Vorwort

“

”

Problem – Kontext – Kräfte – Lösung

Ein guter Compiler ist in der Sprache geschrieben, die er übersetzen soll. In Selbstanwendung des Musterkonzeptes schreibe ich also das Vorwort zum Thema in der *Musterform*.

Literarisches Bootstrapping

Die Hauptfragen dieses Buches sind *methodische* und *didaktische* der Softwaretechnik; ab einer gewissen Entwurfskomplexität (Entwurf als Prozeß und Produkt verstanden, Komplexität jenseits von hundert vernetzten Modulen) werden Entwerfer und Anwender vor Lehr- und Lernprobleme gestellt:

- Erfahrung ist *erprobtes* Wissen, sie setzt das Erlebnis voraus; Entwerfen in der Industrie läßt dies aber nicht zu: Im Sog neuer Techniken und kurzlebiger Märkte bleibt dem Entwerfer keine Zeit, um das Versuch-und-Irrtum-Prinzip auszuleben. Die Frage heißt: Wie kann der Unerfahrene aus der Erfahrung des Experten lernen? Methoden- und Faktenwissen allein genügen nicht, um *produktiv* zu werden. Umgekehrt: Wie kann der Experte seine Erfahrungen vermitteln – eingängig und in kürzester Zeit?

Transfer und Wiederverwendung von Erfahrungswissen

Orientierung in fremden Entwürfen

- *Wiederverwendung* ist die Maxime für die schnelle Markteinführung eines Entwurfs; Wiederverwendung im großen Stil erlauben objektorientierte *Frameworks,* das sind anpaßbare Systemkomponenten: Software-Halbfabrikate im allgemeinen, Komplettlösungen im Einzelfall.[2] Wiederverwendet werden hier sowohl der Programmcode als auch der Entwurf, der einer Familie von Anwendungen gemeinsam ist. Somit ist der Aufwand, um eine Anwendung aus einem Standard-Framework zu entwickeln, deutlich geringer als für eine Eigenentwicklung. Die Komplexität heutiger Frameworks, meist einige hundert Klassen und Kommunikationspfade, erschwert indes das Finden und Verstehen der Anpassungsstellen. Die Frage lautet: Wie kommt der Anwendungsprogrammierer zum notwendigen *Architekturverständnis?* Wie findet er sich rasch zurecht im fremden Entwurf?

Entwurf und Dokumentation komplexer Komponenten

- Objektorientierte Frameworks erfüllen am wirtschaftlichsten den Wiederverwendungs-Anspruch; sie sind aber auch schwierig zu entwerfen: Wie kann der Entwerfer die Anpassungsstellen flexibel gestalten? Wie dokumentiert er die *variablen* und die *festen* Aspekte seines Frameworks? Verallgemeinert handelt es sich hier um die alten Leiden des Software-Entwerfers, der sich im Wirrwarr seiner oder fremder Unterlagen zurechtfinden muß, um Entwurfsentscheidungen in der Software-Architektur zu revidieren.

Kontext

Das Problem steht im Mittelpunkt des Erfahrungsaustausches im GI-Arbeitskreis „Frameworks & Entwurfsmuster“. [3] Dieser setzt sich großenteils aus industriellen Framework-Entwicklern zusammen; dort bin ich zuständig für das Thema Entwurfsmuster. Der technische Kontext des Buches deckt sich mit den Fragen des Arbeitskreises; zwei Projekte bilden die Basis:

---

2 Die Übersetzung „Rahmen“ ist selten; gemeint sind Software-*Infrastrukturen*, die der Anwender für seinen Zweck um einzelne Funktionen erweitert.

3 http://www.uni-paderborn.de/cs/ag-engels/ag_dt/GI/gi-fg219.htm

Forschung und Praxis

1. ein Evaluierungsprojekt mit der SAP AG, Walldorf
   - Thema: Entwurf und Implementierung einer Schnittstelle zwischen dem SAP-R/3 Business Object Repository und der Open Scripting Architecture [Odenthal & Quibeldey-Cirkel, 1996]
   - Problem: Entwerfen und Dokumentieren mit Entwurfsmustern
2. ein Kooperationsprojekt mit der Fachhochschule Konstanz
   - Thema: HTML-basierte Softwaredokumentation mit Entwurfsmustern [Blachnik, 1997]
   - Problem: Entwurf eines werkzeuggestützten Dokumentationsmodells für die Nachdokumentation eines Frameworks aus der Fertigungstechnik

Lehre

Mit dem softwaretechnischen Kontext überlappt sich der fachdidaktische: Für das Studium der Technischen Informatik wurde ein Studienmodell entworfen – *Informatik-Systemtechnik*[4] [Lang et al., 1995; Quibeldey-Cirkel, 1994c, 1995b]. Hier dienen Entwurfsmuster als Leitbild für die berufliche Qualifizierung. Zentraler Teil dieses Studienmodells ist die Vorlesung „Objektorientierter Systementwurf".[5] Aus meiner Lehr-Erfahrung und der Betreuung von Studienprojekten konnte ich *fachdidaktische* Muster ableiten; sie ergänzen die Lehrmuster, die derzeit im Internet gesammelt werden – „Pedagogical Patterns: Successes in Teaching Object Technology".[6]

Kräfte

Gibt es andere Wege des Wissenstransfers und Erfahrungserwerbs? Wann sind diese vorzuziehen? Lehr- und Handbücher, Bauteilekataloge und DIN-Blätter – unsere Fachliteratur bietet sich an als Meterware. Vermittelt die traditionelle Prozeß- und Produktdokumentation Entwurfserfahrung in *wiederverwendbarer* Form?

---

4 http://www.ti.et-inf.uni-siegen.de/ECBS/leitbilder.htm
5 http://www.ti.et-inf.uni-siegen.de/courses/oos/oos.html
6 http://www-lifia.info.unlp.edu.ar/ppp/

Was sind die Kräfte, an deren Kompensation eine Lösung zu bewerten ist?

Ökonomie

- Lern- und Anwendungsökonomie des Transfermediums
  Effizienz des Zugriffs? Steile Lernkurve? Nachhaltigkeit des erworbenen Wissens? Aktualisierbarkeit des Wissens?

Ergonomie

- Benutzerakzeptanz des Transfermediums
  Einfach zu erlernen? Leserfreundlich? Gleichermaßen anwendbar vom Anfänger und Experten? Verläßlichkeit? Verbreitung?

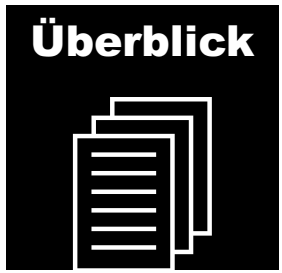
Der Prüfstein für die Kompensation dieser Kräfte heißt *Verfügbarkeit* und *Wiederverwendbarkeit* von Expertenwissen in allen drei Arten: Methoden- und Faktenwissen einerseits, Erfahrungswissen andererseits.

Lösung

Die Lösung des fachdidaktischen Problems – wie lehre ich Erfahrungswissen? – liegt in der Anwendung eines neuen Lehrvehikels: der *Musterform*. Sie vermittelt Erfahrungen kurz und bündig, so daß ein unerfahrener Entwerfer schnell produktiv wird. Entwurfsmuster und ihre Verknüpfung zu *Mustersprachen* verdichten das Expertenwissen eines Fachgebiets.

Die Lösung des softwaretechnischen Problems – wie entwerfe und dokumentiere ich komplexe Komponenten? – liegt in der methodischen Verzahnung beider Handlungen: „Documenting by Designing“. Die Verzahnung erreichen wir durch eine *Hypertext*-Dokumentationsmethode, die sich an Entwurfsmustern orientiert. Indem wir Form und Inhalt der Muster wechselseitig anwenden, werden unsere Entwürfe lesbarer und verständlicher; wir entwerfen und dokumentieren zugleich und schöpfen so aus der *dualen* Natur der Muster: sie sind sowohl *generativ*, die Lösung erzeugend, als auch *deskriptiv*, die Lösung beschreibend.

Die Lösungsstrukturen, ihre Anwendbarkeit und Folgerungen werde ich in dieser Arbeit zeigen und Praxisbeispiele geben.



Das aktuelle Schlagwort: Entwurfsmuster [1]

Zur Aktualität

Der Architekt Christopher Alexander hält die Gastrede auf der OOPSLA 96 – der bedeutendsten Konferenz über die objektorientierte Softwaretechnik. Sein Musterbegriff, entwickelt in den 70er Jahren für die Gebäude-Architektur und Städteplanung [Alexander et al., 1977], hat das Schlagwort geprägt. Es war wohl die Suche nach der Konzeption eines *Architektur-Handbuchs* für den Software-Entwurf (OOPSLA 92), die zur alexandrinischen Musterform führte. 1995 hieß der Bestseller der Informatik „Design Patterns: Elements of Reusable Object-Oriented Software" [Gamma et al., 1995]. Der Musterbegriff hat eine Bewegung ausgelöst, die ihresgleichen sucht; zwei internationale Konferenzen finden jährlich statt: „Pattern Languages of Programming" PLoP 94 …, EuroPLoP 96 …[2]

Begriffsbestimmung

Alltagssprachlich verstehen wir unter einem Muster dreierlei: eine *Vorlage*, nach der wir etwas herstellen, ein beispielhaftes *Vorbild* oder eine regelmäßige, sich wiederholende *Struktur*.[3] In allen drei Bedeutungen wird der Muster-

---

1 Beruht auf einem Beitrag für die GI-Zeitschrift „Informatik-Spektrum" [Quibeldey-Cirkel, 1996b].

2 http://hillside.net/patterns/conferences/

3 Interdisziplinäre Diskussionen zum Musterbegriff finden sich in [Coad, 1992; Hombach, 1995; Riehle, 1995].

begriff in der objektorientierten Softwaretechnik angewandt: konstruktiv als Vorlage (generative Entwurfsmuster), deskriptiv als Vorbild für die Dokumentation von Entscheidungen und strukturell als Orientierung in einem komplexen Entwurf. Muster sind erprobte Lösungsstrukturen für wiederkehrende Probleme; im Gegensatz zum Halbfabrikat Klassenbibliothek werden Entwürfe *uncodiert* wiederverwendbar. Die handlungsbezogene Musterform macht das Erfahrungswissen des Experten zugänglich; zur Disposition des Entwerfers stehen nicht nur Einzelklassen, sondern Klassenkonfigurationen – *Mikro-Architekturen*.

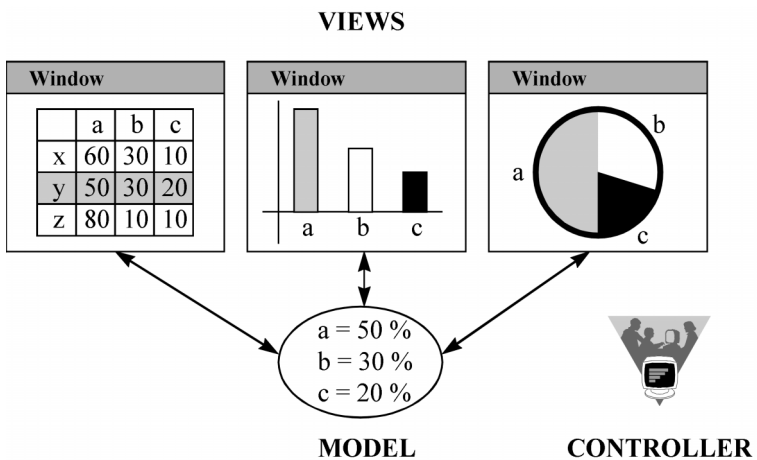
Beispiel

Mustergestützter GUI-Entwurf

Der Entwurf einer grafischen Benutzerschnittstelle (GUI) wäre ohne Erfahrung mühsam, ad hoc und fehleranfällig. Ein mustergestützter Entwurf könnte nun folgendermaßen verlaufen: Wir definieren zunächst unser Problem, zum Beispiel die Entkopplung unterschiedlicher, aber konsistenter Visualisierungen eines Datenbestands, und suchen dann in einem „Musterbuch" nach geeigneten Vorlagen. Wir finden ein *strategisches* Muster namens MODEL-VIEW-CONTROLLER (MVC, Bild 1) und erfahren, daß es sich in interaktiven Grafikanwendungen bewährt hat. Wenige Seiten beschreiben das immer wiederkehrende Entwurfsproblem, den Anwendungskontext und eine Lösungsstruktur.

Bild 1: Illustration zum Strategiemuster MVC

[Gamma et al., 1995, S. 293]



Musterform

Grafik und Text sind anschaulich und intuitiv; wir erfahren, wie die MVC-Dreiteilung[4] die ehernen Prinzipien der Softwaretechnik umsetzt: „schwache Kopplung zwischen und starke Kohäsion in den Modulen“ von Herbert A. Simon und „Information hiding“ von David Parnas [Simon, 1962; Parnas, 1972]. Wir werden auf feinkörnige *taktische* Muster verwiesen, sie haben dieselbe Ordnung: Musternamen und Synonyme, Einordnung, Zweck und Einsatzmotive, Anwendungs-Szenarien, allgemeine Lösungsstruktur, beteiligte Klassen und deren Zusammenspiel, Vor- und Nachteile, Beispiele, Nachweise über den erfolgreichen Praxiseinsatz, Querverweise und Abgrenzungen zu ähnlichen Mustern.

Objekt-Entkopplung: PUBLISHER-SUBSCRIBER

Ein solches Taktikmuster heißt PUBLISHER-SUBSCRIBER (Klassendiagramm in Bild 2). Es beschreibt die Korrelation zwischen dem zentralen Datenbestand (MODEL) und den verschiedenen Sichten (VIEWS). Das PUBLISHER-Objekt unterrichtet alle SUBSCRIBER-Objekte über Zustandsänderungen (NotifySubscribers()), indem es deren Aktualisierung veranlaßt (Update()). Dabei sind die SUBSCRIBER-Objekte untereinander *entkoppelt* und somit austauschbar – dynamisch zur Programmlaufzeit. Das Muster beschreibt also eine *1-zu-n*-Abhängigkeit zwischen Objekten. Auf diese Weise läßt sich zum Beispiel ein Datenbestand (in der PUBLISHER-Rolle) auf verteilten Rechnern oder in mehreren Fenstern eines Rechners unterschiedlich darstellen: als Tabelle oder Diagramm (SUBSCRIBER-Rollen).

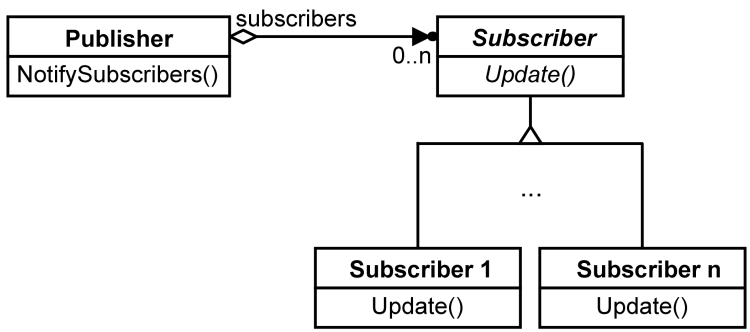
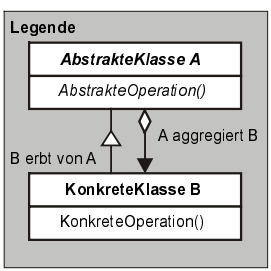


Bild 2: Lösungsstruktur des Taktikmusters PUBLISHER-SUBSCRIBER



4 Der CONTROLLER für die Abfrage und Steuerung der Benutzeraktionen ist für unser Beispiel nicht wichtig.

Muster und Frameworks

Viele Muster wie PUBLISHER-SUBSCRIBER stammen aus GUI-Frameworks: ET++, UniDraw und InterViews [Lewis (Hrsg.), 1995]; es waren erfahrene Framework-Entwickler [Gamma, Helm, Johnson, Vlissides, 1995], die die Musterform favorisierten. Muster zur Framework-Dokumentation legen das Erfahrungswissen der Entwickler offen und verringern so den Lern- und Einarbeitungsaufwand für den Framework-Anwender.

Softwaremuster sind allgemein kein akademisches Thema, vielmehr gehen sie aus der industriellen Praxis hervor: Immer mehr Softwarehäuser versuchen, ihren Erfahrungsfundus in hausinternen Musterbüchern zu dokumentieren. Publizierte Musterbücher unterscheiden sich hauptsächlich in ihrem Abstraktionsgrad; so gibt es Muster für die Systemanalyse [Fowler, 1997], für den Programmentwurf [Gamma et al., 1995] und für die Codierung [Coplien, 1991].

Das Spektrum der Entwurfsmuster erfordert eine Bestandsaufnahme in komprimierter Form:

ETHOS [5]

E wie „Economic“

Bekanntlich liegt das Potential der Objektorientierung in der Wiederverwendbarkeit: Die Kapselung von Daten und Operationen erlaubt die systematische Wiederverwendung von Struktur und Verhalten in *einer* Komponente, der Klasse. Entwurfsmuster erweitern nun die Wiederverwendung auf *kommunizierende Gruppen* von Klassen; Wiederverwendung findet im großen statt, das heißt auf einer architektonischen Ebene, ähnlich der eines Frameworks. Im Gegensatz dazu sind Entwurfsmuster aber abstrakt (kein Code!) und somit anwendungsunabhängig; sie sind deshalb gut geeignet, ein Framework zu dokumentieren.

T wie „Technical“

Die Form eines Softwaremusters wird noch diskutiert: Der alexandrinische Vierteiler *Problem-Kontext-Kräfte-Lösung* ist erzählend, von epischer Art; *Port-*

5 Zum Kürzel ETHOS (ein didaktisches Muster) mehr im Anhang zu Kapitel 2.

land-Muster halten diese Form weitgehendst ein; [6] *Gamma*-Muster sind schematischer, sie werden nach dreizehn Gliederungspunkten beschrieben. Im Hinblick auf die Granularität der Lösungsstruktur lassen sich zwei Musterkategorien unterscheiden: Sprachspezifische Codierungs-Muster stellen die kleinste Kategorie, Idiome aus syntaktischen Strukturen [Coplien, 1991]; Gamma-Muster dagegen sind grobkörnige Strukturen aus drei bis vier Klassen, sie unterstützen den Programmentwurf.

Das zweckmäßige Medium für Entwurfsmuster ist *Hypertext*, da ein Muster nicht isoliert auftritt, sondern stets assoziativ im Verbund, und erst das Navigieren im Informationsraum eine effiziente Verwendung erlaubt. Alexander spricht hier von „Mustersprachen" und betont damit die Vernetzung. Ihre Gutenberg-Form werden Musterbücher bald überwinden: proprietäre Entwurfsmuster, also solche, die firmenspezifisches Know-how widerspiegeln, werden in Intranetzen verfügbar gehalten, nichtproprietäre im Internet.

H wie „Human"

Muster sind lernpsychologisch Problem-Lösungs-Paare gleicher Struktur; sie unterstützen den Erwerb von Fertigkeiten. In der industriellen Schulung und der akademischen Lehre werden sie bereits *didaktisch* eingesetzt. [7] Motivationspsychologisch steigern Entwurfsmuster das Selbstwertgefühl des Programmierers: Orientiert er sich an den Mustern eines Experten, dokumentiert er damit die Qualität seines Programms und distanziert sich zugleich vom Hackertum.

O wie „Organizational"

Derzeit sind Bemühungen im Gange, die bewährten Prinzipien und Strategien der Projektführung und der Teamorganisation in Musterform zu beschreiben.[8] Das Wissen über gute Organisationsformen entbehrt bisher der pragmatischen Vermittlung, Organisationsmuster versprechen hier Abhilfe.

S wie „Social"

Entwurfsmuster schaffen ein gemeinsames *Vokabular;* es vereinfacht die Teamkommunikation und erleichtert so die Diskussion über komplexe Zusammenhänge. Originalton aus einem Projekt-Meeting: „Nehmen wir an dieser Stelle das PUBLISHER-SUBSCRIBER-Muster, um die Komponenten zu entkoppeln." Für Sprachkundige, die das Mustervokabular beherrschen, ist damit alles gesagt!

---

6 http://c2.com/ppr/

7 http://www-lifia.info.unlp.edu.ar/ppp/

8 http://www.bell-labs.com/people/cope/Patterns/Process/index.html

Gesamtbewertung

Erfahrungstransfer

Die Gattung Musterbuch kann auf eine lange Tradition verweisen; sie hat ihre Anfänge nicht erst bei Christopher Alexander. Für die Entwicklung des Handwerks und der Industrie im 19. Jahrhundert waren Musterbücher von grundlegender Bedeutung: sie stellten ein Kompendium praktischen Wissens dar. Die Berufserfahrung aus vergangenen Jahrhunderten schlug sich hier nieder.

Die heutigen Handwerke wandeln sich mit der Technik – so der Software-Entwurf. Er wurde schon immer als *Handwerkskunst* verstanden, im guten Sinne als Kunsthandwerk: „Computer Programming as an Art" [Knuth, 1974], im schlechten Sinne als Spezialistentum: kryptische Programme, die nur der „Künstler" warten kann. Mit Hilfe von Musterbüchern à la Erich Gamma et al. können wir fortan die Kunst des Softwarehandwerks beschreiben und vor allem *lehren* …

Kapitelvorschau

1 Symmetrie und Software: Die Suche nach Entwurfsmustern

Im ersten Entwurfsmuster-Seminar betreute ich eine Querschnittsbefragung zum Musterbegriff, befragt wurden die Dozenten aller Fachbereiche unserer Universität [Hombach, 1995, auf CD-ROM im Anhang]. Die Quintessenz der Antworten ist eindeutig: Muster sind entweder ein etabliertes oder ein aktuelles Thema in Forschung und Lehre. Damals rief die Siegener Hochschulzeitschrift „Diagonal" dazu auf, fachübergreifende Beiträge zum Symmetriebegriff einzureichen. Dies ermutigte mich zu einem ungewöhnlichen Einstieg: Ich schrieb ein Essay über „Symmetrie und Software"; mein Ziel ist es, die Suche nach Entwurfsmustern *interdisziplinär* zu motivieren.

Eine Parallele zum Essay zeigt sich in der Altersprosa von Christopher Alexander – dem Spiritus rector der Musterbewegung. In seinem mehrbändigen Werk „The Nature of Order“ [Alexander, 1999] beschreibt er den Weg jenseits der Muster: *Geometrie* und *Ästhetik* sind hier die Leitmotive; beide vereint der Symmetriebegriff.

2 Erfahrung vermitteln: eine neue Schreibkultur

Entwurfsmuster tradieren eine andere Qualität des Wissens: nicht das methodische oder faktische Wissen steht im Vordergrund – vielmehr soll das *Erfahrungswissen* der Experten übermittelt werden. Ich widme mich also fachdidaktischen Betrachtungen zum *Skill*-Erwerb, das heißt zu Transfer und Wiederverwendung von Erfahrungswissen im allgemeinen und in der Softwaretechnik im besonderen. Ich formuliere hier diese Qualität des Wissens; sie ist vornehmlich darauf ausgerichtet, die Zeitspanne zu verkürzen, bis ein Informatikabsolvent für die einstellende Firma produktiv wird.

Der *Stand der Kunst* softwaretechnischer Entwurfsmuster ist kaum systematisiert; das ist durchaus verständlich, denn die Musterbewegung in der Informatik nahm erst Anfang der 90er ihren Lauf (mit der Dissertation von Erich Gamma und dem ersten Architektur-Workshop auf der OOPSLA 91). Ich sortiere in diesem Kapitel die Literatur (meist Webseiten[9]) und führe eine Mustersystematik ein. Die Fachgemeinschaft, die sich der Erfahrungsvermittlung per Musterform verschrieben hat, pflegt eine neue Schreibkultur; wie sich diese äußert und wodurch sie sich von der traditionellen Schreibkultur unterscheidet, ist der zweite Gegenstand des Kapitels.

3 Entwerfen und Dokumentieren mit Mustern

Dienten die vorigen Kapitel dazu, die Gesamtperspektive auf die Musterbewegung zu erfassen und das Gesichtete zu sammeln und zu ordnen, so ist

9 http://www.cetus-links.org/oo_patterns.html

das letzte Kapitel einem aktuellen Forschungsthema gewidmet – dem mustergestützten Entwerfen und Dokumentieren. Mein Ansatz geht zurück auf die Arbeiten von Donald E. Knuth zu „Literate Programming“ und erweitert diese um die Belange des Entwerfens zum *Literate Designing*. Die Ergebnisse wurden in Industrieprojekten gewonnen und als Erfahrungsbericht der Fachgemeinschaft auf der ECOOP 97 zur Diskussion gestellt. Auf der EuroPLoP 98 organisierte ich einen Expertenworkshop zum Thema „Pattern-Aided Software Documentation“; über den hier eruierten Stand der Technik berichte ich im Anhang. Dort findet sich auch eine Projektskizze; sie beschreibt, wie die Forschungsergebnisse zum verzahnten Entwerfen und Dokumentieren überführt werden in eine Werkzeugumgebung mit kommerziellen Komponenten.

Anhänge

Die Musterbewegung ist reich an Facetten, ihre Ideen und Methoden finden sich in Forschung und Lehre unterschiedlichster Entwurfsdisziplinen. Um dies zu dokumentieren, habe ich einige Projekt- und Erfahrungsberichte teilweise im Original (Englisch) und mit eigenen Literaturverzeichnissen an die Kapitel angehängt; sie sind in sich geschlossen und verdeutlichen die Konzepte und Methoden des jeweiligen Kapitels.

CD

Sie ist eine geordnete Materialsammlung zu den Themen der einzelnen Kapitel. Die Musterbewegung versteht sich primär als eine Internet-Bewegung; mit Ausnahme der Musterkataloge findet sie ihren Niederschlag nur selten in gebundener Form. Eine Arbeit, die den aktuellen Technikstand und seine Tendenzen behandelt, kann sich derzeit im wesentlichen nur auf Webquellen beziehen. Da diese bekanntermaßen flüchtig sind, habe ich die wichtigsten Webseiten auf die Buch-CD kopiert; sie lassen sich einfach per Mausklick im *Compuskript*, der Online-Version des Buches, aufrufen. Weiterhin findet der Leser einiges Beispielmaterial in Form von HTML-, PDF- und WinHelp©-Dateien auf der CD.

http://www.springer.com/978-3-540-65825-2

Entwurfsmuster
Design Patterns in der objektorientierten Softwaretechnik
Quibeldey-Cirkel, K.
1999, XI, 195 S., Hardcover
ISBN: 978-3-540-65825-2