Patterns in der Wirtschaftsinformatik

Im Software-Engineering sind Patterns sowohl aus Forschungs- wie auch aus Praxissicht nicht wegzudenken. In der Wirtschaftsinformatik ist dagegen nicht so ganz klar, wo und wie Patterns im System der etablierten Ergebnis-Artefakttypen Konstrukt, (Referenz-)Modell, Methode und (Informationssystem-)Instanz sowie im Konstruktions- und (Wieder-)Verwendungsprozess von Artefakten zu positionieren sind. Zwar haben Patterns gewisse Ähnlichkeiten mit Referenzmodellen; Sie entstehen aber offenbar auf andere Weise und werden auch in anderer Weise verwendet.

Diese Diskussionsrunde beschäftigt sich deshalb mit der Frage, warum Patterns als sehr erfolgreicher Ansatz des Software-Engineerings in der Wirtschaftsinformatik vergleichsweise wenig Beachtung/Anwendung finden. Die Beitragenden wurden gebeten, neben einer generellen Darstellung Ihrer Erfahrungen möglichst auch zu den folgenden Fragen Stellung zu nehmen:

- Warum sind Patterns im Software-Engineering so erfolgreich?
- Warum hat bislang kaum jemand die „Pattern-Mechanik“ in die Wirtschaftsinformatik übertragen?
- Liegt in der „richtigen“ Übertragung der „Pattern-Mechanik“ in die Wirtschaftsinformatik der Schlüssel zu einem ungenutzten Potenzial für das Design von Geschäftsmodellen, Prozessen etc.?
- Oder sind Referenzmodelle die Patterns der Geschäftsarchitektur? Was unterscheidet dann Referenzmodelle und Patterns?
- Sollte es neben Patterns, welche die Struktur des Ergebnisartefakts beschreiben, auch Patterns für den Ergebniserstellungsprozess geben?
- Gibt es erfolgreiche Beispiele für Patterns in der Wirtschaftsinformatik?

Da die Sichtweisen von Forschenden und Praktikern durchaus unterschiedlich sein können, wurden Personen zu Beiträgen eingeladen, welche diese verschiedenen Perspektiven vertreten. Meine Einladung zu dieser Diskussionsrunde haben die folgenden Herren angenommen (in alphabetischer Reihenfolge):

- Prof. Dr. Jan vom Brocke, Hilti-Lehrstuhl für Business Process Management, Institut für Wirtschaftsinformatik, Hochschule Liechtenstein,
- PD Dr. Peter Fettke und Prof. Dr. Peter Loos, Institut für Wirtschaftsinformatik am Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI), Saarbrücken,
- Dr. Stefan Junginger, Mitglied des Vorstandes, und Christoph Moser, Produktmanager, BOC AG, Wien
- Wolfgang Keller, Autor und Berater, objectarchitects, Gräfelfing,
- Prof. Dr. Florian Matthes und Alexander Ernst, Lehrstuhl I19, Fakultät für Informatik, Technische Universität München.

Kollege Matthes und Herr Ernst berichten von eigenen Erfahrungen bei der Identifikation und Beschreibung von Patterns in der Wirtschaftsinformatik und geben einen Überblick über die wesentlichen Pattern-Charakteristika.

Kollege vom Brocke plädiert dafür, die in der Pattern-Community oder in der Referenzmodellierung gemachten Erfahrungen in eine interdisziplinäre, auf die Wiederverwendung von Artefakten ausgerichtete Konstruktionslehre einfließen zu lassen.

Herr Dr. Junginger und Herr Moser wie auch Herr Keller analysieren die Eigenschaften von Patterns sowie die Kulturunterschiede zwischen Pattern-Community und Wirtschaftsinformatik aus der Perspektive der Pattern nutzenden Praxis. Sie zeigen auf, in welchen Punkten die Wirtschaftsinformatik von der Pattern-Community lernen könnte.

Die Kollegen Fettke und Loos schließlich schlagen – basierend auf einer Analyse der Entwicklungen in der Referenzmodelldiskussion wie auch der Pattern-Diskussion – vor, nicht mehr terminologisch zwischen Referenzmodellen und Patterns zu unterschieden. Sie schlagen fünf Merkmale vor, hinsichtlich derer sich Patterns und Referenzmodelle als verschiedene Ausprägungen eines ähnlichen Grundkonzepts verstehen lassen.

Die Beiträge machen deutlich, dass einerseits signifikante Ähnlichkeiten zwischen den Artefakttypen Pattern und Referenzmodell im Software-Engineering und der Wirtschaftsinformatik gesehen werden. Andererseits lassen sich insbesondere die Patterns nicht allein auf die jeweiligen Artefakte reduzieren. Der Erfolg von Patterns ist vielmehr geprägt durch ein lebendiges Ökosystem der Pattern-Community. Writer-Workshops, Review-Kultur und gelebte Wiederverwendung sind wesentliche Elemente dieses Phänomens, das sich so im Bereich der Referenzmodelle nicht oder erst in Ansätzen findet.

Wenn auch Sie zu diesem Thema oder einem Artikel der Zeitschrift Wirtschaftsinformatik Stellung nehmen möchten, dann senden Sie Ihre Stellungnahme (max. 2 Seiten) bitte an den Hauptherausgeber der WIRTSCHAFTSINFORMATIK, Prof. Dr. Hans Ulrich Buhl, Universität Augsburg, Hans-Ulrich.Buhl@wiwi.uni-augsburg.de.

Prof. Dr. Robert Winter

Institut für Wirtschaftsinformatik

Universität St. Gallen

Wissen zur Gestaltung komplexer Systeme als Muster erfassen, strukturieren und weitergeben

In den vergangenen fünfzehn Jahren haben sich Muster (patterns) im Software-Engineering weltweit als ein erfolgreiches Hilfsmittel etabliert, um Wissen zur Gestaltung komplexer Systeme zu erfassen, zu strukturieren und weiterzugeben (Buschmann et al. 2007). Muster dokumentieren Lösungen zu wiederkehrenden Problemstellungen in einem gegebenen Kontext, die sich in der Praxis bewährt haben. Die Lösung wird dabei durch Strukturen, Beziehungen, In-

teraktionen und Prinzipien und weniger durch Algorithmen oder detaillierte Prozeduren beschrieben.

Sogenannte Anti-Muster (anti patterns) dokumentieren in der Praxis häufig auftretende schlechte Lösungen für gegebene Gestaltungsaufgaben. Sie entstehen zum Beispiel mangels Erfahrung, aufgrund fehlender Qualifikation und Weitsicht, oder durch schlechte Kommunikation und Zusammenarbeit in Teams.

Muster überall?

Schrittweise hat sich der Gegenstandsbereich von Mustern erweitert: Entwurfsmuster (Gamma et al. 1995) leisten Unterstützung beim Entwurf einzelner objektorientierter Softwarekomponenten, Architekturmuster (Buschmann et al. 1997) bei der Komposition von Softwarekomponenten zu Anwendungen, Architekturmuster für Unternehmensanwendungen (Fowler 2002) schließlich erfassen Gestaltungsprinzipien für die Vernetzung von Anwendungen zu IT-Landschaften.

Neuere Arbeiten verwenden Organisationsmuster, um soziale Prozesse und Strukturen im Umfeld der IT zu beschreiben und zu gestalten. Das als Muster gesammelte Wissen bezieht sich dabei nicht nur auf die jeweils relevanten Artefakte (Dokumente, Diagramme, Modelle, technischen Systeme), sondern auch auf Rollen, soziale Strukturen, Aktivitäten und Prozesse.

Coplien und Harrison (2004) etwa konzentrieren sich auf Organisations- und Verhaltensmuster in der agilen Softwareentwicklung, Manns und Rising (2004) untersuchen förderliche Muster für das Change-Management. DeMarco et al. (2008) charakterisieren zahlreiche konstruktive und destruktive Verhaltensmuster in Projekten und Teams und bieten für diese systematische Handlungsanweisungen, um Konflikte zu vermeiden und die Teamproduktivität (insbesondere in IT-Projekten) zu steigern.

Mit ähnlicher Zielsetzung entwickeln Buckl et al. seit 2007 mit Praxispartnern einen öffentlich als Wiki zugänglichen Katalog von Mustern zum Management der Unternehmensarchitektur (sebis EAM Pattern Catalog, Ernst 2008). Er umfasst nicht Unternehmensarchitekturmuster, sondern Muster, die bei der Erstellung eines unternehmensspezifischen Ansatzes zum Management der Unternehmensarchitektur orientiert an der individuellen Situation eines konkreten Unternehmens hilfreich sind.

Hierzu differenziert der Musterkatalog 55 verschiedene mögliche Gestaltungsaufgaben (concerns) im EAM und identifiziert für diese Aufgaben in der Praxis bewährte Vorgehensmuster (methodology patterns). Diese beschreiben Rollen und Organisationsstrukturen (z. B. Gremien, Projektteams) und ihre Verantwortlichkeiten und Interaktionen. Präzisiert werden diese

durch Verweise auf die dabei jeweils von den Akteuren verwendeten Diagramm- und Dokumenttypen. Diese sind zielgruppenspezifische Sichten auf die Unternehmensarchitektur und werden im Katalog als viewpoint patterns gesammelt und beschrieben. Für jede Sicht dokumentiert der Katalog per Verweis die benötigten Architekturinformationen (information model patterns), dies sind UML-Klassendiagrammfragmente ähnlich den klassischen Entwurfsmustern.
Ein Beispiel für ein im Katalog dokumentiertes EAM-Anti-Muster ist die fehlende Fokussierung auf wenige unternehmensspezifisch zentrale Gestaltungsaufgaben bei der Etablierung des EAM. In der Folge wird ein zu detailreiches Architekturmodell konzipiert und es gelingt nicht, die benötigten zahlreichen Architekturinformationen mit ausreichender Qualität zu sammeln und zu pflegen. Letztendlich scheitern solche Projekte früh, da der erwartete Nutzen der Modellierung aufgrund der mangelnden Qualität der Modelle nicht nachgewiesen werden kann.

Charakteristika musterbasierter Ansätze

Etwas holzschnittartig lassen sich die Charakteristika musterbasierter Ansätze wie folgt zusammenfassen:
Muster sind

- konstruktiv und kompositionell Ein Muster leistet einen abgegrenzten Beitrag zur Lösung eines überschaubaren Aspekts eines Gesamtproblems. Ein Systementwurf nutzt typischerweise diverse Muster.
- situativ Treibende Kräfte und Randbedingungen für die Musteranwendung werden diskutiert, positive wie negative Konsequenzen der Musteranwendung aufgezeigt.
- in natürlicher Sprache formuliert Besonders wichtig ist der Name des Musters, der das Fachvokabular der Nutzer bereichern soll. Beispiele, Abbildungen, Diagramme, Querverweise und Zitate werden gefördert. Eine Einengung durch Metamodelle und formale Sprachen findet nicht statt.
- gering formalisiert Die Auswahl eines passenden Musters und die Anwendung eines Musters im Entwurf erfordert menschliche Intelligenz und Kreativität, sie lässt sich nur in den seltensten Fällen auf ein einfaches „Customizing“ durch Konfiguration, Variantenselektion, Instanziierung und Parametrisierung einer generischen Vorlage zurückführen.

- praxisorientiert Muster werden durch die sorgfältige Studie der Praxis gewonnen (pattern mining). Die Praxisrelevanz soll durch (drei oder mehr) konkrete und nachprüfbare Anwendungsbeispiele dokumentiert sein. Muster sollen für Novizen verständlich und für Experten interessant formuliert sein, was sie auch für die Ausbildung attraktiv macht. Sie sollen auch durch Praktiker begutachtet werden.
- inkrementell nutzbar Die Anwendung eines Musters wird häufig als die optimierende Transformation eines bestehenden System(entwurf)s beschrieben. Dies erlaubt ein evolutionäres Vorgehen in dynamischen Umgebungen und bei unvollständiger Information (fix problem and observe effects).
- vernetzt Manche Muster sind nur im Kontext anderer Musters sinnvoll anwendbar; es gibt zusammengesetzte Muster und einander ausschließende Muster, etc. Diese Beziehungen werden in Musterbüchern und Musterkatalogen ausführlich diskutiert und durch Diagramme und Querverweise erläutert.
- diskursförderlich Die genannten Charakteristika von Mustern vereinfachen die Zusammenarbeit zwischen Muster-Autor und Muster-Nutzern über Artefakte geringer Komplexität und mit hoher Kohärenz. Der typische Lebenszyklus eines Musters beginnt mit der Entdeckung, gefolgt von Dokumentation, Peer-Review, Publikation und öffentlichem Feedback, jeweils mit inkrementeller Verbesserung der Dokumentation. Über die Jahre hat die Pattern-Community hierfür sehr effektive und konstruktive Techniken der Zusammenarbeit entwickelt. Hierzu gehören Wikis, Autorenworkshops, Autorencoaching (shepherding), und Pattern-Konferenzen mit speziellen Reviewrichtlinien und die Forderung, Musterbeschreibungen offen zugänglich zu publizieren.

Ein nahe liegendes Einsatzgebiet für Muster ist das Informationsmanagement. Hier stehen sie in Konkurrenz zu etablierten Methoden der Wirtschaftsinformatik, insbesondere der Referenzmodellierung und dem Business Engineering und es ergeben sich spannende Forschungsfragen, wie dort eine gegenseitige Befruchtung aussehen könnte, beispielsweise:

- Sind Referenzmodellkataloge eine geeignete Quelle für das Pattern Mining? (vgl. Fettke 2009)
- Kann die von Referenzmodellen angestrebte Wiederverwendung von Konstruktionsergebnissen auch mit musterbasierten Entwurfsmethoden erreicht werden?
- Welchen Nutzen können Modelle und Metamodelle beim musterbasierten Entwurf stiften?

- Was können Initiativen zum offenen Austausch und zur unternehmensübergreifenden Entwicklung von Modellen (open models) in der Wirtschaftsinformatik von der Pattern-Community lernen?
- Was ist das Gegenstück zu Anti-Mustern im Informationsmanagement?

Prof. Dr. Florian Matthes
Alexander Ernst
Fakultät für Informatik
Technische Universität München

Literatur

Buckl S, Ernst A, Lankes J, Schneider K, Schweda C (2007) A pattern based approach for constructing enterprise architecture management information models. In: 8. Internationale Tagung Wirtschaftsinformatik, Karlsruhe

Buschmann F, Henney K, Schmidt D (2007) Pattern oriented software architecture, volume 5: On patterns and pattern languages. Wiley

Buschmann F, Meunier R, Rohnert H, Sommmerlad P (1997) A system of patterns: Pattern-oriented software architecture. Wiley

Coplien J, Harrison N (2004) Organizational patterns of agile software development. Prentice Hall PTR

DeMarco T, Hruschka P, Lister T, Robertson S, Robertson J, McMenamin S (2008) Adrenaline junkies and template zombies: Understanding patterns of project behavior. Dorset House

Ernst A (2008) Enterprise architecture management patterns. In: Pattern languages of programs conference (PLoP08), Nashville

Fettke P (2009) Reference model catalog. http://rmk.iwi.uni-sb.de/index.php. Abruf am 2009-07-28

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns. Elements of reusable object-oriented software. Addison-Wesley

Manns M, Rising L (2004) Fearless change: Patterns for introducing new ideas. Addison-Wesley

Interdisziplinäre Konstruktionslehre

Als Beitrag zu dieser Diskussion möchte ich vorschlagen, dass wir uns von einzelnen Begriffsdiskussionen lösen und vielmehr in Richtung einer interdisziplinären Konstruktionslehre denken. Ist es tatsächlich so, dass „Patterns“ in der Wirtschaftsinformatik bisher kaum ver-

breitet sind? Werden sie dort vielleicht nur anders genannt? Herrschen möglicherweise auch spezifische Anforderungen an Konstruktionen in der Wirtschaftsinformatik, die andersartige Konzepte erfordern? Eine interdisziplinäre Konstruktionslehre böte die Chance, den Prozess des Konstruierens von Artefakten besser zu verstehen und die Ergebnisse einer Vielzahl an Disziplinen zugutekommen zu lassen. Dies soll nun genauer ausgeführt werden:

Zunächst erscheint es sinnvoll, auf die Intention von Patterns zurückzugehen, um die Diskussion von einzelnen Erscheinungsformen dieses Ansatzes im Software-Engineering zu lösen: Patterns stammen bekanntlich aus der Stadt- und Raumplanung (Alexander et al. 1977). Die Intention bestand darin, bewährte Problemlösungen derart zu beschreiben, dass sie in mehreren Kontexten wiederverwendet werden können. Bereits hier hatten sich gewisse Schablonen zur Beschreibung von Patterns etabliert, wie etwa die Benennung des Namens, des Problems, des Kontextes und der Lösung (Alexander 1979, S. 247). Dieser Ansatz ist auf das Software-Engineering übertragen und anhand verschiedener Patternarten konkretisiert worden (Coad 1992; Gamma et al. 1996; Fowler 1996).

In der Wirtschaftsinformatik ist die Übertragung der Pattern-Idee nicht in gleicher Weise zu beobachten. Doch lässt sich daraus folgern, dass es Patterns hier nicht gibt? Denken wir z. B. an Modelle wie ITIL (IT Infrastructure Library) oder SCORM (Supply Chain Operation Reference-Model). Diese Modelle werden explizit mit der Intention der Wiederverwendung bewährter Problemlösungen entwickelt und finden durchaus weltweit Beachtung. Auch viele „kleinere“ Modelle können hier genannt werden, wie etwa Konten- und Belegstrukturen (Scheer 1994), die im Sinne der Wiederverwendung durchaus als „Patterns“ fungieren – wenngleich sie auch nicht explizit als solche entwickelt worden sind. Selbst Fragen der Konstruktion von „Patterns“ haben lange Tradition in der Wirtschaftsinformatik, wie z. B. die Forschung zur Referenzmodellierung zeigt (Becker et al. 2007). Dort werden spezifische Methoden entwickelt, um vor allem Informationsmodelle (speziell: Prozess- und Datenmodelle) in unterschiedlichen Kontexten wiederzuverwenden (vom Brocke 2007).

Aber nicht nur Modelle, als Ergebnisse von Konstruktionen, werden in der Wirtschaftsinformatik wiederverwendet. Auch Wissen über den Prozess des Konstruierens ist Gegenstand der Betrachtung. Hier ist das Gebiet des Methoden-Engineerings zu nennen (Brinkkemper 1996). Speziell beim situativen Methoden-Engineering werden Fragen der Wiederverwendung methodischer Kernelemente sowie deren Adaption in unterschiedlichen Anwendungskontexten untersucht (Bucher und Winter 2009). Ebenso wie Referenzmodelle könnten auch solche „Referenzmethoden“ als Patterns betrachtet werden, in denen – in diesem Fall – Regeln zur Errei-

chung spezifischer Zielsetzungen betrachtet und anhand von Aktivitäten, Ergebnisdokumenten und Rollen beschrieben werden.

Insgesamt wird deutlich, wie nahe die einzelnen Ansätze beieinander liegen. Im Kern geht es um die Wiederverwendung von Artefakten im Konstruktionsprozess. Letztlich könnte gefolgert werden, dass geradezu jedes Artefakt, das wir im Rahmen eines gestaltungsorientierten Forschungsprozesses konstruieren (Winter 2008), durchaus als Pattern begriffen werden könnte. Schließlich handelt es sich um generische Artefakte, die gerade über den Einzelfall hinaus in unterschiedlichen Kontexten wiederverwendet werden sollen. Dies würde im Übrigen auch die Terminologiekonstruktion mit in die Diskussion einschließen, in der tatsächlich ganz ähnliche Fragestellungen behandelt werden, wie dies etwa die umfangreiche Forschung zu Ontologien zeigt.

Selbstverständlich kann diskutiert werden, ob in all diesen Fällen stets von „Patterns“ gesprochen werden sollte oder nicht. Wichtiger erscheint es aber, das zugrundeliegende Phänomen der Wiederverwendung besser zu verstehen, um Synergien zwischen den Arbeiten der unterschiedlichen Disziplinen zu nutzen. Zumindest drei Gestaltungsaspekte wären dabei zu berücksichtigen:

- Methodische Aspekte: Nach welchen Regeln wird die Wiederverwendung durchgeführt? Während Patterns typischerweise per Analogieschluss auf den Einzelfall übertragen werden (Fowler 1996, S. 8), stand in der Referenzmodellierung lange Zeit die Konfiguration von Modellvarianten im Fokus (Becker et al. 2007). Mittlerweile existiert ein umfassenderer Methodenbaukasten, der u. a. auch eine Instanziierung, Aggregation und Spezialisierung vorsieht (vom Brocke 2007). Hinzu kommen Fragen der Spezifikation des situativen Kontexts (Bucher und Winter 2009). Welche Attribute, aber auch welches Vokabular, sollen verwendet werden, um den Anwendungsbereich eines Artefakts zu beschreiben? Dies ist ein zentrales Problem, das sich auch bei der Wiederverwendung von Komponenten und Services zeigt.
- Organisatorische Aspekte: Wie sind die institutionellen Austauschverhältnisse der beteiligten Akteure? Anhaltspunkte liefert hier z. B. die Transaktionskostentheorie (Coase 1937). Oft wird implizit von einem marktlichen, oder auch Open-Source-basierten, Austausch von Artefakten ausgegangen. Der überwiegende Teil der Patterns in der Wirtschaftsinformatik wird aber womöglich unternehmensintern verwendet, etwa in Form so genannter Blueprints größerer Unternehmen oder Beratungsgesellschaften. Die Ursache kann darin liegen, dass diesen Artefakten eine relativ hohe „Spezifität“ oder auch „strategische Bedeutung“ beigemessen wird (vom Brocke und Buddendick

2004). Viele Initiativen scheitern, da solche grundlegenden organisatorischen Aspekte der Wiederverwendung außer Acht gelassen werden.

- Technische Aspekte: Mit welchen Werkzeugen kann der Prozess der Wiederverwendung unterstützt werden? Noch immer ist es so, dass Konstruktionstechniken zur Wiederverwendung von den meisten Modellierungs- und Case-Werkzeugen nur rudimentär unterstützt werden. Eine umfassendere Werkzeugunterstützung wäre aber nötigt, und zwar möglichst vertikal integriert auf allen Ebenen des Entwicklungsprozesses, um Wiederverwendung auch praxistauglich zu machen. Zudem sind Kollaborationsfunktionen erforderlich, um eine evolutionäre Weiterentwicklung von Artefakten zu ermöglichen (vom Brocke 2004). Auch hier stellen sich zu einem großen Teil generelle Fragestellungen.

Die Beispiele zeigen, welche Fragen sich generell mit der Wiederverwendung von Artefakten im Software-Engineering und in der Wirtschaftsinformatik verbinden. Ziel einer interdisziplinären Konstruktionslehre ist es, diese nicht weiterhin fallweise (und für verschiedene Konzepte erneut), sondern stellvertretend für Konstruktionsprozesse im Allgemeinen zu untersuchen.

Neben dem am Beispiel der Patterns diskutierten Aspekt der Wiederverwendung stehen weitere wichtige Themen an, wie etwa die Nachhaltigkeit von Konstruktionsergebnissen. Es erscheint daher vielversprechend, diese Themen im Schulterschluss mit anderen Disziplinen zu diskutieren. Software-Engineering und Wirtschaftsinformatik könnten hierzu einen Anfang machen. Weitere Disziplinen, wie etwa die Architektur, die Ingenieurwissenschaften oder auch die bildenden Künste, können sicherlich weitere spannende Anregungen liefern.

Prof. Dr. Jan vom Brocke
Hilti Lehrstuhl für Business Process Management
Institut für Wirtschaftsinformatik
Hochschule Liechtenstein

Literatur

Alexander C (1979) The timeless way of building. Oxford University Press, New York

Alexander C, Ishikawa S, Silverstein M (1977) A pattern language: Towns, buildings, construction. Oxford University Press, New York

Becker J, Delfmann P, Knackstedt R (2007) Adaptive reference modeling: Integrating configurative and generic adaptation techniques for information models. In: Becker J, Delf-

mann P (Hrsg) Reference modeling. Efficient information systems design through reuse of information models. Physica, Heidelberg, S 23–49

Brinkkemper S (1996) Method engineering: Engineering of information systems development methods and tools. Information and Software Technology 38(4):275–280

Bucher T, Winter R (2009) A taxonomy of business process management approaches. In: vom Brocke J, Rosemann M (Hrsg) Handbook on business process management. Springer, Heidelberg (im Druck)

Coad P (1992) Object oriented patterns. Communications of the ACM 35(9):125–159

Coase RH (1937) The nature of the firm. Economica 4(11):386–405

Fowler M (1996) Analysis patterns: Reusable object models. Addison-Wesley Professional

Gamma E, Helm R, Johnson R (1995) Design patterns. Elements of reusable object-oriented software. Addison-Wesley Longman, Amsterdam

Scheer AW (1994) Business process engineering: Reference models for industrial enterprises. Springer, New York

vom Brocke J (2004) Internetgestützte Referenzmodellierung. State-of-the-Art und Entwicklungsperspektiven. WIRTSCHAFTSINFORMATIK 46(5):390–404

vom Brocke J (2007) Design principles for reference modelling. Reusing information models by means of aggregation, specialisation, instantiation, and analogy. In: Fettke P, Loos P (Hrsg) Reference modelling for business systems analysis. Idea Group Publishing, Hershey, S 47–75

vom Brocke J, Buddendick C (2004) Organisationsformen in der Referenzmodellierung, Forschungsbedarf und Gestaltungsempfehlungen auf Basis der Transaktionskostentheorie. WIRTSCHAFTSINFORMATIK 46(5):341–352

Patterns sind mehr als nur Lösungen zu einem Problem in einem Kontext

Mit Design-Patterns verbinden viele Leser der Wirtschaftsinformatik das, was Sie in dem Buch „Design Patterns: Elements of Reusable Object-Oriented Software“ (Gamma et al. 1995) finden. Lösungen für technische Probleme der objektorientierten Programmierung, die in einem bestimmten Kontext bevorzugt anwendbar sind. Dass Patterns ursprünglich von dem (Bau-) Architekten Christopher Alexander in größerem Stil geschrieben wurden und um 1987 herum über Kent Beck und Ward Cunningham den Sprung in die Community der objektorientierten Softwareentwickler geschafft haben, sei hier der Vollständigkeit halber noch erwähnt. Wenn Sie sich für die Geschichte interessieren, finden Sie zum Beispiel in folgenden Quellen (Portland Pattern Repository oJ, Coplien 1996) jeweils einen guten Überblick.

Inzwischen gibt es Patterns in sehr vielen Bereichen der Softwareentwicklung und auch in anderen Disziplinen. Die Frage hier war, warum nicht auch Wirtschaftsinformatiker Patterns in größerem Stil benutzen. Dazu kann man sich allerdings zunächst fragen, worin der Nutzen der Verwendung von Patterns liegt.

Patterns selbst schreiben

Vordergründig sollte man meinen, der Nutzen stammt daher, dass man es als Softwaredesigner (oder Designer in einem anderen Bereich, in dem es Patterns gibt) leichter hat, gute Lösungen zu bauen, wenn man die Lösungen anderer, erfahrener Designer wiederverwenden kann. Das (und nicht mehr) ist der Nutzen, den die meisten Anwender von Patterns sehen. Man kann jedoch noch deutlich mehr Nutzen aus dem Umgang mit Patterns ziehen. Der stellt sich vor allem dann ein, wenn man selbst Patterns schreibt. Verschiedene Pattern-Formen und die Anleitungen erfahrener Pattern-Autoren motivieren dazu, sich damit auseinanderzusetzen, warum eine Lösung gut ist. Die konfliktären Kräfte, die eine Lösung bestimmen, werden explizit in Form von so genannten Forces sichtbar gemacht. Wer sich hiermit beschäftigt, lernt, seine Designentscheidungen sauber zu begründen und das Umfeld einer zukünftigen Lösung zunächst sauber zu analysieren, bevor es zu der berühmten Lösung kommt, für die man leider das Problem nicht nennen kann.

Produktive Reviews

Weiterer Nutzen ergibt sich über die Pattern-Konferenzen, bei denen Autoren ihre Patterns von einer Community begutachten lassen und nach mehreren Runden von Feedback in einem so genannten Writers‘ Workshop besprechen lassen. Wenn Sie sich hier für das WIE interessieren, sei wieder auf Coplien (1996) verwiesen. Abgesehen davon, dass man durch den Prozess vor dem Workshop, das so genannte Shepherding, meist gute Anregungen zur Verbesserung seiner Entwürfe bekommt, lernt man auch einen sehr produktiven Reviewprozess kennen, der darauf beruht, dass jeder Feedbackgeber durch Regeln und Rituale dazu gezwungen wird, ausschließlich Anmerkungen zu machen, die für den Pattern-Autor nützlich sind.
Es kann hier also als Zwischenbilanz festgehalten werden: Die Beschäftigung mit Patterns – am besten aktiv schreibend – ist für Praktiker von mehrfach hohem Wert: Gute Lösungen werden schneller gefunden und besser begründet. Reviews werden extrem produktiv.

Patterns und Academia

Darüber, warum sich Patterns in der Wirtschaftsinformatik im speziellen und in der Academia im Allgemeinen nicht breit durchsetzen, kann der Autor nur spekulieren. Tatsache ist jeden-

falls, dass man völlig unterschiedliche Ziele verfolgt, wenn man eine Menge von Patterns oder eine Dissertation schreibt.

Patterns sind nicht zur Originalität verpflichtet. Im Gegenteil. Es gibt hier das Prinzip der „3 independent known uses“ – etwas ist nur dann ein Pattern, wenn man nachweisen konnte, dass eine Lösung an mindestens drei unterschiedlichen Stellen eingesetzt wurde – und das am besten in einem produktiven Kontext, zum Beispiel in der Wirtschaft.

Dissertationen, die darauf beruhen, dass das Konzept bereits dreimal beschrieben wurde, kann es nicht geben. Die Forschungskultur lebt von Originalität und neuen Lösungen – egal, ob es dafür ein kommerzielles Problem gibt oder nicht. Ob dies wirtschaftlich sinnvoll ist, steht hier nicht zur Diskussion. Es ist auf jeden Fall eine Kultur, die tief im genetischen Code des akademischen Betriebs verwurzelt ist.

Will heißen: Patterns bringen einen Universitätsassistenten auf seinem Weg zur Dissertation nicht wirklich weiter. Man kann dann Patterns als Hobby schreiben. Man wird allerdings nur in den seltensten Fällen eine Sammlung von Patterns in einer Dissertation verwenden können.

Pseudopatterns

Eine weitere Komponente des Daseins als Akademiker ist, dass man veröffentlichen sollte. Wie dargestellt, bekommt man durch den Reviewprozess von Patterns sehr hilfreiches Feedback, und es gibt dort nicht die Kultur, „selbst erfundene Lösungen“ sofort zurückzuweisen. Weiter produzieren die meisten Pattern-Konferenzen eine zitierfähige Publikation, die aufgrund der positiv und hilfreich eingestellten Pattern-Community häufig leichter zugänglich ist als konventionelle Zeitschriften, bei denen es lediglich Annahme oder Ablehnung gibt – aber selten aufwändige Überarbeitungsprozesse und Workshops mit Experten. Pattern-Konferenzen verführen also viele Akademiker dazu, Dinge einzureichen, die beim besten Willen keine „3 independent known uses“ haben – weil sie ihre neuesten Erfindungen in der Form von Patterns beschreiben.

Prinzip der Nicht-Originalität

Es kann also festgehalten werden: Patterns sind für „echte Akademiker“ nicht der Weg, um an besonders respektierte Veröffentlichungen zum kommen. Beide Communitys verfolgen unterschiedliche Ziele: Die Pattern-Community will bekanntes Wissen so dokumentieren, dass es optimal verwendbar ist und optimal kombiniert werden kann. Hier ist genau die Nicht-Originalität der Wert der Community. Die Forschung will neues Wissen und neue Konzepte generieren. Hier geht im Zweifelsfall Originalität vor unbedingter sofortiger praktischer Ein-

satzfähigkeit. Beide Kulturen können problemlos koexistieren, voneinander lernen und sich gegenseitig befruchten. Grenzgänge werden oft produktiv sein.

Wolfgang Keller
Objectarchitects

Literatur

Coplien JO (1996) Software patterns. SIGS Management Briefings

Gamma E, Helm R, Johnson R, Vlissides J (1995) Design patterns: Elements of reusable object-oriented software. Addison-Wesley Longman, Amsterdam

Portland Pattern Repository (oJ) History of patterns. http://c2.com/cgi-bin/wiki?HistoryOfPatterns. Abruf am 2009-07-11

Patterns in der Wirtschaftsinformatik

Im Software-Engineering erfreuen sich die ursprünglich aus der Architekturlehre stammenden Patterns seit mehr als 15 Jahren großer Beliebtheit. Bevor Gründe erörtert werden, warum der Pattern-Ansatz bis heute kaum Anwendung für wirtschaftsinformatiktypische Fragestellungen fand, einige Bemerkungen, wie Patterns entstehen (siehe beispielsweise http://hillside.net/patterns/writing/writingpatterns.htm): „Gute“ Patterns sind praxiserprobt und werden von der Community als solche akzeptiert. Sie erfordern somit Praxis-Know-how und -erfahrung, in vielen Fällen entsprechende Branchenkenntnisse und eine Community, in der diese diskutiert und verfeinert werden.

Prädestiniert für die Erarbeitung und Evaluation von „Wirtschaftsinformatik-Patterns“ sind Unternehmensberater. Überspitzt formuliert besteht ihr Geschäftsmodell darin, möglichst viele Situationen in unterschiedlichen Unternehmen kennen gelernt zu haben und dann beurteilen zu können, welcher Lösungsansatz in der vorliegenden Situation trägt. Berater wenden somit permanent Patterns an, nur sind diese nicht als solche formuliert. Allerdings haben viele Berater und Beratungsunternehmen meist kein Interesse daran, ihr Know-how Dritten zugänglich zu machen. Gleiches gilt für die Unternehmen selbst, die sich über ihre Geschäftsmodelle und/oder die den Geschäftsmodellen zugrunde liegenden Produkte und Geschäftsprozesse am Markt differenzieren.

Warum ist dies im Software-Engineering anders? Dort scheint ein anderer Geist zu herrschen: Es gibt genug Experten, die gerne über ihre Erfahrungen berichten und diese auch publizieren. Einmal sind Software-Patterns meist ein ganzes Stück von den differenzierenden Elementen der Unternehmen entfernt. Ein weiterer Grund ist sicher, dass die Entwicklung von Softwaresystemen schwierig genug ist; alleine mit Patterns werden solche Systeme nicht gebaut. Dem-

gegenüber steht bei vielen Wirtschaftsinformatik-Fragestellungen die Entwicklung eines Konzepts im Vordergrund, und da kann die Anwendung von Patterns schon ein Großteil der Lösung sein.

Zentrale Technik der Wirtschaftsinformatik ist die Modellierung. Folgerichtig werden in der Wirtschaftsinformatik permanent neue Modellierungsmethoden, oft mit einem Fokus auf Modellierungssprachen, entwickelt, vorgeschlagen und evaluiert. Dies hat durchaus seine Berechtigung, aber Anwender suchen eine Lösung für ihre Probleme. Hierfür würden in vielen Fällen als Lösungsmuster formuliere Patterns mehr helfen als Methoden für die Erarbeitung von Lösungen. Ein Ansatz der Wirtschaftsinformatik, Lösungen wiederverwendbar zu beschreiben, sind Referenzmodelle. Jedoch zeigen viele Praxisprojekte, dass beispielsweise Referenzgeschäftsprozessmodelle nur eingeschränkt hilfreich sind. Spätestens auf der Ablaufebene sind die Unterschiede zu groß. Wer Geschäftsprozesse mehrerer Unternehmen einer Branche kennt, wird dies bestätigen können. Auch sind den genutzten Modellierungssprachen meist keine Konzepte inhärent, um die mit ihnen erstellten Referenzmodelle auf den eigenen Bedarf anzupassen. Dies ist Aufgabe komplementärer Methoden. Untersucht man diese Methoden, findet man viele Gemeinsamkeiten zur Beschreibung von Patterns. Hier sehen wir in der Beschreibung als Patterns einen zielführenderen Ansatz. Der große Erfolg von ITIL (IT Infrastructure Library) zeigt, dass dies möglich ist – auch wenn die dort beschriebenen „Best Practices“ (leider) nicht als Patterns beschrieben sind.

Wir sehen wenige Unterschiede zwischen dem Problemlösungsprozess und „normalen“ Geschäftsprozessen. Bei letzteren wird zwar nicht von Modellierung gesprochen, gleichwohl weisen beispielsweise die in der Finanzbuchhaltung, dem Controlling und auch in den Kerngeschäftsprozessen bearbeiteten Geschäftsobjekte viele Eigenschaften von Modellen auf. Wenn also Patterns für Geschäftsprozesse formuliert werden können, dann genauso auch für den Problemlösungsprozess, also aus Modellierungssicht den Prozess, wer die Modelle erstellt, aktualisiert, freigibt und wie sie genutzt werden. Ein entsprechender Bedarf ist jedenfalls vorhanden. Erfolgreiche Beispiele finden sich wieder im Software-Engineering bei der Beschreibung von Vorgehensmodellen und ihren Tailoring-Konzepten als Patterns.

Bei Wirtschaftsinformatik-Patterns ist allerdings zu berücksichtigen, dass die Domäne erheblich größer ist als im Software-Engineering. Patterns können sich auf unterschiedliche Betrachtungsobjekte beziehen, beispielsweise für Geschäftsmodelle, Geschäftsprozesse, Organisationsstrukturen, Services u. v. a. m. Auch sollte die Branche als weitere Dimension berücksichtigt werden. Viele Praktiker – und ihre Mitarbeit ist bei der Formulierung von Patterns unabdingbar – werden aber von ihren Unternehmen nur dann die Genehmigung zur Mitarbeit

erhalten, wenn es sich um Bereiche handelt, die kein Differenzierungspotenzial aufweisen. Anhand der öffentlichen Verwaltung und Teilen des Gesundheitsbereichs wird allerdings deutlich, dass dies auch in ganzen Branchen funktionieren kann. ITIL, das ursprünglich aus der öffentlichen Verwaltung stammt und sich auf den immer mehr zur Commodity werdenden IT-Bereich bezieht, ist auch hier ein gutes Beispiel. Im Hinblick auf die Differenzierungsmöglichkeiten ist anzumerken, dass sich diese durch Konzepte wie SOA (serviceorientierte Architekturen) und SaaS (Software as a Service) zukünftig noch stärker auf die fachliche Ebene verschieben werden. Dadurch entstehen ein entsprechender Bedarf und großes Potenzial für Patterns zur Identifikation und Nutzung von Services; insbesondere auch deshalb, weil das Problem der semantischen Beschreibung von Services nach wie vor ungelöst ist.

Zusammenfassend würden sich die Autoren freuen, wenn Patterns in der Wirtschaftsinformatik zukünftig eine größere Rolle spielen; sind aber skeptisch, ob nicht die oben aufgeführten Gründe dem an einigen Stellen entgegenstehen.

Dr. Stefan Junginger
Christoph Moser
BOC AG, Wien

Morgenstern, Abendstern und Venus – Zum Gebrauch der Wörter „Referenzmodell“ und „Pattern“

Während man früher glaubte, dass der Morgen- und Abendstern zwei verschiedene Sterne seien, weiß man heute, dass der Planet Venus morgens als Morgenstern am Osthimmel und abends als Abendstern am Westhimmel beobachtet werden kann. Ähnlich scheint uns die Situation bei Referenzmodellen und Patterns zu sein: Wir glauben, dass Referenzmodelle und Patterns zwei Formen ein und derselben zentralen Idee darstellen. Daher schlagen wir vor, nicht mehr terminologisch zwischen Referenzmodellen und Patterns zu unterschieden.

Vermutlich wird unser Vorschlag auf Widerspruch stoßen: So zeigt die Durchsicht der einschlägigen Literatur, dass bisher zwischen der Referenzmodell- und Pattern-Community kaum ein wissenschaftlicher Austausch stattfindet. Trotzdem plädieren wir dafür, beide Konzepte zukünftig nicht mehr zu differenzieren.

Wir räumen ein, dass die Situation Anfang der 1990er-Jahre anders gelagert war: Scheer hat mit dem Y-CIM-Modell ein klassisches Referenzmodell entwickelt; Gamma hat mit seiner 1992 erschienenen Dissertation den Grundstein der Pattern-Community gelegt, die 1995 mit dem Werk „Design Patterns“ von Gamma et al. auf eine breite Basis gestellt wurde.

Die ursprünglichen Ideen eines Referenzmodells beziehungsweise eines Patterns sind zweifelsfrei grundverschieden:

- Merkmal „Abstraktionsgrad“: Klassische Referenzmodelle beschreiben fachkonzeptionelle Aspekte eines Informationssystems. Dagegen beschreiben klassische Patterns implementierungstechnische Aspekte eines Softwaresystems.
- Merkmal „Domänenbezug“: Klassische Referenzmodelle beschreiben betriebswirtschaftliche Anwendungsdomänen wie Industrie, Handel und Dienstleistung. Klassische Patterns haben keinen betriebswirtschaftlichen Domänenbezug.
- Merkmal „Granularität“: Klassische Referenzmodelle sind grobgranular, sie beschreiben ein ganzes Unternehmen. Klassische Patterns beschreiben nur einen kleinen Aspekt eines Softwaresystems.
- Merkmal „Anspruch“: Klassische Referenzmodelle haben den Anspruch, das Wissen über eine Anwendungsdomäne wissenschaftlich zu formulieren. Klassische Patterns erheben diesen Anspruch nicht, vielmehr weisen sie explizit darauf hin, innerhalb der Praxis der Softwareentwicklung nützlich zu sein (Faustregel: „Ein Pattern muss mindestens dreimal in einem realen Projekt eingesetzt werden, um als solches gelten zu können.“).
- Merkmal „Sprache“: Klassische Referenzmodelle nutzen ereignisgesteuerte Prozessketten, Entity-Relationship-Modelle oder Funktionsbäume als Modellierungssprache. Klassische Patterns basieren auf objektorientierten Sprachkonzepten.

In den letzten Jahren haben sich allerdings die ursprünglichen Ideen grundlegend verändert:

- Merkmal „Abstraktionsgrad“: Referenzmodelle werden auch für DV-technische Aspekte beschrieben (Boles 2002), viele Patterns erfassen jetzt auch fachkonzeptionelle Aspekte (Fowler 1997).
- Merkmal „Domänenbezug“: Einhergehend mit der Verschiebung des Abstraktionsgrades verlieren Referenzmodelle teilweise ihren Domänenbezug und Patterns erhalten teilweise einen Domänenbezug.
- Merkmal „Granularität“: Innerhalb der Referenzmodellierung gibt es inzwischen eine Fülle von Arbeiten, die auch versuchen, feingranulare Modelle zu beschreiben (Remme 1997). Gleichzeitig werden auch grobgranular Patterns beispielsweise in Form von „Pattern Languages“ beschrieben (Evitts 2000).
- Merkmal „Anspruch“: Mit Referenzmodellen wird häufig nicht mehr explizit der Anspruch des wissenschaftlichen Wissens verbunden, vielmehr wird „nur“ die Wieder-

verwendung betont (Fettke u. vom Brocke 2008). Gleichzeitig gibt es Patterns, die den Anspruch an Wissenschaftlichkeit erheben (Sinz 1998; Tichy 1997).

- Merkmal „Sprache“: Inzwischen gibt es auch objektorientierte Sprachen, die bei der Formulierung von Referenzmodellen eingesetzt werden (Schlagheck 2000). Ebenso werden Patterns nicht mehr auf objektorientierte Konzepte reduziert (Ambler 1998).

Darüber hinaus gibt es Entwicklungen, die in beiden Communitys zu beobachten sind: Während sowohl klassische Referenzmodelle als auch klassische Patterns jeweils Artefakte zur Repräsentation von Wissen über Realitätsausschnitte darstellen, werden inzwischen auch systematische Verfahren zur Erfüllung gleichartiger Aufgaben in beiden Communitys diskutiert – man spricht von so genannten Referenzvorgehensmodellen (Vering 2002) beziehungsweise Process und Workflow Patterns (van der Aalst et al. 2003).

Obige Beispiele verdeutlichen, dass Referenzmodelle und Patterns in einem klassischen Verständnis zweifelsfrei zu unterscheiden sind. Aber: Anhand welcher Merkmale kann heute sinnvoll eine Grenze zwischen beiden Konzepten gezogen werden?

Da uns eine Abgrenzung beider Konzepte anhand der genannten Merkmale allzu willkürlich erscheint, schlagen wir vor, die terminologische Differenzierung beider Konzepte fallen zu lassen und nur noch die zentrale Gemeinsamkeit der ursprünglichen Ideen herauszustellen: Bei der Entwicklung von Informationssystemen werden Handlungen zur Planung, zur Entwicklung, zur Implementierung, zum Betrieb und zur Anwendung von Informationssystemen erforscht. Vorschläge zum Vollzug derartiger Handlungen werden insbesondere durch spezifische Instrumente diskutiert. Das Wort „Instrument“ wird dabei in einer sehr weiten Bedeutung gebraucht, es bezeichnet „nicht nur Methoden im engeren Sinn von systematischen Verfahren zur Erfüllung von gleichartigen Aufgaben, sondern auch Modelle als Artefakte zur Repräsentation von Wissen über Realitätsausschnitte, Techniken als aufgaben- oder problembezogene Kombinationen von Methoden und Modellen sowie Werkzeuge als computergestützte Implementierungen von Techniken“ (Zelewski 2009). Die gemeinsame Idee von Referenzmodellen und Patterns, quasi die „Venus“, besteht darin, idealtypische Instrumente zu identifizieren und öffentlich zu dokumentieren. Das auf diese Weise produzierte Wissen über Instrumente kann sowohl wissenschaftlich erforscht wie gelehrt als auch praktisch genutzt werden.

Unser Vorschlag zur terminologischen Normierung lautet: Die Wörter „Referenzmodell“ und „Pattern“ werden gebraucht, um idealtypische Instrumente zur Entwicklung von Informationssystemen zu bezeichnen. Zur weiteren sprachlichen Differenzierung dienen die genannten

Merkmale. Inwiefern zur Bezeichnung idealtypischer Instrumente die Wörter „Referenzmodell“ oder „Pattern“ gebraucht werden, liegt somit im Auge des Betrachters.

PD Dr. Peter Fettke
Prof. Dr. Peter Loos
Institut für Wirtschaftsinformatik
Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)

Literatur

Ambler SW (1998) Process patterns. Building large-scale systems using object technology. Cambridge University Press, Cambridge

Boles D (2002) Integration von Konzepten und Technologien des Electronic Commerce in digitale Bibliotheken. Dissertation, Universität Oldenburg

Evitts P (2000) A UML pattern language. MTP, Indianapolis

Fettke P, vom Brocke J (2008) Referenzmodell. In: Kurbel K, Becker J, Gronau N, Sinz EJ, Suhl L (Hrsg) Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/, Oldenbourg, München

Fowler M (1997) Analysis patterns: Reusable object models. Addison-Wesley, Menlo Park

Remme M (1997) Konstruktion von Geschäftsprozessen – Ein modellgestützter Ansatz durch Montage generischer Prozeßpartikel. Gabler, Wiesbaden

Schlagheck B (2000) Objektorientierte Referenzmodelle für das Prozess- und Projektcontrolling. Grundlagen – Konstruktion – Anwendungsmöglichkeiten. DUV, Wiesbaden

Sinz EJ (1998) Geschäftsprozessmodellierung mit Patterns. In: Becker J, Eversheim W, Luczak H, Mertens P (Hrsg) Referenzmodellierung ′98. Anwendungsfelder in Theorie und Praxis, RWTH Aachen, S 3–1 bis 3–10

Tichy WF (1997) A catalogue of general-purpose software design patterns. In: Technology of object-oriented languages and systems (TOOLS). Santa Barbara (CD-ROM)

van der Aalst WMP, van Dogen BF, Herbst J, Maruster L, Schimm G, Weijters AJMM (2003) Workflow mining: A survey of issues and approaches. Data & Knowledge Engineering 47:237–267

Vering O (2002) Methodische Softwareauswahl im Handel – Ein Referenz-Vorgehensmodell zur Auswahl standardisierter Warenwirtschaftssysteme. Logos, Berlin

Zelewski S (2009) Wissenschaftstheorie. In: Kurbel K, Becker J, Gronau N, Sinz EJ, Suhl L (Hrsg) Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon. http://www.enzyklopaedie-der-wirtschaftsinformatik.de/, Oldenbourg, München