g. using a document management system as Lotus Notes or a database-like repository using HTML, SGML or XML)

B. Automated presentation of patterns in models/source code (e.g. comment tags or an ellipse with role associations, UML 1.3)

C. Automated discovery of patterns (e.g. automated pattern mining in existing code)

D. Automated introduction/implementation of patterns into models/source code (e.g. code refactoring or a 'pattern wizard' in a design tool)

E. Other (please describe)

(Yes) (No) (A) (B) (C) (D) (E) If you selected (E), please describe:

- Patterns for Requirement Analysis
- Analysis Patterns
- Architectural Patterns
- Design Patterns
- Idioms (Implementation Patterns)
- Patterns for Testing
- Patterns for Maintenance
- Process Patterns
- Organisational Patterns
- Educational Patterns
- Patterns on Pattern Writing

(3d)

How do you apply patterns?

(Yes) (No)

- I am using patterns as a part of my previous experience and education.
- I am using patterns directly from books, papers, magazines etc.
- I am using a software tool, which supports pattern usage.

Anhang 8: Musterproduktion und Musternutzung im Vorgehensmodell (siebte Seite)

4. Pattern Usage and Pattern Mining in the Process Model

(4a)

Do you use a process/method/model in your company? A process/method/model is a formal and disciplined approach to assigning task and responsibilities within a software development organisation (e.g. Rational Unified Process, Waterfall Model).

(Yes) (No)

If you choose no, you may proceed directly to the first question of the next section (section 5). Otherwise, please anwer the following questions too.

(4b)

Do you think the process/method/model, as you are using it in your company, supports the following concepts:

(Yes) (No)

- Incremental/Iterative Development
- Evolutionary Prototype (A prototype which evolves into a final product)
- Explorative Prototype (A Throwaway Prototype)
- Reuse (e.g. reuse of components, code, design models, …)
- Object-oriented Concepts (e.g. object-oriented programming languages, object-oriented design)

(4c)

Is the utilisation of patterns an explicit part of your development process/method/model (something that is formally described as part of your development process/method/model)?

(Yes) (No)

(4d)

Is the discovery of new patterns and pattern ideas (pattern mining) an explicit part of your development process/method/model?

(Yes) (No)

Anhang 9: Die Charakterisierung von Mustern – Musterbeschreibungsformen und Musterorganisationsschemata (achte Seite)

5. Usage of Pattern Organisation Schemes

(5a)

Which of the following pattern forms do you use? If you use a pattern form, click (Yes), otherwise click (No)

(Yes) (No)

- None
- GoF (GAMMA et al) pattern form (Pattern Name and Classification, Intent, Also Known As, Motivation, Applicability, Structure, Participants, Collaborations, Consequences, Implementation, Sample Code, Known Uses, Related Patterns)
- Canonical form (Name, Alias, Problem, Context, Forces, Solution, Example, Resulting Context, Rationale, Known Uses, Related Patterns)
- Informal Portland form (Opening statement, therefore)
- Alexander's informal pattern form (IF Context, Examples, Problem and Forces THEN Reasons, Design Form, Rule, Solution, Context and Other Patterns)
- Other - If you selected Other, please describe:

(5b)

Which of the following pattern organisation schemes (e.g. Pattern Systems) do you use?

(Yes) (No)

- None (please proceed to question 5d)
- According to the GoF (GAMMA et al) scheme for Design Patterns

    - Purpose - Creational, Structural, Behavioural
    - Scope - Class, Object
- According to the Buschmann scheme
    - Pattern Type - Architectural Pattern, Design Pattern, Idiom
    - Problem Area - e.g. Concurrency Issues
- Other - If you selected Other, please describe:

(5c)

Do you consider the used pattern organisation scheme as a suitable means for pattern retrieval?

(Yes) (No)

(5d)

Considering pattern retrieval, which of the following terms/approaches do you think is most appropriate as a pattern organisation scheme?

Choose one.

- None
- Pattern Catalogue
- Pattern System
- Pattern Language
- Other - If you selected Other, please describe:

(5e)

Considering the development of a new pattern organisation scheme for pattern retrieval and pattern application, how important do you rate the following pattern relationships for your work with patterns?

Please select a rating from 1 to 5 with the following meanings: 1 = Totally unimportant - 2 = Not important - 3 = Somewhat important - 4 = Fairly important - 5 = very important

- Pattern X uses Pattern Y as part of its solution
- Pattern X refines Pattern Y
- Pattern X solves the same problem as pattern Y
- Pattern X can be combined with pattern Y
- Pattern X frequently causes pattern Y (e.g. Model-View-Controller causes Observer Pattern)

Anhang 10: Gesammelte Erfahrungen mit Mustern (neunte Seite)

6. Experience with Patterns

(6a)

Based solely on your experience only, which of the following statements would you support? Please select a rating from 1 to 5 with the following meanings: 1 = I disagree - 2 = I tend to disagree - 3 = I tend to agree - 4 = I agree - 5 = I strongly agree

Software Properties and Pattern Usage

- Patterns support the development of software with specific qualities.
- Patterns help to develop complex and heterogeneous software architectures.
- Patterns improve the documentation of the software development process.
- Patterns improve extensibility, portability and reusability of software.
- Use of patterns results in a clearer design.
- Patterns guarantee the retention of the original design by restricting changes in maintenance.
- It is difficult to recognise a pattern in source code although using patterns developed the code.
- Patterns increase runtime costs, by introducing an additional layer of abstraction.

Individual Experience with Patterns

- Patterns improve communication in the development process between the individuals who are familiar with the respective patterns.
- Patterns support the integration of new software developers.
- It is difficult to decide which pattern is the best for a specific problem.

- Pattern descriptions are insufficient for a direct application.
- Pattern mining is a human-intensive task.
- Patterns are difficult to learn.
- Patterns lead developers to think they know more, than they actually do.

Patterns in the Software Process

- Object-oriented patterns help in the transition to object-oriented technology.
- Patterns increase the predictability of decisions in analysis, design and implementation and increase standardisation of models and program code.
- In contrast to other reuse approaches patterns have to be implemented again and again.
- Patterns rely on the experience of previous projects. They do not generate new solutions.

Costs and Benefit of Patterns

- Use of patterns is not linked to high costs.
- By using patterns, one can avoid costs related to the development of a new solution.
- Patterns reduce the risk inherent in software development projects.

Other

If you want to tell us something else about your experience with patterns, you can do it here:

(6b)

Did patterns meet your expectations?

Choose one

- I am very satisfied (patterns exceeded my expectations)
- I am satisfied (all of my expectations have been met)
- I am somewhat satisfied (most of my expectations have been met)
- I am dissatisfied (some of my expectations have not been met)
- I am very dissatisfied (none of my expectations have been met)

(6c)

Do you think patterns exist or could be developed for the following areas? Do you think there is a need for these patterns?

(Yes) (No)

Patterns Could be Developed?

- Patterns for Reuse
- Patterns for Documentation
- Patterns for Installation/Deployment

Need for Patterns?

- Patterns for Reuse
- Patterns for Documentation
- Patterns for Installation/Deployment

(6d)

Which areas do you consider problematic with patterns? Please select a rating from 1 to 5 with the following meanings: 1 = Totally unproblematic - 2 = Not problematic - 3 = Somewhat problematic - 4 = Fairly problematic - 5 = Very problematic

- Management of patterns within the pattern community (e.g. the "approval" of new patterns)
- Descriptions of patterns
- Descriptions of relations between patterns
- Finding patterns matching to your current problem
- Trust in quality of patterns
- Insufficient number of patterns

Other problematic areas:

Anhang 11: Die Wahrnehmung von Mustern als potenziellen Wettbewerbsvorteil (zehnte Seite)

7. The Value of Patterns for an Organisation

(7a)

Could you give an estimate for your usage of patterns developed within your organisation against patterns from somewhere else (e.g. from books, conference papers)?

Choose one.

- Own 0% : other 100%
- Own 20% : other 80%
- Own 50% : other 50%
- Own 80% : other 20%
- Own 100% : other 0%

(7b)

Do you openly discuss patterns of your organisation's domain with people external to your organisation?

(Yes) (No)

(7c)

Do you share patterns with other organisations?

(Yes) (No)

(7d)

A competitive advantage makes it possible to produce at lower costs or to differentiate yourself from your competitors by providing unique or superior value to the buyer. Do you consider the use of patterns as a competitive advantage?

(Yes) (No)

(7e)

Do you consider the pool of patterns specific to your organisation as a competitive advantage?

(Yes) (No)

(7f)

Did you conduct studies regarding potential savings by using patterns in your organisation?

(Yes) (No)

(7g)

Do you expect a pattern-related market to emerge?

(Yes) (No)

- No, not at all
- Yes, a market for single patterns
- Yes, a market for groups of related patterns
- Yes, a market for software tools, that apply patterns
- Yes, other - If you selected Other, please describe:

Anhang 12: Allgemeine Angaben (elfte Seite)

8. General Information, Comments, Receiving a Summary

(8a)

Which of the following activities are part of your work?

(Yes) (No)

- Requirements Analysis
- Analysis
- Architectural Design
- Detailed Design
- Implementation/Coding
- Testing
- Deployment/Introduction
- Operations and Maintenance
- Documentation
- Reuse Analysis
- Other - If you selected Other, please describe:

(8b)

Do you use general tools for one of the following activities?

Examples:

Rational Suite AnalystStudio for Requirements Analysis

Together ControlCenter for Detailed Design

Microsoft Visual Studio for Implementation

(Yes) (No)

- Requirements Analysis
- Analysis
- Architectural Design
- Detailed Design
- Implementation
- Testing
- Deployment/Introduction
- Operations and Maintenance
- Documentation
- Reuse Analysis
- Other - If you selected Other, please describe:

(8c)

I'm working in a ... Choose one

- Company.
- University.
- Self-employed.
- Other - If you selected Other, please describe:

(8d)

How did you learn about this questionnaire?

Choose one

- I am subscribed to a mailing list where a link to this questionnaire was published.
- I have got an email directly from the survey's conductors.
- A friend/colleague sent me a link to this questionnaire.
- Other - If you selected Other, please describe:

(8e)

Please include any additional comments. If you have any further inquiries about this survey you can also send an email to pattern-survey@wiseweb.wiwi.tu-dresden.de.

(8f)

If you are interested in receiving a summary of results when available, please enter your email address in the following textfield:

If you have completed all sections, you are now done. Click to go to the final page.

Anhang 13: Danksagung für das Ausfüllen des Fragebogens (zwölfte Seite)

Final Page

Many thanks for your time and effort.

If you have entered your email address in the previous section, a summary of this survey will be sent to you when it becomes available.

If you know other people who might be interested, click here to send them a link to this page (http://wiseweb.wiwi.tu-dresden.de/survey). Everyone's help is deeply appreciated.

If you have any further questions, feel free to send an email to pattern-survey@wiseweb.wiwi.tu-dresden.de.

Sincerely,

Thoralf Czichy, Student, University of Technology, Dresden

Andreas Dietzsch, Department of Systems Engineering, University of Technology, Dresden

Prof. Dr. Werner Esswein, Department of Systems Engineering, University of Technology, Dresden

Anhang 14: Zusammenfassung der verschiedenen Musterarten

Descriptions for the Patterns used in this Survey

Patterns for Requirement Analysis

Description: A Pattern for Requirement Analysis describes techniques and processes for capturing and validating the behavioural and non-behavioural requirements of a software system.

Example: Customer Rapport. How do you build and establish a good relationship with a customer.

References:

Creel, C. "Requirements by Patterns", Internet: http://www.sdmagazine.com/articles/1999/9912/9912d/9912d.htm, 01-03-21.

Whitenack, B. "RAPPeL: A Requirements-Analysis-Process Pattern Language for Object-Oriented Development", in Coplien, J., and Schmidt, D., eds. "Pattern Languages of Program Design", pp. 259-291, Wokingham: Addison-Wesley Publishing, 1995.

Analysis Patterns

Description: Analysis Patterns are concepts that represent a common construction in analysis or business modelling.

Example: Organisation Hierarchies. This pattern describes the modelling of (multiple) organisational hierarchies in an object-oriented class diagram.

Reference:

Fowler, M. "Analysis Patterns: Reuseable Object Models", Menlo Park: Addison Wesley Longman, 1997.

Architectural Patterns

Description: Architectural Patterns help you to specify the fundamental structure of a software system.

Example: Layers. The Layers pattern helps to structure applications that can be decomposed into groups of subtasks in which each group of subtasks is at a particular level of abstraction.

References:

Buschmann, F., et al. "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", New York: John Wiley & Sons, 1996.

Martin, R., et al., eds. "Pattern Languages of Program Design 3", Wokingham: Addison Wesley Longman, 1998.

Design Patterns

Description: A Design Pattern is an effective solution to a recurring problem in the design stage. Object-oriented Design Patterns consist of a specification of class and object interactions and their underlying intent.

Example: Observer. Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.

References:

Gamma, E., et al. "Design Patterns: elements of reuseable object-oriented software", Wokingham: Addison-Wesley Publishing Company, 1994.

Mowbray, T., Malveau, R. "Corba Design Patterns", New York: John Wiley & Sons, 1997.

Idioms (Implementation Patterns)

Description: An Idiom is a pattern defined for a specific programming language on a low level of abstraction. It describes how to implement particular aspects of components and their relationships in a programming language.

Example: Virtual Constructor. How to build an object of known abstract type, but of unknown concrete type, without violating the encapsulation of the inheritance hierarchy.

References:

Buschmann, F., et al. "Pattern-Oriented Software Architecture, Volume 1: A System of Patterns", New York: John Wiley & Sons, 1996.

Coplien, J. "C++ Idioms", Internet: http://www.bell-labs.com/user/cope/Patterns/C++Idioms/EuroPLoP98.html, 01-07-02.

Patterns for Testing

Description: Patterns for Testing are related to Design Patterns. A Pattern for Testing accompanying a Design Pattern defines a configuration of objects needed to test the interactions between classes that are integrated according to the design described by the Design Pattern.

Example: ObserverTestPattern. This pattern allows for testing the correct implementation of the Design Pattern Observer. It pays special attention to the completeness of notifications, the correctness of queries sent by the Observers and the consistency of the data in the Subject and the Observers.

Reference:

McGregor, J. "Test Patterns: Please Stand By", in: Journal of Object-Oriented Programming, pp. 14-19, June 1999.

Patterns for Maintenance

Description: Patterns for Maintenance describe techniques for typical maintenance activities as optimisation, stabilisation or dealing with depreciation.

Example: Consolidate the Program to Support Evolution and Reuse. Insight gained during the prototype and consolidation phases can be employed to refactor the software system.

References:

Dewar, R., et al. "Identifying and Communication Expertise in Systems Engineering: A Patterns Approach", in: IEEE Software, pp. 145 - 152, Volume 146 Issue 3, 1999.

Foote, B., and Opdyke, W. "Lifecycle and Refactoring Patterns That Support Evolution and Reuse", in Coplien, J., and Schmidt, D., eds. "Pattern Languages of Program Design", pp. 239-257, Wokingham: Addison-Wesley Publishing, 1995.

Process Patterns

Description: A process pattern is a collection of general techniques, actions, and/or tasks (activities) for developing software systems within a software development organisation.

Example: The Selfish Class. Describes how to overcome psychological problems related to the reuse of existing artifacts (e.g. source code).

References:

Ambler, S. "An Introduction to Process Patterns", Internet: http://www.ambysoft.com/processPatterns.pdf, 01-07-05

Foote, B., and Yoder, J. "The Selfish Class", in Martin, R., et al., eds. "Pattern Languages of Program Design 3", pp. 451-470, Wokingham: Addison Wesley Longman, 1998.

Organisational Patterns

Description: Organisational Patterns describe the structure and practices of human software development organisations.

Example: Design by Team. A general pattern that describes how people can work together to design and develop software.

References:

Harrison, N. "Organizational Patterns for Teams", in Coplien, J., and Schmidt, D., eds. "Pattern Languages of Program Design 2", pp. 345-352, Wokingham: Addison-Wesley Publishing, 1996.

"Organizational Patterns", Internet: http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns?OrganizationalPatterns, 01-03-21

Educational Patterns

Description: Educational Patterns describe proven techniques used for teaching technical knowledge.

Example: Iterative Course Development. Describes the use of iterative improvement to develop a course that takes the needs of all kinds of students into account.

Reference:

Anthony, D. "Patterns for Classroom Education", in: Coplien, J., and Schmidt, D., eds. "Pattern Languages of Program Design 2", pp. 391-406, Wokingham: Addison-Wesley Publishing, 1996.

Patterns on Pattern Writing

Description: Patterns on Pattern Writing describe and demonstrate writing practices which have been observed to be particularly effective.

Example: Code Samples. Describes the use of code samples to improve the quality of a pattern description.

Reference:

Meszaros, G., and Doble, J. "A Pattern Language for Pattern Writing", Internet: http://www.hillside.net/patterns/Writing/pattern_index.html, 01-07-05

Anhang 15: Aufstellung angeschriebener Mailinglisten

1. Testmuster

Beschreibung: Eine Mailingliste, auf der diskutiert wird, wie Muster für den Bereich des Tests von Software genutzt werden können.

Adresse (Email): `test-patterns@yahoogroups.com`

Adresse (WWW): `http://groups.yahoo.com/group/test-patterns`

Anzahl der Abonnenten: 100

2. J2EE-Muster

Beschreibung: Eine Mailingliste, die sich mit der Anwendung von Entwurfsmustern in der Java 2 Enterprise Edition beschäftigt.

Adresse (Email): `j2eepatterns-interest@java.sun.com`

Adresse (WWW): `http://archives.java.sun.com/j2eepatterns-interest.html`

Anzahl der Abonnenten: 1159

3. Muster

Beschreibung: Eine Mailingliste, die sich mit allgemeinen Themen wie der Beschreibung und Darstellung von Mustern beschäftigt.

Adresse (Email): `patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

4. Business Pattern

Beschreibung: Eine Mailingliste, die sich mit allgemeine Prozess- und Organisationsmustern beschäftigt.

Adresse (Email): `business-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

5. Musterdiskussion

Beschreibung: Eine Mailingliste, auf der allgemeine Aspekte von Mustern diskutiert werden. Dazu zählen Themen wie die Organisation von Mustern in Musterorganisationsschemen oder die Bedeutung von Mustern.

Adresse (Email): `patterns-discussion@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

6. Muster nach GAMMA et al.

Beschreibung: Hier werden die von GAMMA et al. publizierten Entwurfsmuster diskutiert (vgl. [Gam+94]).

Adresse (Email): `gang-of-4-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

7. Muster nach BUSCHMANN et al.

Beschreibung: Hier werden die von BUSCHMANN et al. publizierten Muster diskutiert (vgl. [Bus+98]).

Adresse (Email): siemens-patterns@cs.uiuc.edu

Adresse (WWW): http://www.hillside.net/patterns/Lists.html

Anzahl der Abonnenten: unbekannt

8. Organisationsmuster

Beschreibung: Eine Mailingliste, auf der Organisationsmuster diskutiert werden.

Adresse (Email): organization-patterns@cs.uiuc.edu

Adresse (WWW): http://www.hillside.net/patterns/Lists.html

Anzahl der Abonnenten: unbekannt

9. IPC-Muster

Beschreibung: Eine Mailingliste, auf der Architektur- und Entwurfsmuster zu den Themen Verteiltes Rechnen, Paralles Rechnen und Interprozesskommunikation (IPC) diskutiert werden.

Adresse (Email): ipc-patterns@cs.uiuc.edu

Adresse (WWW): http://www.hillside.net/patterns/Lists.html

Anzahl der Abonnenten: unbekannt

10. Power-Builder-Muster

Beschreibung: Eine Mailingliste, auf der die Anwendung von Mustern im Sybase Power Builder diskutiert wird.

Adresse (Email): pb-patterns@cs.uiuc.edu

Adresse (WWW): http://www.hillside.net/patterns/Lists.html

Anzahl der Abonnenten: unbekannt

11. CORBA-Muster

Beschreibung: Eine Mailingliste, auf der Architektur- und Entwurfsmuster in Beziehung zu CORBA diskutiert werden (vgl. [MoMa97]).

Adresse (Email): `corba-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

12. Konfigurationsmanagementmuster

Beschreibung: Eine Mailingliste, auf der die Anwendung von Mustern im Softwarekonfigurationsmanagement diskutiert wird.

Adresse (Email): `scm-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

13. Managementmuster

Beschreibung: Eine Mailingliste, auf der die Anwendung von Organisations- und Prozessmustern für die Komplexitätsbewältigung und Entkopplung von Systemen diskutiert wird.

Adresse (Email): `dacm-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

14. Telekommunikationsmuster

Beschreibung: Eine Mailingliste, auf der die Anwendung von Mustern in der Telekommunikationsbranche diskutiert wird.

Adresse (Email): `telecom-patterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

15. Antimuster

Beschreibung: Eine Mailingliste, auf der die Anwendung von Antimustern diskutiert wird (vgl [Bro+98]; [Bor+00]).

Adresse (Email): `antipatterns@cs.uiuc.edu`

Adresse (WWW): `http://www.hillside.net/patterns/Lists.html`

Anzahl der Abonnenten: unbekannt

Anhang 16: Zusammenfassung der Mustereigenschaften E1 bis E15

E1. Ein Muster ist eine Vorlage.

E2. Ein Muster ist vollkommen.

E3. Ein Muster ist ein Vorbild.

E4. Ein Muster erscheint regelmäßig immer wieder.

E5. Ein Muster kann wieder und wieder angewandt werden.

E6. Ein Muster beruht auf der Kombination einzelner Elemente.

E7. Ein Muster verändert sich bei jeder Anwendung, das generelle Schema wird jedoch beibehalten.

E8. Muster fixieren Expertenwissen.

E9. Muster vereinheitlichen die Kommunikation von Experten.

E10. Muster wurden schon mehr als einmal erfolgreich angewandt - „Rule of Three."

E11. Muster sind bewährte praktische Vorgehensweisen, keine innovativen Lösungen.

E12. Ein Muster beschreibt ein Problem-Lösungs-Paar.

E13. Ein Muster beschreibt die Kräfte bzw. Umstände, durch die es in seiner Lösung beschränkt ist.

E14. Muster lassen sich isoliert und in relativ kurzer Form beschreiben.

E15. Muster sind unabhängig von objektorientierten Konzepten.

Anhang 17: Exakte Prozentangaben für Frage 6A (vgl. Abbildung 37)

Nr.	Beschreibung	Ich widerspreche	Ich tendiere dazu zu widersprechen	Ich tendiere dazu zuzustimmen	Ich stimme dem zu	Ich stimme dem nachhaltig zu	Anzahl der Nichtantworten
N1	„Muster unterstützen die Entwicklung von Software mit definierten Eigenschaften“ ([Bus+98], S.7). Dabei zielen Muster explizit auf nichtfunktionale Eigenschaften von Software ab.	2%	8%	31%	39%	20%	6
N2	„Muster helfen bei der Entwicklung komplexer, heterogener Softwarearchitekturen“ ([Bus+98], S. 7). Muster sind zwar nicht immer besser als eigene Lösungen, letztlich vereinfachen sie jedoch in vielen Fällen die Anwendung von bewährten Vorgehensweisen.	2%	14%	24%	36%	24%	4
N3	Muster verbessern die Dokumentation des Prozesses der Systementwicklung (vgl. [Bus+98], S. 6; [ScSt95], S. 12)	0%	10%	20%	41%	29%	3
N4	Muster verbessern die Erweiterbarkeit, die Portierbarkeit und die Wiederverwendbarkeit von Software (vgl. [ScCl99], S. 18; [Helm95], S. 338f; [Clin96], S. 47).	1%	9%	32%	36%	22%	3
N5	Muster verbessern die Klarheit eines Entwurfs (vgl. [ScCl99], S.18).	0%	7%	20%	47%	27%	3

(Fortsetzung auf nächster Seite)

Nr.	Beschreibung	Ich wider-spreche	Ich tendiere dazu zu wider-sprechen	Ich tendiere dazu zuzu-stimmen	Ich stimme dem zu	Ich stimme dem nach-haltig zu	Anzahl der Nichtant-worten
N6	Muster geben Programmierern in der Wartungsphase einen Rahmen für vorzunehmende Änderungen, der garantiert, dass die ursprüngliche Entwurfsvision beibehalten wird (vgl. [Clin96], S. 48).	13%	36%	30%	16%	5%	4
N7	Muster sind mitunter im Quellkode nur schwer zu erkennen, obwohl der Quellkode unter Nutzung von Mustern entwickelt wurde ([Mann00], S. 15; [Bosc98], S. 19).	3%	29%	38%	21%	8%	6
N8	Muster erhöhen den Aufwand zur Laufzeit durch die Einführung einer zusätzliche Ebene der Abstraktion (vgl. [ScCl99], S.18).	26%	42%	22%	8%	2%	7
N9	Muster verbessern die Kommunikation zwischen allen am Systementwicklungsprozess beteiligten Personen (vgl. [ScSt95], S. 12; [Bec+01], S. 10; [Helm95], S. 339).	1%	1%	9%	32%	57%	3
N10	Muster unterstützen die Integration neuer Entwickler (vgl. [Helm95], S. 339; [Schm01], S. 8).	1%	10%	33%	35%	20%	4
N11	Mitunter ist es schwierig zu entscheiden, welches Muster ein gegebenes Problem am besten löst (vgl. [Shaw94], S. 4).	4%	31%	42%	18%	5%	2

(Fortsetzung auf nächster Seite)

Nr.	Beschreibung	Ich wider-spreche	Ich tendiere dazu zu wider-sprechen	Ich tendiere dazu zuzu-stimmen	Ich stimme dem zu	Ich stimme dem nach-haltig zu	Anzahl der Nichtant-worten
N12	Muster sind für eine direkte Anwendung nicht ausreichend spezifiziert (vgl. [ScSt95], S. 12).	14%	46%	24%	12%	4%	4
N13	Die Suche nach Mustern ist ein personalintensiver Vorgang (vgl. [Schm01], S.7).	0%	4%	24%	42%	30%	9
N14	Einige Muster sind nur schwer zu erlernen (vgl. [Clin96], S. 49).	12%	44%	27%	14%	4%	4
N15	Muster führen Entwickler dazu zu glauben, mehr zu wissen, als es tatsächlich der Fall ist (vgl. [Helm95], S. 339; [Schm01], S. 6).	16%	41%	23%	14%	6%	6
N16	Muster helfen beim Übergang zur Nutzung objektorientierter Technologien (vgl. [Schm01], S. 7).	6%	17%	36%	23%	18%	12
N17	Muster erhöhen die Vorhersagbarkeit von Entscheidungen in Analyse, Entwurf und Programmierung und tragen somit zu einer Standardisierung von Modellen bei.	2%	12%	41%	35%	10%	8
N18	Muster erfordern im Vergleich zu anderen Wiederverwendungsansätzen eine wiederholte Implementierung ([Mann00], S. 15).	10%	29%	32%	21%	8%	8

(Fortsetzung auf nächster Seite)

Nr.	Beschreibung	Ich wider-spreche	Ich tendiere dazu zu wider-sprechen	Ich tendiere dazu zuzu-stimmen	Ich stimme dem zu	Ich stimme dem nach-haltig zu	Anzahl der Nichtant-worten
N19	Muster verlassen sich nur auf die Erfahrung aus vorangegangenen Projekten. Sie generieren keine neuen Lösungen (vgl. [Pesc97], S. 130).	8%	29%	34%	18%	10%	8
N20	Der Einsatz von Mustern ist nicht mit hohen Kosten verbunden (vgl. [See+00], S. 447).	2%	10%	26%	44%	18%	7
N21	Muster vermeiden Kosten, die mit der Neuerfindung von Lösungen verbunden wären (vgl. [See+00], S. 447).	1%	12%	34%	37%	14%	6
N22	Muster verringern das Risiko in Systementwicklungsprojekten (vgl. [Schm01], S. 5).	7%	11%	39%	33%	10%	7

Anhang 18: Freie Angaben zu Frage 6A (vgl. Abschnitt 6.3.6)

„You can regard patterns as a compression method. For example, describing a design with patterns takes less space than doing it without. Patterns compress without trading precision for space."

„Using design patterns without language specific good implementation examples may cause more harm than good. For instance, many of the C++ examples in the GoF book are simply bad. Since patterns are often used by newbies it is of utmost importance than the examples leads the reader in the right and not the wrong direction when it comes to the actual implementation."

„They are too informal they should be validated (argument to show that solution solves the problem)."

„I've used patterns in the design and implementation of large software systems. They are a good mechanism for communicating abstractions and discussing alternative approaches. I'm not that dogmatic about their use, however."

„Patterns are only part of the total solution. They are an excellent communications media between individuals on a project."

„Patterns can make developers happy, when they understood and implemented a new concept(pattern) successfully sometimes too many patterns result in over-engineered solutions it is easier to introduce flexibility than to get rid of unneeded flexibility."

„Some background: during my time as a research assistant at the university, I dealt a lot with all kinds of pattern related question. In my current affiliation, patterns are not really supported. I am using them more on a personal basis. There's a good chance I'll use patterns more frequently in near future. Personally, I really like the idea behind patterns. I believe, however, that the term pattern is somewhat excessively stressed by some individuals. I believe patterns are valuable for many steps in software development, but I do not believe they can solve all issues raised by the inherent complexity of today's software systems. For me, it is a good concept ‚above single objects'. Not more - and not less!"

„Object-oriented patterns greatly aid understanding how to use object-oriented concepts properly.“

„I think that hype (especially the hype surrounding MVC) and lazy dissemination of patterns understanding (bad seminars about Singleton) are damaging to the cause of good development of pattern usage.“

„Unfortunately most developer don't know patterns, if they are using an object-oriented language, like Java. most of them use Idioms like Event Generator, but they don't know that this is just a application of GOF's Observer/Subscriber pattern. If you need to understand the design behind some Java APIs you need to understand patterns. But most developers hack. Patterns avoid long, ugly methods. I use refactoring to reduce long methods and I use Patterns to avoid this refactoring. We need more people that teach and proclaim the benefit of using patterns.”

„For some reason, it took a lot of time for me to understand the pattern language by various Schools of Thought. For this reason, I have had bitter experiences trying to understand the various terminologies that has proliferated even among the leading pattern advocates like GOF and Siemens. I would wish it were possible that there was less confusion regarding the terminology and a common standard for expressing a pattern is developed. Once and for all, the patterns are standardized. My experience with patterns have indicated that it takes a lot of time and its easy for you to misinterpret it. But once understood, well, you will see the dividents. I like patterns if it was possible for someone to guide me and the books are not the best of places. I must admit that I still am struggling to create a pattern language of my own.”

„Once one is familiar with patterns, they probably flow naturally, and the ‚pattern mining’ process probably becomes more efficient. When first introduced to the concept, though, current pattern terminology tends to be confusing and abstract; only with experience does the use of patterns begin to come naturally.”

„Patterns are more useful when one has already some experience in the context. They can not help understand the context.”

„Patterns reduce complexity and thus helps to comprehend an unknown design for new developers, though they have to be familiar with the most known patterns. The ability to handle abstraction is a prerequisite."

„Patterns are good - without them I would not be a respected coder/designer!"

„For many of these questions my answer would be ‚It depends' or ‚It can be'. There are trade-offs to be made when using patterns. Patterns can be very useful but they don't necessarily make a hard problem easy - they just help in solving the problem."

„Patterns help elucidate the differences between surface structure and deep structure of types of problems. Experienced developers and designers recognize the deep structure for the sake of less experienced developers and designers (like artificial intelligence systems do), so less experienced designers do better - if they understand the pattern. There's no harm in that!"

„The primary value of patterns, in my experience, is that they facilitate communication of design concepts, between software designers, and as software designers try to understand software written by others. When the members of a team are all familiar with a set of patterns, they provide a concise vocabulary for discussing design concepts. This value is greatly diminished, and patterns may even cause more harm than good if some members of the team are unfamiliar with the patterns being used and talked about by others."

„Risks: are hardly related to software design issues, so patterns do not play a major rule. Maintenance: patterns help understanding an existing system when there is a short hint given which role a specific class plays to other classes I would not overestimate the use of pattern tools. Most patterns are basically ideas, and contain little code to implement. A very global base class could make your design worse rather than save some lines of code. Templates can sometimes help here."

„It was difficult for me to decide whether to agree or disagree with the statement above, ‚Patterns rely on the experience of previous projects. They do not generate new solutions.' I have observed that OO design involving competent use of patterns can lead to quicker applications with smaller memory foot prints than OO applications designed without patterns. In some cases this has been due to patterns such as fly-weight that directly addresses the

memory footprint issue (and has improved speed as a side effect), in other cases the use of patterns gave a deeper level of understanding of what was being designed, which lead to a design that was fast because it was simple, and that could be optimised because it was simple."

„As an academic, I have mostly been involved with creation rather than application."

„Usability patterns are the ones I am most familiar with. These have not been highlighted in this survey. Responses are partly from the usability patterns perspective, but not entirely."

„The pattern form can capture design experience and design decisions. Documenting a design with the pattern form even if the design decisions are not proved patterns is a good idea."

„My experience with patterns is this: I'm a student about to begin my masters thesis in computer science and have chosen patterns and their implications in the development process as my headline (Do they in reality improve communication, and between who?). Unfortunately most firms I have contacted (now counting 5 major companies) state that it is a very relevant question and than they consider patterns a very applicable tool (in whatever field they have chosen to use it in), but unfortunately there is no time in their schedules to get involved in such a project. It is my feeling that they (till now) see this project as a pressure to really use patterns ‚after the book' and this will take too much time. I hope to get some data soon."

Die folgenden drei Anmerkungen wurden als freie Antwort am Ende des Fragebogens gegeben. Sie sind jedoch hier aufgeführt, da sie sich ebenfalls auf Vor- bzw. Nachteile der Anwendung von Mustern beziehen.

„One big problem with patterns is that individual ones are not all that useful by themselves. It is having a structured and hierarchical pattern language that is useful, letting people go from abstract to concrete, at different scales. Lots of books out there with lots of isolated patterns are not very useful. This makes it hard to find things. The GoF book has lots of links between their patterns, but they are all at the same level of architecture (i.e. a few objects at most). Need macro-design, as well as micro-Design Patterns."

„This survey make impression of a fairly formalistic approach to patterns. Patterns are more literature than technique, they have a different value system than formal approaches."

„I have spent several years participating in the patterns community, writing patterns, attending workshops, and on the whole, I have been disappointed. What the industry needs is fewer, better patterns, some general-purpose, some domain-specific."

Ehrenwörtliche Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt sowie die aus fremden Quellen direkt oder indirekt übernommenen Gedanken als solche kenntlich gemacht habe.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ich erteile der Technischen Universität Dresden ein unentgeltliches, zeitlich unbefristetes, nicht ausschließliches, nicht übertragbares, unwiderrufliches einfaches Nutzungsrecht an der Diplomarbeit für beliebige Medien, und zwar sowohl für schutzfähige Ergebnisse als auch für urheberrechtlich schutzfähige Programme.

Helsinki, den 14. Oktober 2001 - Thoralf Czichy -