Mustergestützter Entwurf und Dokumentation mit Entwurfsmustern

– Erfahrungen aus einem SAP-Projekt –

Georg Odenthal, Klaus Quibeldey-Cirkel
57068 Universität Siegen, Technische Informatik
{odenthal,quibeldey}@ti.et-inf.uni-siegen.de

Ausgehend von einer Darstellung des Musterbegriffs stellen wir zwei Verfahren für die *Übertragung* von Mustern in den Entwurf vor. Die Arbeitsschritte werden an Beispielen aus einem Realprojekt geschildert. Die *Dokumentation* mit Entwurfsmustern ist die Weiterführung des mustergestützten Entwurfs. Wir zeigen, daß durch Muster referenzierte Entwurfskomponenten durch diese Muster auch effizient dokumentiert werden können, und stellen eine erweiterte Mustergliederung zur Dokumentation von wiederverwendbaren Komponenten (*Frameworks*) vor.

1 Einleitung

Die industrielle Softwaretechnik ist im Vergleich mit anderen Ingenieurdisziplinen in puncto „nachvollziehbare Konzepte“ und „Dokumentation von Arbeitsergebnissen“ immer noch unterentwickelt. In diesem Kontext sind Entwurfsmuster eine wichtige Entdeckung für die praktische Informatik [1, 5]. Muster bilden ein ausdrucksmächtiges *Vokabular*, das eine effiziente Kommunikation der Entwerfer erlaubt; eine Sprache des Entwurfs entsteht. Auch die psychologische Wirkung der Muster ist nicht zu vernachlässigen: Programmierer können durch sie die Qualität ihrer Arbeit darstellen. Das Potential der Muster für die Softwareproduktion in wenigen Worten:

- ein *durchgängiges Konzept* für die effiziente Aufbereitung und Darstellung von Analyse-, Architektur- und Entwurfswissen bis hin zu Stilregeln für die Implementierung;
- ein *didaktisches Vehikel* zur Vermittlung allgemeinen und firmenspezifischen Know-hows;
- der *mustergestützte Entwurf* und die *Dokumentation in der Musterform* sind Gegenstand des Beitrags.

Die in den folgenden Abschnitten vorgestellten Konzepte zur Verwendung von Mustern im Entwurf und zur Dokumentation werden an Beispielen aus einem SAP-Projekt erläutert. Der zusätzliche Aufwand für den Leser wird unseres Erachtens durch eine realistische Darstellung der eigentlichen Inhalte kompensiert. Aus Platzgründen können wir nur ein Beispiel im Rahmen dieses Beitrags darstellen. Weitere finden Sie unter: www.ti.et-inf.uni-siegen/Entwurfsmuster.

1.1 Entwurfsmuster

Noch vor kurzem drehte sich die Diskussion zur „Objektorientierung“ um die richtige Wahl der Implementierungssprache oder um die Eigenschaften des Objekt-Paradigmas [10]. Spätestens seit dem Erscheinen der „Entwurfsmusterbibel“ der *Gang of Four* (GoF) [5] hat sich ein neues Thema etabliert und die Diskussion auf eine höhere Ebene verlagert. Es geht nicht mehr um das Handwerkszeug, sondern um die vergleichende Bewertung und den Einsatz von *Mikro-Architekturen*. Der oft ausbleibende Erfolg beim Einsatz objektorientierter Konzepte scheint mit Mustern erreichbar: Vielfach erprobte und von Experten entworfene Lösungsstrategien können in eigenen Entwürfen eingesetzt werden.

Was also sind Entwurfsmuster? Unter dem Begriff „Muster“ assoziiert man eine Vorlage, nach der etwas hergestellt wird, ein beispielhaftes Vorbild oder eine regelmäßige Struktur. In der Softwaretechnik bezeichnet ein Muster, angelehnt an [5], ein *Problem-Lösungs-Paar*. Der hinreichend verallgemeinerten Darstellung eines *immer wiederkehrenden* Entwurfsproblems wird eine praxiserprobte Lösung gegenübergestellt. Beispiele dienen der Illustration des abstrahierten Problems. Die Dualität legt die *Kräfte* offen (Randbedingungen und Konsequenzen) und zeigt den Weg vom Problem zur Lösung. Sie führt so zu den eigentlichen Fragen, vor die sich der Entwerfer gestellt sieht. Der wohl bekannteste Vertreter ist das in [7] erstmalig vorgestellte MVC-Muster.

Muster haben eine *literarische Form*: Im Zentrum steht die Beschreibung mit Worten anhand einer festen Gliederung [5]. Strukturierte Grafik kann zur Verdeutlichung eingesetzt werden. Formalismen spielen eine untergeordnete Rolle. Die Kenntnis von Musterbüchern (*Pattern Languages* [1]) kann die Produktivität des Entwerfers erhöhen. Diese Kenntnis kann durch das Lesen der Bücher allein aber nicht erlangt werden. Die praktische Erprobung des Wissens ist ausschlaggebend. Muster müssen in ihrem Potential aber auch in ihren Restriktionen verstanden sein, um sie erfolgreich einzusetzen. Je mehr Muster der Entwerfer verstanden hat, desto häufiger wird er Musterkandidaten in seinen Problemen erkennen.

Generell kann zwischen *fachlichen* und *technischen* Mustern unterschieden werden. Fachliche Muster stammen aus der Analyse – die Abstraktion grundlegender fachlicher Eigenschaften [3]. Ein Beispiel ist die *Buchung*: der Transfer einer Teilmenge als Transaktion. Technische Muster haben direkten Bezug zur Implementierung (z. B. *Composite*) oder zur Systemarchitektur (z. B. MVC).

Was sind Muster nicht? Goldgräberstimmung herrscht in so manchen Köpfen – das Wissen aus dem eigenen Problembereich soll in die Musterform gebracht werden. An dieser Stelle ist aber Vorsicht angebracht: schnell ist das mächtige Konzept verwässert, die Musterinflation droht. Die Idee der Muster sollte nicht als schlichte Repräsentationsform mißbraucht werden. Der Musterautor muß sich immer die Frage stellen: Ist der Inhalt so allgemeingültig, daß aus der Form viele verschiedene Lösungen in unabhängigen Problembereichen abgeleitet werden können? Muster sind ein *Destillat* aus zahllosen konkreten Beispielen. Es gilt: Je kompakter eine Mustersprache, desto leichter erlernbar und desto mächtiger ist sie.

Diese Ausführungen sollten Sie nicht abschrecken, den eigenen Problembereich auf Musterkandidaten hin zu untersuchen. Es gibt noch viele Muster zu entdecken, und Sie werden sicherlich fündig. Muster müssen reifen: Eine erste Niederschrift sollte zunächst in der Praxis erprobt und mit anderen diskutiert werden, die Ergebnisse in Revision und Vervollständigung der Musterbeschreibung einfließen. Erweist es sich als nicht „mustergültig“ genug, löschen Sie es!

Im Gegensatz zur Entdeckung neuer Muster steht die Verwendung der Musterform als *Dokumentationsmittel*. Ausgehend von der Ausprägung eines Musters in einem Entwurf wird sie als syntaktisches und semantisches Gerüst zur Begründung einer Entwurfsentscheidung herangezogen.

1.2 Das Projekt

Ziel des Projekts, aus dem die Beispiele stammen, war die Entwicklung einer objektorientierten Kommunikationskomponente zwischen dem *SAP Business Object Repository* und der *Open Scripting Architecture*, kurz OSA, von Apple/IBM. OSA ist vergleichbar mit *OLE Automation*. Salopp gesagt, ging es darum, R/3 „OSA-fähig“ zu machen. Das *Business Object Repository* ist die Verwaltungsinstanz für *Business-Objekte*, die im *SAP Business Workflow* verwendet werden. *Business-Objekte* ermöglichen den objektorientierten Zugriff auf die relational verwalteten Daten von R/3. Der Anwender definiert im *Business Object Repository* Objekttypen aus Datenbankfeldern und Funktionsbausteinen (entsprechend Attributen und Operationen einer Klasse). Neben der Erstellung des OSA-Servers war es ein wichtiges Ziel des Vorhabens, Entwurfsmuster bei der Entwicklung und Dokumentation zu verwenden. Ein möglichst großer Teil des OSA-Servers sollte so wiederverwendbar werden, um z. B. andere Anbindungen darauf aufzusetzen [8].

2 Der mustergestützte Entwurf

Wir stellen zwei Vorgehensweisen zur Verwendung von Mustern des GoF-Typs dar: *Musterausprägung* und *Musteridentifikation* – entwerfen mit Mustern und flexibilisieren durch Muster. Die Tätigkeit des Entwerfers bei der Musterausprägung ist das Zuordnen von erkannten Eigenschaften des Problembereichs auf die verallgemeinerte Sicht der Muster und umgekehrt. Im Gegensatz hierzu stehen bei der Musteridentifikation die Klassen und Beziehungen annähernd fest; der Entwurf ist schon relativ stabil. Der Einsatz von Mustern zu diesem Zeitpunkt kann aus der Unzufriedenheit mit bestimmten Eigenschaften des Entwurfs resultieren, beispielsweise einer mangelnden Flexibilität,

und/oder aus dem Willen, einen Erkenntnisgewinn in den Entwurf einzubringen. Besonders hinsichtlich der Produktion wiederverwendbarer Einheiten (in diesem Kontext sind das Frameworks) eignet sich die nachträgliche Zuordnung von Mustern. Ordnet man die Vorgehensweisen in ein Phasenmodell ein, so erfolgt die Ausprägung von Mustern beim Übergang von der Analyse- zur Entwurfsphase. Die Identifizierung von Mustern im Entwurf steht am Ende des Entwurfs und kann auch aus der Implementierung heraus angestoßen werden.

Generell lassen sich die Vorgehensweisen in vier Arbeitsschritte unterteilen. Die ersten zwei sollen vor allem den Planungs- und Entwicklungsprozeß explizit machen; die Notation ist an [5] angelehnt, aber um Ausdrucksmittel für die Zuordnung zum Problembereich erweitert. Die entstehenden Diagramme im zweiten Schritt dienen vor allem der *Kommunikation* unter den Entwicklern und können „frei Hand“ oder mit einem entsprechend konfigurierten grafischen Editor erstellt werden. Der Anspruch ist, daß sich der mit dem Problembereich vertraute Entwickler hier gleichermaßen zurechtfindet wie der Musterexperte. Die Schritte drei und vier stellen die Umsetzung der getroffenen Entscheidungen in einer Entwicklungsumgebung dar. Aus diesem Grund wurde auch die Notation gewechselt (Klassendiagramme der Entwicklungsumgebung *objectiF*). Zur Kennzeichnung der Musterausprägungen im Entwurf gibt es (noch) keine einheitliche Notation. Wir verwenden im Beispiel die *Pattern-Role-Annotation* von Gamma [4]. Die Annotation mit Rahmen (Venn-Diagramme) ist geeigneter, um das Zusammenspiel mehrerer Musterausprägungen darzustellen. Der Vorschlag in der UML V.0.9 (www.rational.com) erscheint uns zu überladen (Muster als *Use Cases* notiert).

Die zwei Vorgehen – Musterausprägung und Musteridentifikation – sind in der folgenden Beschreibung der Arbeitsschritte zusammengefaßt:

1. Aus einem Musterbuch ein **geeignetes Entwurfsmuster auswählen**, bzw. in einem konkreten Entwurf einen **Musterkandidaten identifizieren**. Beide Tätigkeiten erfordern die Kenntnis von Mustern; je mehr Muster ein Entwerfer parat hat, desto häufiger wird er musterbasiert entwerfen, respektive Klassenstrukturen als Musterkandidaten identifizieren.

2. **Übertragung des Musters in den Entwurf**. Bei der Musterausprägung geht es vor allem um die Zuordnung der Klassen, Operationen und Beziehungen: Wie heißen die Klassen des Musters im Problembereich? Welche zusätzlichen Dienste werden in die Klassen aufgenommen? Die allgemeinen Vorgaben des Musters geben eine gute Richtschnur ab, Klassen nicht mit mehreren Rollen zu überfrachten und die richtigen Beziehungen zu wählen, analog dem Grad der Kopplung zwischen den Klassen des Musters.
   Zuordnung der Entwurfskomponente auf das Muster. Bei der Musteridentifizierung steht die Frage, welche Rollen die Klassen des konkreten Entwurfs im Muster einnehmen und ob diese Zuordnung den gewünschten Effekt hat, im Vordergrund. Um die Annahmen genauer zu überprüfen, sollten auch die konkreten Operationen und Attribute zugeordnet werden.

3. **Strukturelle Änderungen des Musters bei der Ausprägung**. Manchmal kann es sinnvoll sein, ein Muster zu vereinfachen und Klassen aber auch Beziehungen des Musters bei der Ausprägung nicht oder verändert in den konkreten Entwurf zu übernehmen.
   Strukturelle Veränderungen im Entwurf. Meist wird es erforderlich sein, den Entwurf durch im Muster vorhandene Klassen zu vervollständigen. Dies sind oft die abstrakten Klassen, die die Flexibilität und somit auch die Änderungsstabilität in einen Entwurf einbringen.

4. **Technische Klassen einführen**. Nachdem soweit eine Entsprechung zwischen Muster und Entwurf herbeigeführt wurde, ist es meist notwendig, weitere technische Klassen einzuführen. Beispiele hierfür sind *Container*-Klassen (Listen, Mengen) aus einer Klassenbibliothek. Auch eine Erweiterung der Klassenschnittstellen durch zusätzliche Dienste kann erforderlich sein (beispielsweise Laufzeittypüberprüfung: RTTI).

2.1 Beispiel zur Musterausprägung: Speicherung der Typinformation (Composite)

Im *Business Object Repository* werden die Objekttypen (Klassen) der *Business-Objekte* definiert und verwaltet. Die Schnittstelle einer solchen Klasse besteht aus einer Liste von Attributen und Operationen; eine Operation besitzt eine Liste von Parametern. Die Aufgabe besteht nun darin, eine Verwaltungskomponente für die Klassenschnittstellen zu entwickeln. Aufgrund der vorliegenden hierarchischen Strukturierung dieser Daten, bietet sich eine Konstruktion basierend auf dem *Composite*-Muster [5] an. Im ersten Ansatz würde der Entwurf wahrscheinlich wie in Bild 2 aussehen.

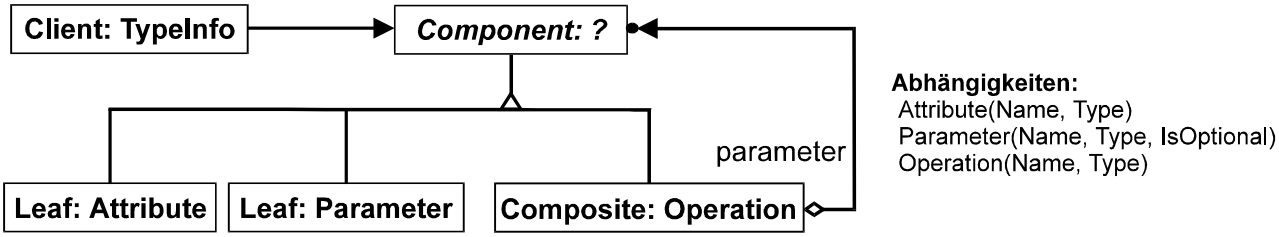

Bild 1: Der erste Strukturierungsansatz

Die Übertragung des Musters in den Problembereich (Bild 3, Schritt 2) führt zu dem Ergebnis, daß sich Attribute und Parameter der *Business-Objekte* unter der Sicht der hier gestellten Anforderungen nur gering unterscheiden und so eine Aufteilung unnötig ist. Als strukturelle Änderung (Schritt drei) ergibt sich, daß auf eine *Leaf*-Klasse gänzlich verzichtet werden kann. Statt dessen wird *Component* zu einer konkreten Klasse. Die Flexibilität des Musters wird hier also aus Gründen der Einfachheit bewußt verringert. Es sei angemerkt, daß die nachträgliche Erweiterung wieder leicht möglich ist durch die erneute Einführung von *Leaf*-Klassen. Das Potential des Musters bleibt somit erhalten. Um flexibel gehaltene Teile eines Entwurfs später leicht identifizieren zu können, ist eine Dokumentation, die von der Musterausprägung ausgeht, besonders geeignet.

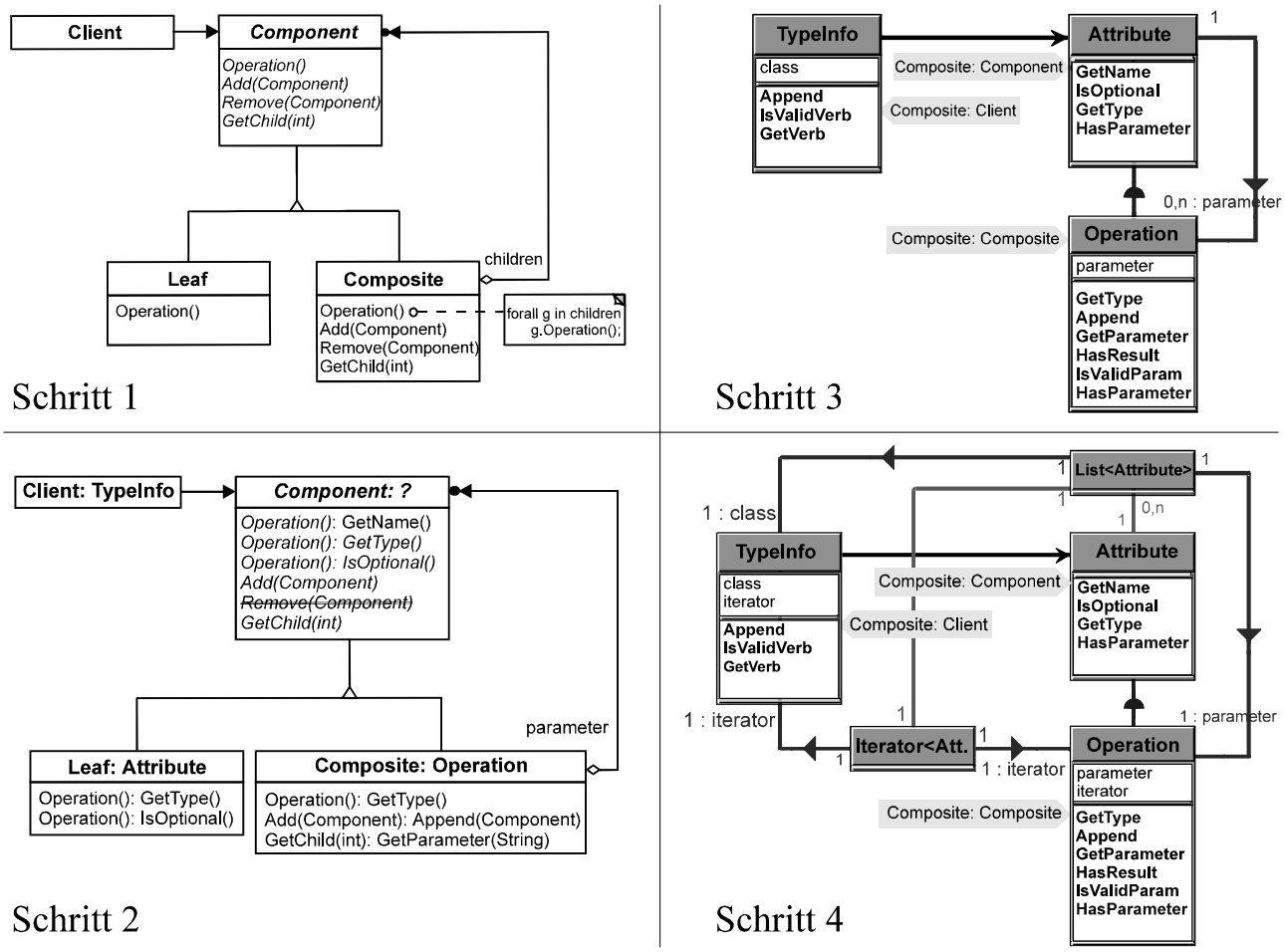

Bild 2: Musterausprägung am Beispiel

3 Dokumentation mit Entwurfsmustern

Folgend stellen wir dar, was Software-Dokumentation *mit* Entwurfsmustern ist und wie durch sie die Qualität und Verwendbarkeit und somit der Stellenwert der Dokumentation entscheidend verbessert werden kann. Bei dieser Art des Einsatzes von Mustern geht es nicht darum, neue Muster zu finden und zu beschreiben, sondern Musterbücher (auch eigene) zu verwenden. Das Vorgehen ist die Weiterführung des mustergestützten Entwurfs. Wurde ein Entwurf annähernd vollständig mit Mustern abgedeckt, stellen die Musterausprägungen die Ausgangsbasis für die Entwurfsdokumentation dar. Durch sie wurde das entwickelte System strukturiert und eine Abstraktionsebene oberhalb der Klassen geschaffen, der Entwurf unabhängig vom Problembereich semantisch strukturiert (Meta-Semantik). Die Dokumentation mit Entwurfsmustern stellt den Bezug zum Problembereich her: sie ist die Erklärung, *warum* und *wie* das Muster ausgeprägt wurde.

Übersicht: Ein Klassendiagramm und evtl. weitere Visualisierungen, die zur Verdeutlichung geeignet sind.
Intention: Die knappe Beschreibung der wesentlichen Intention. Der Grund, warum genau dieses Muster ausgeprägt wurde, sollte deutlich werden.
Motivation: Die Ausführliche Beschreibung der Zusammenhänge. Übersicht über die Komponente. Es kann sinnvoll sein, die "Entwicklungsgeschichte" anhand des Vorgehens zur Musterausprägung zu illustrieren. Auch Verweise auf Dokumente aus der Analysephase (z.B. der Bezug zu fachlichen Mustern).
Rollen: Hinter die Klassen des Musters werden die Klassen der Implementierung geschrieben. Diese Zuordnung ist für den Leser hilfreich, der das Muster kennt und sich über die spezielle Ausprägung informieren will. Eine stichpunktartige Beschreibung der Rolle, die die Klasse in der Musterausprägung einnimmt.
Zusammenarbeit: Beschreibung der Interaktion zwischen den Klienten und der Musterausprägung.
Konsequenzen: Beispielsweise die Erweiterbarkeit, Vergleich mit anderen Realisierungen, Einschränkungen.
Implementierung: Besonderheiten der Implementierung.

Bild 3: Gliederung für die Dokumentation von Musterausprägungen

Die Idee besteht darin, die Musterform auch als Strukturierung für die Dokumentation zu verwenden (Bild 3). Hierdurch und durch die bestehende Strukturierung des Entwurfs mit Mustern ist die Entwicklungsdokumentation vollständig vorstrukturiert, was das Dokumentieren erleichtert. Die so erreichte *Trennung von Struktur und Inhalt* der Dokumente ist eine wichtige Voraussetzung für die Konsistenzerhaltung zwischen Entwurf und Dokumentation sowie für eine Repräsentation als Hypertext. Derart strukturierte Dokumente eignen sich schließlich gut für die Verwendung in *Retrieval*-Systemen [11]. Mit adäquater Werkzeug-Unterstützung kann aus der passiven schnell überholten Papierdokumentation ein aktives *Informationssystem* werden. Die Dokumentation der Entwurfsentscheidungen erfolgt dann iterativ und verzahnt mit der eigentlichen Tätigkeit des Entwerfens.

Hot Spots, ***Frozen Spots***: Für die Dokumentation von Frameworks ist die gesonderte Beschreibung der festgelegten Eigenschaften des Frameworks und der durch den Benutzer anzupassenden bzw. zu erweiternden Komponenten essentiell (vgl. [9]). Aus dieser Beschreibung sollte eindeutig hervorgehen, welche Flexibilität vom Framework zu erwarten ist und wo dessen konzeptionelle Grenzen sind.
Rezept: Der C*ookbook*-Stil wurde u.a. von Johnson [6] verwendet, um den Einsatz eines Frameworks anhand eines schrittweisen, durch Beispiele ergänzten Vorgehens zu verdeutlichen. Diese Beschreibung kann durch ein *Ready-to-use*-Beispiel, das die direkte Ausführung und somit das Ausprobieren eines Frameworks erlaubt, ergänzt werden. Auch eine vorkonfigurierte Debugger-Sitzung eignet sich zum besseren Verständnis.
Ist eine feste Entwicklungs- und Laufzeitumgebung vorhanden, bilden interaktive Hilfen für die Nutzung eines Frameworks (Assistenten, *Wizzards*) das Optimum an Unterstützung für den Anwender.
Integrierbarkeit: Wie paßt sich das Framework in bestehende Umgebungen ein? (Beispielsweise Portabilität, Realisierung der Ausnahmebehandlung).
Verwendungen: Anhand der Verweise auf Einsatzbeispiele kann der Anwender am schnellsten die Eignung des Frameworks überprüfen.
Strukturelle Erweiterungen: Vor allem bei Inhouse-Entwicklungen kann es sinnvoll sein, den Entwurf eines Frameworks eingehend zu beschreiben, um dessen Erweiterbarkeit zu erleichtern. Ein Framework durchläuft mehrere Designzyklen, bis die *Hot Spots* identifiziert sind und so das richtige Maß an Flexibilität erreicht wurde. Wird es in bisher nicht vorgesehenen Bereichen eingesetzt, kann eine Erweiterung nötig sein. Aus beiden Gründen ist die Dokumentation des Entwurfs sinnvoll.

Bild 4: Erweiterte Gliederung für die Dokumentation von Frameworks

Frameworks sind *wertvoller* als andere Komponenten, da sie wesentlich teurer in der Entwicklung waren (Praxiserfahrungen liegen bei einem Faktor von 4-5) und in weiteren Projekten wiederverwendet werden sollen. Wir unterscheiden daher zwischen der Dokumentation von *bereichsspezifischen Komponenten*, bei der die Wiederverwendung nicht im Vordergrund steht, sondern „nur“ die möglichst einfache Nachvollziehbarkeit der Entwurfsentscheidungen, und der Dokumentation von *Frameworks* zur Wiederverwendung (Bild 4).

Zwei Bemerkungen zum Schluß: Auf Gliederungspunkte für die Musterverwaltung in Bibliotheken und die Zuordnung zu Bearbeitern und dem Vorgehensmodell (Versions- und Prozeßmanagement) haben wir hier verzichtet. Derartige Gliederungen lassen sich in modifizierter Form auch erfolgreich für die Dokumentation auf Klassen- und Operationsebene einsetzen [8].

4 Erfolge

- Die Qualität eines Entwurfs erhöht sich durch den Einsatz erprobten Expertenwissens.
- Der mustergestützte Entwurf als „sanfter Übergang“ zur Framework-Entwicklung: Eine Musterausprägung ist soweit flexibilisiert, daß der Weg zu einem Framework meist nicht mehr weit ist.
- Leichteres Ergründen eines Entwurfs durch die Musterabstraktion. Im Beispielentwurf reduzierte sich die Beschreibungs-Komplexität von über dreißig Klassen auf vier wesentliche Muster [8].
- Entwurfsentscheidungen werden explizit und damit nachvollziehbar. Die Denkwelt der Muster und die Strukturierung des Entwurfs durch Muster erleichtern die Dokumentation, indem diese gleichermaßen vorstrukturiert wird.
- Musterausprägungen sind eine vom Problembereich unabhängige *semantische Strukturierung* des Entwurfs (Meta-Semantik) und ein ungleich mächtigeres Konzept als die durch Anforderungen aus dem Problembereich motivierte Annotation mit Rahmen (S*ubjects* [2]).
- Der Wiederverwendungsgrad kann durch den Einsatz von Mustern gesteigert werden.

Literatur

1. Alexander, C.: A Pattern Language. New York: Oxford University Press 1977.
2. Coad, P; Yourdon, E.: Object-Oriented Analysis. Englewood Cliffs: Yourdon Press, Prentice–Hall Inc. 1991.
3. Fowler, M.: Analysis Patterns: Reusable Object Models. Addison-Wesley 1996.
4. Gamma, E.: Entwurfsmuster: ein weiteres Allheilmittel für die Softwareentwicklung? In: Beiträge der GI-Fachtagung Softwaretechnik ´96 am 12./13.9.96 in Koblenz.
5. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Reading, MA: Addison-Wesley 1995.
6. Johnson, R.: Documenting Frameworks Using Patterns. In: SIGPLAN 27(19), 1992, S. 63ff.
7. Krasner, G.; Pope, S.: A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80. In: JOOP 8/9, 1988, S. 26ff.
8. Odenthal, G.: Entwurf und Implementierung einer Schnittstelle zwischen dem SAP-R/3 Business Object Repository und der Open Scripting Architecture (OSA). Universität Siegen: Technische Informatik, 1996.
9. Pree, W.: Design Patterns for Object-Oriented Software Development. Reading, MA: Addison-Wesley 1994.
10. Quibeldey-Cirkel, K.: Das Objekt-Paradigma in der Informatik. Stuttgart: Teubner 1994.
11. Zendler, A.; Gastinger, S.; Haggenmüller, R.: Vergleichende Analyse von Werkzeugen zum Aufbau und Einsatz von Bibliotheken für wiederverwendbare Software-Dokumente. FAST e.V., München, Februar 1995.