

# Design Patterns zur Dokumentation von Erfahrungswissen

Christian Kohls

Institut für Wissensmedien, Tübingen  
c.kohls@iwm-kmrc.de

**Abstract:** Design Patterns sind eine erprobte Möglichkeit, implizites Expertenwissen zu externalisieren. Gerade im didaktischen Umfeld wird dieser Ansatz zunehmend gewählt, um wiederkehrende Designentscheidungen zu dokumentieren und somit das damit verbundene Wissen anderen Personen auf eine leicht verständliche Weise zugänglich zu machen. Dieser Artikel beleuchtet die formalen und theoretischen Grundlagen von Design Patterns und gibt einen Überblick über die historische Entwicklung nebst Beispielen.

## 1 Einführung

Man stelle sich zwei Städte vor, die miteinander Handel treiben wollen, jedoch vollständig durch einen Fluss voneinander getrennt sind. Wie könnte man dieses Problem lösen? Man wird schnell an eine Brücke denken, mit der die beiden Städte verbunden werden könnten. Obwohl eine Brücke sicherlich kein triviales Artefakt ist, fällt einem diese Konstruktion sofort als Lösung ein. Wir haben Brücken vielfach als erprobte und passende Lösung zur Überwindung von Hindernissen erlebt. Treffen wir auf eine Problemstellung dieser Art, so kommt uns die Lösung sofort in den Sinn, ohne dass wir einen Problem-Lösen-Prozess anstoßen müssen. Doch wie schwierig wäre die Aufgabe für jemanden, der noch nie eine Brücke gesehen hat? Würde er auf Anhieb die optimale Form für eine Brücke finden? Die uns bekannte Gestalt der Brücke enthält das Lösungswissen implizit, d.h. eine Brücke wäre geeignet, um die beiden Städte miteinander zu verbinden. Wir können auf dieses Lösungswissen zurückgreifen, da wir Brücken schon tausendfach gesehen haben. Nicht in allen Bereichen können wir jedoch auf soviel Erfahrung bauen.

Häufig sind wir gefordert, neue Objekte oder Prozesse zu gestalten. Dabei fällt es einem Experten in der Regel leichter zu einer erfolgreichen Lösung zu gelangen, als einem Anfänger. Wer zum ersten Mal eine große Konferenz organisieren muss, wird viele Entscheidungen und Aufgaben als schwierige Probleme empfinden, während ein „alter Hase“ es zu vermeiden weiß, jedes Problem von Grund auf neu anzugehen. Stattdessen wird er auf Lösungen zurückgreifen, die sich in der Vergangenheit bei gleichen oder ähnlichen Problemstellungen bewährt haben.

Stößt ein Experte auf ein ihm bekanntes Problem, so kann er auf seine Erfahrung zurückgreifen und die passende Lösung wieder verwenden. Die invarianten Teile dieser Lösungsstruktur werden im Folgenden als Muster betrachtet. So wie alle Brücken musterhafte Gemeinsamkeiten aufweisen und doch jede Brücke anders aussieht, finden

sich für Problem-Lösungs-Paare Strukturen, die vom Einzelfall abstrahieren. Was den Experten vom Anfänger unterscheidet, ist die Kenntnis dieser Muster.

Dieses Erfahrungswissen liegt jedoch meist nur in impliziter Form vor. Der Experte kann auf dieses Wissen zwar zurückgreifen, ihm ist das Problem-Lösungs-Paar jedoch nicht explizit bewusst. Wir finden zwar die Brücke als passende Lösung zur Überwindung des Flusses, aber eben erst wenn wir vor dem konkreten Problem stehen. Zielsetzung des Entwurfsmuster-Ansatzes ist es, dieses implizite Wissen in eine explizite Darstellung zu überführen, externalisiert zu dokumentieren und für zukünftige Entwürfe zu nutzen.

Entwurfsmuster unterstützen dabei den Wissenstransfer und sind besonders für den Einsatz in der Lehre geeignet. Zudem besitzen sie eine wichtige Kommunikationsfunktion, da sie komplexen Lösungsstrukturen einen eindeutigen Namen geben – so wie man die Konstruktion, um ein Ufer mit dem anderen zu verbinden, einfach „Brücke“ nennen kann. Die explizite Darstellung des Musters verlangt zudem, dass nicht nur Problem und Lösung sondern auch Anforderungen und Kontext erfasst werden, z.B. sollte die Brücke nicht an der breitesten Stelle des Flusses gebaut werden. Auch sei überlegt, dass bei sehr breiten Flüssen der Fährbetrieb oder ein Tunnelbau Ziel führender sein kann und eine Brücke nicht immer die beste Lösung ist.

Im Folgenden sollen zunächst verschiedene Einsatzgebiete, in denen Entwurfsmuster zunehmend eingesetzt werden, beispielhaft betrachtet werden. Danach wird der formale Aufbau von Entwurfsmustern dargestellt. Erst dann folgen Überlegungen, in welchem theoretischen Rahmen sich dieser von Erfahrungs- und Praxiswissen geprägte Ansatz bewegt. Abschließend werden Gütekriterien zur Bewertung und Evaluierung von Entwurfsmustern vorgeschlagen.

## **2 Historische Entwicklung und Beispiele**

Der Pattern-Ansatz stammt ursprünglich aus der Architekturtheorie und soll Individuen dabei helfen, Anforderungen und Bedürfnisse für ihren eigenen Wohn- und Lebensraum zu identifizieren und kommunizieren [Al77]. Insgesamt werden 253 Muster zur Gestaltung von Städten, Gebäuden und Konstruktionen beschrieben. Beck & Cunningham [BC87] greifen den Ansatz auf und übertragen ihn auf den Bereich der objektorientierten Softwareentwicklung. Der Durchbruch gelingt 1995, als Gamma et al. mit „Design Patterns“ [Ga95] ein Standardwerk zu objektorientierter Softwareentwicklung publizieren und Cunningham [Cu95] das erste Wiki programmiert, um darin kooperativ Entwurfsmuster zu sammeln.

Weitere Disziplinen greifen den Ansatz auf, darunter vor allem die Gebiete Mensch-Maschine-Interaktion [VM02, Tid05] und Portal-Design [DLH04, Ma06]. Auch in der Didaktik gibt es mehrere Ansätze, Expertenwissen in Form von Mustern zu dokumentieren: Unterrichtsmethoden als Handlungsmuster [Ba06], E-Learning Design Patterns Repository [ND05], Pedagogical Pattern Project [PPP], Patterns zum Design

von Networked Learning [Go05], Patterns zur Dokumentation didaktischen Wissens an Hochschulen [VW04]. Im folgenden sollen einige Beispiele in stark verkürzter Form wieder gegeben werden.

In „Bus Stop“ beschreiben Alexander et al. [Al77] die Zielsetzung, dass Bushaltestellen leicht erkennbare und angenehme Orte sein sollten. Es soll sich um lebendige Plätze handeln, an denen sich die Menschen sicher und wohl fühlen. Um dieses Ziel zu erfüllen schlagen sie vor, dass Bushaltestellen mit weiteren Angeboten wie Zeitungskiosken, Cafés, Stadtkarten, Sitzgelegenheiten usw. ausgestattet sein sollten. Es wird also beschrieben, wie man eine Haltestation gestaltet, an der man gerne ein-, um- oder aussteigt und sich nicht langweilt oder fürchtet beim Warten auf den nächsten Bus.

Gamma et al. [Ga95] beschreiben mit „Singleton“ eine Lösung, wie man beim Entwurf von Softwareapplikationen sicherstellen kann, dass es von bestimmten Objekten nur ein einziges Exemplar geben darf. Ein Beispiel ist die Dateiverwaltung eines Betriebssystems; es darf zwar mehrere Laufwerke geben, aber die Verwaltung muss von einer zentralen Stelle geschehen, um Konflikte zu vermeiden. Weitere Beispiele sind Benutzereinstellungen, die überall im Programm gültig sein sollen oder die Verwaltung von Gruppen in Content Management Systemen.

Goodyear [Go95] beschreibt mit „Discussion Group“, wie man die Struktur einer online geführten Diskussion so gestaltet, dass sich für die Teilnehmer ein möglichst großer Lernerfolg einstellt. Als Teil der Lösung wird darauf hingewiesen, dass die Regeln und Zeitpläne explizit bekannt gegeben und die Struktur gleich zu Beginn etabliert werden sollte.

Kohls & Windbrake [KW06] dokumentieren Entwurfsmuster für interaktive Grafiken. In „Transport Objects“ werden die Rollen, Zustände und Interaktionen beschrieben, um visuelle Inhalte dynamisch an einen verschiebbaren Container zu heften.

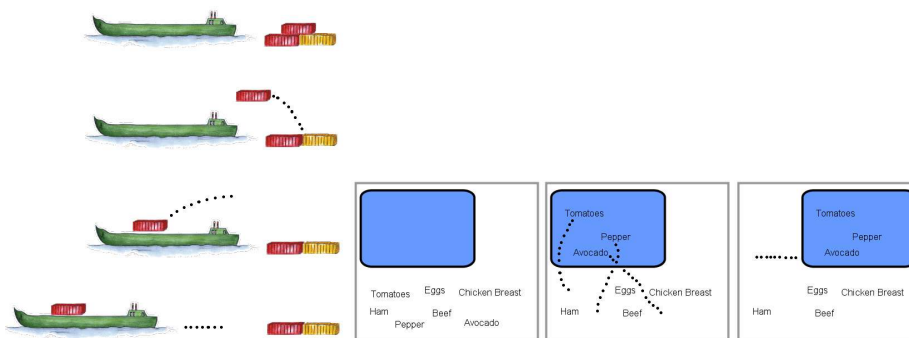


Abbildung 1: Illustrierte Beispiele zeigen das Muster in unterschiedlicher Verwendung

Auf diese Weise lassen sich Transport-Prozesse, aber auch dynamische Gruppierungen veranschaulichen. Als Lösung wird eine Container-Passagiere Beziehung vorgeschlagen. Passagiere bewegen sich gemeinsam mit dem Containerobjekt, wenn sie sich innerhalb des Containers befinden. Ansonsten bewegen sich Container und Passagiere unabhängig.

Auch wenn die beiden interaktiven Grafiken sehr unterschiedlich anmuten, so liegt ihnen doch das gleiche Interaktions-Prinzip zu Grunde.

### 3 Formale Definition eines Entwurfsmusters

Entwurfsmuster sind erprobte Lösungsansätze für wiederkehrende Fragestellungen und Aufgaben: „Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice“ [A177, S. x]. Es handelt sich also um generative Problem-Lösungs-Paare, die vom Einzelfall abstrahieren und nur jene invarianten Teile einer strukturierten Lösung berücksichtigen, die für alle Probleme (bzw. Aufgaben) einer Klasse relevant sind. Nach [A179] enthält ein dokumentiertes Entwurfsmuster in seiner Grundform Name, Problemstellung, Kontext, Forces (konkurrierende Anforderungen) und eine Lösung.

Diese Untergliederung in einzelne Felder hilft dabei, das interne, implizite Wissen in eine nachvollziehbare externe Form zu überführen. Auch nach [MD03] sind die Felder Pattern Name, Context, Problem, Forces und Solution zwingend. Sie sollen daher im folgenden kurz beschrieben werden.

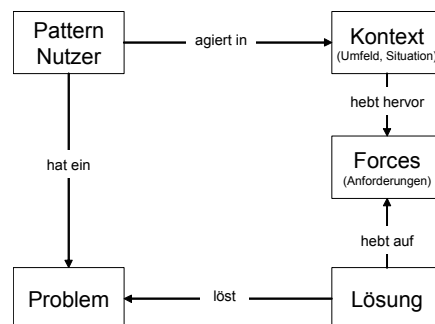


Abbildung 2: Zusammenspiel der Pattern-Bestandteile in Anlehnung an [MD03].

Der **Name** ist eine kurze Bezeichnung, mit dem auf das gesamte Entwurfsmuster Bezug genommen werden kann. Er spielt eine wichtige Rolle bei der Kommunikation zwischen Designern.

Unter **Problemstellung** wird ausgeführt, welche Frage oder welche Aufgabe sich mit einem Entwurfsmuster lösen lässt. Beispiele für solche Problemstellungen sind der Entwurf eines Webportals oder die Gestaltung einer Menüführung.

Der **Kontext** beschreibt in welchem Umfeld beziehungsweise in welcher Situation das Entwurfsmuster anwendbar ist. Zum gleichen Problem, etwa dem Entwurf eines Portals, kann es durchaus unterschiedlich gut geeignete Designstrategien geben. So gibt es zum Beispiel spezialisierte Entwurfsmuster für die Gestaltung von Bildungs-, Nachrichten-, oder Shopping-Portalen.

Der Grund für die Diversifikation liegt darin, dass je nach Umfeld unterschiedliche Anforderungen (**Forces**) erfüllt werden müssen. Der Kontext bestimmt damit im wesentlichen die Anforderungen und Ansprüche, die in einem konkreten Fall für die Lösung eines Problems zu bedenken sind. Design-Anforderungen stehen häufig in einem Konflikt zueinander, z.B. der Anspruch auf Funktionsvielfalt vs. einfache Bedienung oder die Verwendung hochwertiger Materialien vs. preiswerte Produktion, und bauen auf diese Weise ein Spannungsfeld auf.

Die **Lösung** soll dieses Spannungsfeld ausbalancieren. Sie beschreibt und diskutiert in generischer Form eine statische oder dynamische Struktur, die sich in der Praxis für Probleme der gleichen Klasse mehrfach bewährt hat. Fallspezifische Lösungsteile werden dabei klar von generalisierbaren Teilen unterschieden.

Häufig erhalten die Beschreibungsfelder andere Namen, die besser zu einem spezifischen Anwendungsgebiet passen. Zudem kann die Beschreibung eines Musters weiter ausdifferenziert werden, indem zusätzliche Felder eingeführt werden. Ein Beispiel ist das Feld „Related Patterns“, welches auf verwandte Muster hinweist. Dabei kann es sich um alternative Lösungen für dasselbe Problem, um speziellere bzw. verallgemeinerte Beschreibungen oder um Teilprobleme handeln.

Das Trennen und Vernetzen von Teilproblemen lässt aus einer Sammlung von einzelnen Entwurfsmustern eine Pattern Language entstehen, mit der sich Probleme und Aufgaben auf unterschiedlichem Detailniveau angehen und zerlegen lassen. Allerdings ist gerade die Trennung und Wahrnehmung voneinander unabhängiger Strukturen ein sehr schwieriger und subjektiver Prozess.

#### **4 Theoretische Überlegungen zu Entwurfsmustern**

Ogleich die Dokumentation von Expertenwissen in Form von Entwurfsmustern in zunehmender Zahl in wissenschaftlichen Publikationen beschrieben wird, fehlt bislang ein theoretischer Rahmen, der die Arbeitsweise von Mustern erklärt. An dieser Stelle können daher nur Hinweise gegeben werden, auf welchen theoretischen Überlegungen die Verwendung von Entwurfsmustern fundiert.

Zunächst sei darauf hingewiesen, dass sich Entwurfsmuster stets mit der Beschreibung wiederkehrender Problem-Lösungs-Paare für Designaufgaben beschäftigen. Diese Muster befassen sich also nicht mit natürlichen, sondern immer mit artifiziellen, d.h. von Menschenhand geformten und somit auch prinzipiell unterschiedlich gestaltbaren Erscheinungen. Diese Formgebung wird als kritischer Prozess aufgefasst, nach Alexander [Al64, S.15] als Design-Problem bei dem es darum geht, die am Besten passende Form unter Berücksichtigung des Kontexts zu finden: „It is based on the idea that every design problem begins with an effort to achieve fitness between two entities: the form in question and its context. The form is the solution to the problem; the context defines the problem.“. Design wird also als Problem-Lösen-Prozess aufgefasst.

Das Finden einer guten Form ist für die meisten Design-Aufgaben nicht trivial, da wir es in der realen Welt meist mit einem durch Komplexität gekennzeichneten Umfeld zu tun haben. Jede Design-Entscheidung führt zu unterschiedlichen Befriedigungsgraden für die einzelnen Anforderungen, z.B. kann die Wahl eines bestimmten Materials die Stabilität eines Objekts garantieren, aber die Produktionskosten erhöhen. Ebenso kann die Wahl eines bestimmten Unterrichtsmediums den Lerneffekt steigern, aber Schüler oder Dozenten über Gebühr beanspruchen. Eine passende Lösung zu finden wäre praktisch unmöglich, wenn jede nur erdenkliche Entwurfsvariable wechselseitige Auswirkungen auf den Passungsgrad einer gestalteten Form hätte. Dies ist aber nicht der Fall: Einige Entscheidungen haben gar keine Auswirkungen (z.B. ob ein Dozent im blauen oder weißen Hemd erscheint, dürfte den Lernprozess nicht beeinflussen), andere Designfragen können unabhängig voneinander betrachtet werden (z.B. die Wahl der Unterrichtsmedien mag unabhängig von späteren Testformen sein). Diese Dekomposition in voneinander unabhängige Design-Probleme ist Grundlage für das Finden von einander abgrenzbaren Entwurfsmustern. Alexander lehnt sich dabei gedanklich an die Arbeit von Simon [Si96] an, der grundlegende Untersuchungen zum Problem-Lösen durchgeführt hat und davon ausgeht, dass Systeme stets hierarchisch aufgebaut sind, woraus deren Komplexität resultiert.

Betrachtet man klar voneinander unterscheidbare Artefakte, so stellt man fest, dass bestimmte Strukturen immer wieder auftauchen: Türen, Fenster, Automobile, Vorlesungen, Seminare, Diskussionsgruppen, Wikis, Blogs usw. Dabei handelt es sich jeweils um eine Lösung für ganz unterschiedliche Probleme. Allerdings repräsentiert nicht jedes Exemplar einer Klasse stets eine gute Form: Eine Teekanne, die ständig tropft, den Tee kalt werden lässt, einen zu engen Griff hat oder wackelig auf dem Tisch steht, ist schlecht entworfen. Zwar mag diese Kanne als Behälter für Tee dienen, doch sie vernachlässigt viele Anforderungen, die ein Teetrinker an die Kanne stellen mag. Dies ist um so ärgerlicher, da wir aus Erfahrung wissen, dass es sehr wohl möglich ist, eine Kanne zu entwerfen, die all diesen Anforderungen gerecht wird. Weitere Beispiele für solche Design-Fauxpas finden sich bei Norman [No88], der sowohl von Alexander wie auch Simon inspiriert wurde. Die Aufgabe von Entwurfsmustern ist es, die invarianten Teile der guten Lösungen zu erfassen, insbesondere mit Hinblick auf deren Gebrauchstauglichkeit für den Menschen.

Ob es tatsächlich für alle Bereiche objektiv messbare Strukturen gibt, die wiederholt in der Welt auftauchen, ist eine Grundannahme, deren Beweis von der Pattern-Community bislang nicht erbracht wurde. Sie ist nur deshalb nahe liegend, da wir offenbar in unseren Köpfen solche Schemata bilden, d.h. wir können nicht nur ein Auto eindeutig von einem Baum unterscheiden, sondern z.B. auch einen Vortrag als Vorlesung oder Verkaufsveranstaltung einstufen. Dass sich derartige Wissensstrukturen aufbauen ist Gegenstand der Schema-Theorie [Sc94].

Der Spezialfall, dass wir eine Gestalt, d.h. Objekte oder Prozesse, nicht nur kennen, sondern als Lösung für ein gegebenes Problem identifizieren und abrufen, kann als Problem-Lösen-Schema im Sinne von [Va90] aufgefasst werden. Entwurfsmuster scheinen implizit als mentale Modelle bei Experten vorhanden zu sein [VM02, WV03].

In dieser Form haben Menschen schon immer auf Muster zurückgegriffen. Das neue am Entwurfsmuster-Ansatz ist jedoch, diese Muster in einer bestimmten Form aufzuschreiben und über einen Namen zu referenzieren [VI97]. Dabei muss das implizite Muster dem Autoren zunächst explizit bewusst und dann von ihm formal dokumentiert werden [GLC01].

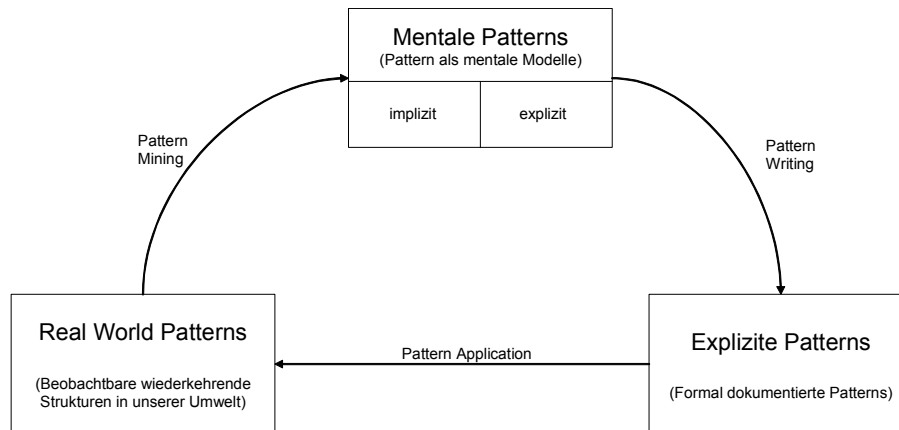


Abbildung 3: Zusammenhang zwischen den verschiedenen Pattern-Formen.

Der Prozess des Pattern-Minings, d.h. wie Experten Musterstrukturen aufbauen oder finden, ist nicht in der Tiefe untersucht. Hier eignet sich die Schema-Theorie als Erklärungsmodell.

Das Pattern-Writing ist dagegen stark instrumentalisiert, mit einer Pattern Language of Pattern Writing [MD03], einem iterativen Review-Verfahren durch einen zugeteilten Mentor (Shepherding-Prozess, siehe [Ha99]) und Writers' Workshops [Ga02].

## 5 Gütekriterien für Entwurfsmuster

Ob formale Entwurfsmuster von Designern in der Praxis berücksichtigt werden (Pattern Application) ist sehr unterschiedlich. Im Bereich der objektorientierten Programmierung und der Informatik allgemein sind Design Patterns sehr etabliert. Im Bereich der Architektur, wo der Pattern-Ansatz ursprünglich entstanden ist, konnte er sich jedoch nie in der Breite durchsetzen. Im folgenden sollen daher Kriterien diskutiert werden, die für den Erfolg einer Pattern Language verantwortlich sein könnten und auch Hinweise zur Qualitätssicherung bei Entwurfsmustern geben können.

**Validität:** Entscheidend für die Qualität eines Musters ist, ob es tatsächlich das vorgegebene Problem löst. Muster sind dann glaubwürdiger, wenn Nachweise

erprobter Beispiele aus der Praxis angeführt werden können oder zumindest hypothetische Einsatzszenarios bestehen, für die es empirische Belege gibt.

**Vollständigkeit:** Wenn ein Entwurfsmuster nicht alle Design-Anforderungen berücksichtigt, dann wird es in der Regel keinen vollständigen Lösungsansatz bieten können. Je mehr Lücken ein Muster enthält, desto weniger Nutzen stiftet es.

**Plausibilität und Verständlichkeit:** Ein Muster kann valide und vollständig beschrieben sein, doch wenn es nicht verständlich und überzeugend formuliert ist, wird es kaum Akzeptanz finden. Expertisegrad und Vorwissen der Zielgruppe sind daher zu berücksichtigen.

**Strukturierung:** Häufig werden die Beschreibungsfelder Problem, Kontext und Forces nicht richtig getrennt. Tatsächlich liegt hierin die Schwierigkeit beim Schreiben, denn diese drei Komponenten hängen sehr eng zusammen. Eine klare Trennung ist jedoch wichtig, da z.B. demselben Problem je nach Kontext mit unterschiedlichen Lösungen begegnet werden muss.

**Granularität:** Die richtige Abgrenzung und Vernetzung von Teilproblemen macht das einzelne Muster nicht nur übersichtlicher sondern steigert auch die Wiederverwendbarkeit, da Muster möglichst unabhängig voneinander sein sollten.

**Fuzzyness:** Nicht in allen Wissensdomänen lassen sich Strukturen gleich scharf voneinander trennen. Was für die formalen Strukturen des Software-Designs gut funktioniert, ist für human- und sozialwissenschaftliche Felder sehr viel schwieriger.

**Objektivität:** Weiter muss man zwischen objektiven und subjektiven Mustern unterscheiden, sobald der Lösungsansatz von persönlichen Einstellungen und Überzeugungen abhängt. So kann man zwar unterschiedliche Lehrmethoden als Muster erfassen, Akzeptanz und Gültigkeit sind jedoch mehr oder weniger von individuellen Ansichten abhängig. Der „Nürnberger Trichter“ würde heute wohl kaum noch als Muster akzeptiert werden.

**Relevanz und Expertise-Grad:** Aufwand und Nutzen der Muster-Dokumentation sind zu berücksichtigen. Zwar lässt sich auch ein Kugelschreiber formal als Entwurfsmuster beschreiben, doch dürfte das Interesse daran sehr gering sein – dieses Muster haben wir ohnehin im Kopf. Interessant sind Entwurfsmuster, die Expertenwissen zugänglich machen.

**Komplexität des Umfelds:** Je schwieriger es ist, wiederkehrende Strukturen aufzuspüren, desto wertvoller sind die gefundenen Muster. So helfen die Muster im Bereich des Software-Designs in komplexen Diagrammen sehr schnell einen Überblick zu erlangen. Im Gegensatz zu Türen oder Fenstern sind im Bereich der Softwarearchitektur wiederkehrende Elemente nämlich nicht mit dem bloßen Auge zu erkennen, so dass Hilfestellungen wie explizite Design Patterns unabkömmlich sind.

**Abstraktionsgrad:** Abstrakte Muster können vielseitiger eingesetzt werden, sind jedoch auch schwerer verständlich und liefern weniger Lösungsdetails, da spezifische Aspekte ausgelassen werden. Zu spezialisierte Muster erschweren dagegen die Übertragbarkeit auf andere Problemstellungen. Das oben erwähnte Muster „Bus Stop“ könnte daher kritisiert werden, da es zu spezifisch gewählt ist – der Lösungsansatz würde auch für Bahnhöfe und Flughäfen anwendbar sein.

## 5 Zusammenfassung

Entwurfsmuster sammeln Expertenwissen in einer formalen Beschreibungsstruktur und grenzen Teilprobleme eindeutig voneinander ab, um die Komplexität zu reduzieren. Die Externalisierung des Expertenwissens kann nicht nur beim Transfer und der Kommunikation helfen, sondern bietet auch eine Grundlage, um unterschiedliche Lösungsvorstellungen von Individuen abzugleichen und eine gemeinsam anerkannte Vorgehensweise abzustimmen. Zudem können sie zur Klassifikation herangezogen werden, da sie hierarchisch aufgebaut sind und wiederkehrenden Designs einen eindeutigen Namen geben.

Ein solides theoretisches Fundament für Entwurfsmuster gibt es bislang nicht. Man begnügt sich damit, dass Entwurfsmuster in der Praxis offensichtlich gut funktionieren. Allerdings ist dies nur für einige Pattern Languages der Fall. Deshalb ist es neben der Suche nach neuen Mustern auch erstrebenswert, die zugrunde liegenden Mechanismen und Erfolgsfaktoren dieses Ansatzes zu tiefer ergründen und besser zu verstehen. Nur so kann sichergestellt werden, dass die aufwändige Dokumentation von Entwurfsmustern nicht bloß zufällig den erhofften Einsatzerfolg in der Praxis erreicht.

## Literaturverzeichnis

- [AI64] Alexander, C.: Notes on the Synthesis of Form. Cambridge, MASS: Harvard University Press, 1964.
- [AI77] Alexander, C.; Ishikawa, S.; Silverstein, M.; Jakobson, M.; Fiksdahl-King, I.; Angel, S.: A Pattern Language; New York: Oxford University Press, 1977
- [AI79] Alexander, C: The Timeless Way of Building. New York: Oxford University Press, 1979
- [Ba06] Baumgarter, P.: Unterrichtsmethoden als Handlungsmuster - Vorarbeiten zu einer didaktischen Taxonomie für ELearning. In Mühlhäuser, M., Rößling, G., Steinmetz, R. (Hrsg.), DeLFI 2006, 4. e-Learning Fachtagung Informatik (S. 51-62). Darmstadt: Gesellschaft für Informatik e.V.
- [BC87] Beck, K. & Cunningham, W.: Using Pattern Languages for Object-Oriented Programs. In: Technical Report CR-87-43, Tektronix, Inc. OOPSLA'87 workshop on Specification and Design for Object-Oriented Programming; 1987.

- [BRP04] Baggetun, R., Rusman, E., & Poggi, C.: Design patterns for collaborative learning: from practice to theory and back. In Cantoni, L. & McLoughlin, C. (Eds.) Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications, Norfolk, USA, 2493-2498, 2004.
- [GLC01] Granlund, Asa; Lafrenière, Daniel ; Carr, Daniel A.: A Pattern-Supported Approach to the User Interface Design Process, Proceedings of HCI International 2001, 9th International Conference on Human-Computer Interaction, 2001, New Orleans
- [Cu95] Cunningham, W. Portland Pattern Repository's Wiki. Verfügbar unter: <http://c2.com/cgi/wiki?WelcomeVisitors> [12.2.2007]
- [DLH04] Duynie, Douglas K. van; Landay, James A.; Hong, Jason I.: The Design of Sites, Addison-Wesley, 2004
- [Ga95] Gamma, E.; Helm, R.; Jonson, R.; Vlissides, J: Design Patterns: Elements of Reusable Object-Oriented Software. Reading, Mass: Addison-Wesley, 1995
- [Ga02] Gabriel, P.: Writers' Workshops & the Work of Making Things. Patterns, Poetry... Boston: Pearson Education, 2002.
- [Go05] Goodyear, P. Educational design and networked learning: Patterns, pattern languages and design practice. Australasian Journal of Educational Technology 2005, 21 (1), 82-101.
- [Ha99] Harrison, N.B.: The Language of Shepherding. A Pattern Language for Shepherds and Sheep. <http://www.mcs.vuw.ac.nz/~kplop/Shp.html> [14.10.2006]
- [KW06] Kohls, C., Windbrake, T. Towards a Pattern Language for Interactive Information Graphics. Pattern Languages of Programming Design 2006. Portland, Oregon: Hillside Group. URL: [http://hillside.net/plop/2006/accepted\\_papers.htm](http://hillside.net/plop/2006/accepted_papers.htm).
- [Ma06] Mahemoff, M.: Ajax Design Patterns. Creating Web 2.0 Sites with Programming and Usability Patterns. Sebastopol: O'Reilly Media, 2006.
- [MD03] Meszaros, Gerard; Doble, Jim: A pattern language for pattern writing. <http://hillside.net/patterns/writing/patterns.htm>, (accessed: 18.10.05)
- [ND05] Niegemann, H. M.; Domagk, S: ELEN project Evaluation Report, Report of Work package 5. E-LEN project: a network of e-learning centres; [http://www2.tisip.no/E-LEN/documents/ELEN-Deliverables/Evaluation\\_Report\\_E\\_LEN.pdf](http://www2.tisip.no/E-LEN/documents/ELEN-Deliverables/Evaluation_Report_E_LEN.pdf) (accessed 29.03.06)
- [No88] Norman, D.: The Design of Everyday Things, New York: Basic Books, 1988.
- [PPP] The Pedagogical Patterns Project, <http://www.pedagogicalpatterns.org/>
- [Sc94] Schnotz, W.: Aufbau von Wissenstrukturen (S. 61-118). Weinheim: Beltz, 1994.
- [Si96] Simon, H.A.: The Sciences of the Artificial. MIT Press, 3rd edition, 1996.
- [Ti05] Tidwell, J.: Designing Interfaces, O'Reilly, Sebastopol, 2005.
- [V197] Vlissides, J.: Patterns: The Top Ten Misconceptions, Object Magazine, 1997
- [Va90] VanLehn, K. (1990). Problem Solving and Cognitive Skills Acquisition. In Posner, M.I. (Hrsg.) Foundations of Cognitive Science (S. 527-579). Cambridge: MIT Press.

- [VW04] Vogel, R; Wipperamnn S.: Dokumentation didaktischen Wissens in der Hochschule  
Didaktische Design Patterns als eine Form des Best-Practice-Sharing im Bereich von  
IKT in der Hochschullehre, Wissenschaftsforschung Jahrbuch 2004, Berlin. 2005
- [VM02] Veer, G. C. van der, Melguizo, M.C.: Mental Models. In J.A. Jacok & A. Sears (Hrsg.)  
The Human-Computer Interaction Handbook: Fundamentals, evolving Technologies and  
emerging applications. (S. 52-80). Lawrence Erlbaum & Associates, 2002.
- [WV03] Pattern Languages in Interaction Design: Structure and Organization: M. van Welie,  
G.C. van der Veer, In: Proceedings of Interact '03, 1-5 September, Zürich, Switzerland,  
Eds: Rauterberg, Menozzi, Wesson, p527-534, ISBN 1-58603-363-8, IOS Press,  
Amsterdam, The Netherlands, 2003.