Relationale Datenbanken und objektorientierte Programmierung. Gibt es zwischen beiden eine Unverträglichkeit in den logischen Grundlagen?

von Holger Maaß, maass@maasster.de, 22.07.2010

Einleitung

Heute - also im Jahr 2010 - ist Informationstechnologie in allen Lebensbereichen präsenter denn je. Die Zeiten, wo Computer und Software lediglich in einigen Forschungslaboren oder militärischen Einrichtungen von Bedeutung waren, gehören längst der Vergangenheit an. Die Entwicklung von Software ist deshalb auch keine rein akademische Angelegenheit mehr, sondern ein Wirtschaftszweig von nicht unerheblicher Bedeutung. Dennoch ist die Herstellung von anwendungstauglicher Software etwas sehr kompliziertes, und von vielen informationstechnischen Modellen haben sich nur wenige als praxistauglich erwiesen.

Zwei in der industriellen Praxis besonders erfolgreiche Modelle sind dabei die relationalen Datenbanken und die objektorientierte Programmierung. So gibt es wohl kaum noch ein IT-System größeren Ausmaßes, das nicht in seinem Kern eine relationale Datenbank nutzen würde. Integrale unternehmensweite Softwaresysteme - sogenannte ERP-Systeme - sind dafür nur ein besonders prägnantes Beispiel. Zum anderen sind heute wohl die meisten Anwendungen und Programme eines gewissen Umfangs mit Hilfe von objektorientierten Programmiersprachen und Methoden entwickelt worden.

Der Erfolg dieser beiden Modelle ist Grund genug, genauer nach den Ursachen für diesen Erfolg zu fragen und sich daraufhin den logischen Grundlagen dieser Modelle zuzuwenden, um dort zumindest eine Teilantwort auf diese Fragen zu finden. Was also das relationale Modell bei den Datenbanken und das objektorientierte Modell in der Programmierung gemeinsam haben, ist ihr breiter Erfolg in der Praxis. Es mag deshalb vielleicht etwas überraschen, dass die Denkweisen, die beiden Modellen zugrunde liegen, doch sehr unterschiedlich sind, so dass man hier sogar von einer gewissen Unverträglichkeit gesprochen hat, wofür sich der Begriff "object-relational impedance mismatch" eingebürgert hat. Zwar bezieht sich dieser Begriff zunächst einmal auf gewisse Schwierigkeiten, die auftreten, wenn eine genuin objekt-orientiert entwickelte Anwendung Daten in einer relationalen Datenbank speichern muss, aber er verweist durchaus auch auf den wahrgenommenen Unterschied der Denkweisen, die sich in den beiden Modellen niederschlagen.

Nun sind die Diskussionen zur genannten Unverträglichkeit der beiden Modelle recht vielfältig und reichen von einer starken Betonung der Unüberwindbarkeit der Kluft bis hin zu Aussagen, dass es eine solche Unverträglichkeit gar nicht gebe. Mit der ersten Position sind dann natürlich auf beiden Seiten Forderungen verbunden, das jeweils andere "gegnerische" Modell in seinem Einfluss zu begrenzen und prospektiv ganz durch etwas anderes zu ersetzen. Die zweite Position verweist auf die täglich gelingende Zusammenarbeit der Modelle und sieht kaum einen Anlass für Diskussionen.

Sicher ist es eine Überdramatisierung, wenn man den gefühlten Unterschied der Modelle zu einer Art Kampf hochstilisiert, und es ist andererseits auch eine Verharmlosung der Unverträglichkeit, wenn man so tut, als gäbe es sie gar nicht.

Dass die Probleme, die mit der Unverträglichkeit auftreten, lösbar sind und auch tagtäglich von neuem gelöst werden, wird kaum jemand bestreiten. Dass die Probleme aber keine Scheinprobleme sind, werden vielleicht auch die meisten anerkennen, denn zur Lösung ist oft ein erheblicher Zeit- und Kostenaufwand nötig, den man nicht unterschätzen sollte. Der Unterschied der Denkweisen zwischen den beiden Modellen kann in Teams, die aus Anwendungsentwicklern auf der einen Seite und Datenbankentwicklern auf der anderen Seite zusammengesetzt sind, sogar zu gewissen Kooperationsproblemen führen, die auch bereits als "cultural impedance mismatch" prononciert worden sind (http://www.agiledata.org/essays/culturalImpedanceMismatch.html).

Dass sowohl die relationalen Datenbanken als auch die objektorientierte Programmierung heute in der Praxis eine herausgehobene Rolle spielen und trotzdem ein wesentlicher Unterschied in den zugrundeliegenden Denkweisen vorliegt, sollte Anlass genug sein, diesem Unterschied weiter nachzugehen, denn wenn man die beiden Denkweisen in ihrem Unterschied besser begreift, wird man sie auch im einzelnen besser verstehen und in ihrer Leistungsfähigkeit angemessener einschätzen können.

Nun kann man sich im Internet umschauen und sieht, dass zu diesem Unterschied viel geschrieben wurde und durchaus gute Ideen zu finden sind, um diesen Unterschied genauer herauszuarbeiten. Es besteht jedoch eine Tendenz, immer neue Aspekte dieses Unterschieds aufzulisten, zwischen denen kein wirklicher Zusammenhang mehr erkennbar ist und sich damit in einer Fülle von disparaten Beobachtungen zu verlieren. Wenn wir bei der Suche nach einem besseren Verständnis dieses Unterschieds wirklich weiter kommen wollen, nützt es nichts, wenn wir lediglich bei zusammenhanglosen oberflächlichen Beobachtungen stehen bleiben. Nur wenn es uns gelingt, den strukturellen Kern dieses Unterschiedes zu erfassen oder diesem wenigstens ein Stück weit auf die Spur zu kommen, können wir sagen, dass sich unser Verständnis hier vertieft hat. Das Ziel sollte also sein, den Unterschied mit einigen wenigen Aspekten charakterisieren zu können und idealerweise einen Grundaspekt herauszuarbeiten, mit dem dann alle weiteren Aspekte in Verbindung stehen.

Die Grundbegriffe "Objekt" und "Relation"

Wir wollen damit beginnen, die jeweiligen Grundbegriffe der beiden Modelle genauer unter die Lupe zu nehmen und diese zu vergleichen. Bekanntlich sind dies die Begriffe "Objekt" und "Relation". Beide Begriffe sind auf einer sehr abstrakten Ebene angesiedelt, aber im Grunde hat jeder sofort ein erstes Verständnis der Begriffe und eine Vorstellung davon, und kurioserweise kann es schon hier zu einem schiefen Verständnis des Grundunterschiedes kommen.

Unter einem Objekt würde man sich zunächst einmal so etwas wie ein Ding vorstellen, z.B. einen Stuhl, einen Ball oder ein Haus. Unter einer Relation dagegen versteht man für gewöhnlich eine Beziehung oder einen Zusammenhang zwischen verschiedenen Objekten. So stehen etwa Vater und Tochter in einer bestimmten Relation zueinander, der Vater-Tochter-Beziehung.

Hat man dies im Blick, drängt sich sofort folgender Gedanke auf: der Begriff "Objekt" deutet darauf hin, dass die Objektorientierung einer Substanz-Ontologie verpflichtet ist, welche Dinge ins Zentrum ihrer Aufmerksamkeit rückt und diese als losgelöste Einheiten zur Grundlage ihres Denkens macht. Der Begriff "Relation" akzentuiert demgegenüber die Beziehung zwischen verschiedenen Dingen und verortet von vornherein alles in seinen Zusammenhängen. Das relationale Modell ist damit einer eher ganzheitlichen Ontologie verpflichtet, die Einzelnes aus seinen Zusammenhängen heraus zu verstehen sucht.

Genau diese Sichtweise ist jedoch ein grobes Missverständnis, das daher rührt, dass wir die Begriffe "Objekt" und "Relation" aus unserem sprachlichen Alltagsgebrauch heraus zu verstehen versuchen. Als Grundlage für die von uns untersuchten informationstechnischen Modelle haben sie jedoch eine stark von diesem Verständnis abweichende Bedeutung, und wir müssen uns diese Bedeutung vor Augen führen, um nicht schon hier auf einen Holzweg zu geraten.

Der Begriff "Objekt"

Natürlich hängt der Objektbegriff des objektorientierten Modells auch mit dem erwähnten alltagssprachlichen Verständnis des Objektbegriffs zusammen. So werden in Büchern oder Artikeln zur Einführung in die objektorientierte Programmierung auch gern Artefakte wie Autos oder Glühbirnen als Beispiele für Objekte angeführt. Das Entscheidende ist jedoch, dass Objekte hier nicht primär als losgelöste Blöcke gesehen werden, sondern als in sich bereits komplex strukturierte Einheiten, die dann auch in einem bestimmten Bezug zu der sie umgebenden Welt stehen.

Was ist nun ein Objekt im objektorientierten Modell? Zwei Aspekte sind hier zur Charakterisierung wichtig.

1.) In einem Objekt werden Attribute und Methoden zu einer unauflöslichen Einheit zusammengefasst.

2.) Die Attribute und Methoden des Objekts werden nach außen hin gekapselt, wie man sagt. Das bedeutet, dass der Zugriff von außen auf diese Attribute und Methoden genau festgelegt und damit geregelt wird.

Ein Objekt hat damit eine innere Komplexität, die nach außen hin verborgen wird. Von außen her gibt es nur bestimmte wohldefinierte Stellen, an denen ein Zugriff auf das Objekt erlaubt ist, die sogenannten Schnittstellen.

Genau dieser Ansatz hat sich als äußerst fruchtbares Modell für die Programmierung herausgestellt und viele Programmierer sind zu Recht begeistert davon und sehen darin einen enormen Fortschritt gegenüber anderen logischen Modellen. Wie ist dies jedoch zu erklären?

Der Hauptgrund liegt meines Erachtens in zweierlei Dingen, die miteinander zusammenhängen. Erstens ist dieses Objektmodell eine sehr passende Abbildung dafür, wie wir in unserer Welt mit komplexen Artefakten umgehen. So kann man ein Auto in der Tat als eine nach innen sehr komplexe Struktur auffassen mit vielen verschiedenen Eigenschaften. Die angemessene Nutzung des Autos besteht dabei in ganz bestimmten Dingen, die man mit dem Auto tun darf. Man darf die Türen des Autos öffnen, sich hineinsetzen und damit auf den Straßen fahren. Natürlich darf man noch viel mehr, z.B. das Lenkrad bedienen, Gas geben, auf die Bremse treten usw. Die Scheibe des Autos einzuschlagen ist zwar auch möglich, aber man wird sie sofort als zweckwidrige Handlung brandmarken.

Auch für sehr komplexe Einheiten in unserem Leben ist das Objektmodell offenbar eine gute Abbildung. Man nehme z.B. eine Firma, die ihren Sitz in einem großen Gebäude hat. Nach außen hin sind nur bestimmte Möglichkeiten freigegeben, zu der Firma einen Bezug herzustellen. Man kann etwa in einen Briefverkehr mit der Firma treten, wenn man etwas bestellt oder sich bewirbt. Oder man geht zum Eingang hinein und meldet sich bei der Information an. Die innere Komplexität der Firma mit ihren eventuell vielen verschiedenen Abteilungen und Mitarbeitern ist dabei nach außen hin größtenteils verborgen.

Das Objektmodell ist also eine gute Abbildung von komplexen Einheiten des modernen Lebens. Das war der erste Punkt. Es gibt jedoch noch einen zweiten Aspekt, warum das Objektmodell ein geeigneter Ansatz für die Programmierung ist, und der besteht m.E. darin, dass es gerade die wesentliche Aufgabe der Programmierung darstellt, komplexe Einheiten unserer modernen Welt zu modellieren sowie deren Beziehungen untereinander und unseren Umgang damit.

Eine wichtige Motivation für die Einführung des objektorientierten Modells waren zunehmende Schwierigkeiten, komplexe Software-Projekte beherrschbar zu halten. Man machte sich dabei die Erkenntnis zunutze, dass wir in unserer Alltagswelt Komplexität dadurch beherrschen, dass wir den Umgang mit komplexen Einheiten regeln und normieren sowie deren innere Komplexität verbergen. Genau dies ist mit der Idee der Kapselung umgesetzt, wo Eigenschaften und Prozesse in einen engen Zusammenhang gebracht werden und eine bestimmte Eigenschaft nur durch wohldefinierte Prozesse geändert oder zugänglich gemacht werden darf. Geregelte und normierte Prozesse erhalten damit gegenüber den Eigenschaften bzw. Attributen oder Daten einen Vorrang, weil die Daten nur noch im Kontext solcher Prozesse überhaupt zugänglich sind. Auf diesen Punkt werden wir noch zurückkommen. Vorher wollen wir jedoch den Begriff der Relation näher beleuchten.

Der Begriff "Relation"

So wie der Begriff "Objekt" im objektorientierten Modell eine ganz bestimmte Bedeutung annimmt, die von der alltagssprachlichen abweicht, hat auch der Begriff "Relation" eine ganz bestimmte Bedeutung, wenn er zur Grundlage des relationalen Modells bei den Datenbanken wird. Der Begriff wird nämlich aus der Mathematik entlehnt und entstammt dort dem Teilgebiet der Mengenlehre. Eine Relation im mathematischen Sinne ist zwar auch eine Beziehung, aber keine Beziehung zwischen einzelnen Objekten, sondern zwischen Mengen. Die Beziehung kann dabei zwischen zwei Mengen bestehen, aber auch zwischen beliebig vielen Mengen.

Um uns den mathematischen Relationsbegriff konkreter zu veranschaulichen, können wir vom einfachen Fall einer Relation zwischen zwei Mengen ausgehen. Nehmen wir z.B. als erste Menge die Buchstaben A, B, C und als zweite Menge die Zahlen 1, 2, 3. Im nächsten Schritt ordnen wir einem Element der ersten Menge ein Element der zweiten Menge zu und bilden daraus ein geordnetes Paar (A, 2). Diesen Vorgang wiederholen wir noch einige Male (B, 3), (C, 2), (A, 1), (A, 3) und erhalten auf diese Weise eine neue Menge, nämliche eine Menge von Paaren. Genau dies bezeichnet man als Relation im Sinne der Mathematik. Wenn man noch genauer sein will, muss man diese Relation als zweistellige bezeichnen, weil sie zwischen zwei Mengen besteht. Nimmt man eine weitere Menge hinzu, z.B. die Rechenzeichen (+, -, *, :), so kann man eine dreistellige Relation bilden. Dafür bildet man nun Tripel, indem man wieder aus jeder Menge ein Element entnimmt und diese zusammenstellt (A, 2, :), (B, 3, *), (C, 1, -). Die Menge dieser Tripel ist die dreistellige Relation. Nun ist leicht zu erkennen, wie man das Verfahren auf Relationen zwischen beliebig vielen Mengen erweitern kann.

Die allgemeine mathematische Definition lautet also: eine n-stellige Relation ist eine Menge von n-Tupeln, wobei die Tupel dadurch entstehen, dass man aus n Mengen jeweils ein Element entnimmt und diese zusammenordnet.

Damit sind wir dem Begriff Relation, wie er im relationalen Modell verwendet wird, schon sehr nahe gekommen. Wir müssen nur noch ein paar kleine Wendungen vollziehen und machen aus den Mengen, deren Elemente wir entnehmen wollen, Spalten, und aus den Tupeln machen wir Zeilen. Das ganze ordnen wir in einer übersichtlichen rechteckigen Gitterform an und erhalten das, was gemeinhin eine Tabelle genannt wird:

Name	Vorname	Wohnort
Schmidt	Erich	Hamburg
Dubois	Jean	Paris
Rossi	Adriana	Florenz
Butcher	James	Bath

Die erste Zeile - hier durch eine Doppellinie abgetrennt - hat dabei eine Sonderstellung. Sie ist die Titelzeile und enthält die Bezeichnungen der Grundmengen. Die weiteren Zeilen sind die Tupel mit den zusammengeordneten Elementen. Eine Zeile entsteht also dadurch, dass man einen Namen, einen Vornamen und einen Wohnort zusammenordnet.

Im relationalen Modell macht man sich also die Einsicht zunutze, dass man eine Relation im mathematischen Sinne als Tabelle aus Spalten und Zeilen darstellen kann bzw. umgekehrt, dass man eine Tabelle als Relation im mathematischen Sinne interpretieren kann. Die Fruchtbarkeit dieser Verknüpfung hat der Begründer des relationalen Modells, E.F. Codd, wohl als erster in ihrer ganzen Tragweite erfasst und zum Ausgangspunkt eines systematisch ausgebauten Modells für Datenbanken gemacht. Entscheidender Punkt dabei ist, dass man die Möglichkeiten der mathematischen Relationen und ihres Operationssystems - der relationalen Algebra - mit einer besonders anschaulichen Repräsentationsform von Relationen - der Tabelle - verknüpft.

Alle Informationen in der relationalen Datenbank sind in Tabellenform angeordnet. Eine andere Darstellungsform gibt es nicht. Die Anordnungsform einer Tabelle ist aber für sehr viele Menschen eine besonders leicht zugängliche und übersichtliche Darstellungsform. Sicher können Tabellen sehr groß sein und nicht mehr einfach so mit einem Blick überschaut werden, aber ein Ausschnitt der Tabelle, der klein genug ist, kann immer leicht erfasst werden. Das ist wohl einer der Hauptgründe dafür, dass das relationale Modell bis heute so erfolgreich geblieben ist: es bringt eine simple Oberfläche mit. Der zweite Hauptgrund sind die weitreichenden Möglichkeiten, die das relationale Modell aus der mathematischen Mengenlehre und dem Operationssystem der relationalen Algebra mitbringt. Wir werden darauf weiter unten noch einmal zurückkommen.

Wir halten also fest: der Begriff "Relation" meint im relationalen Modell die Anordnung von Daten in Form einer Tabelle als Grundelement des Datenbanksystems.

Objekt und Relation im Vergleich

Nachdem wir die beiden Grundbegriffe der hier betrachteten informationstechnischen Modelle jeweils für sich allein näher umrissen haben, wollen wir nun dazu übergehen, die Begriffe miteinander zu vergleichen und aufeinander zu beziehen, um dabei die Grundaspekte der Modelle schärfer konturieren zu können.

Wenn man nun ein Objekt direkt mit einer Relation vergleichen will, so muss man im Grunde zugeben, dass das nicht geht, weil die beiden Begriffe auf verschiedenen logischen Ebenen angesiedelt sind. Ein Objekt ist nämlich eine konkrete Ausformung eines Objekttyps bzw. einer Objektform. Dies wird im objektorientierten Modell meist als Klasse bezeichnet. Eine Klasse ist die formale Definition, der dann konkrete Objekte entsprechen können. Sie ist eine Art Bauplan für Objekte. In der Klassendefinition wird festgelegt, welche Attribute (Eigenschaften) und Methoden (Verhaltensweisen) zu einem Objekt gehören, das der Klasse angehört.

In gewisser Weise liegen also die Begriffe "Klasse" und "Relation" auf derselben logischen Ebene, weil man die Klasse als Menge von Objekten auffassen könnte und die Relation als Menge aller Zeilen einer Tabelle. Ein Objekt könnte man also mit einer Tabellenzeile vergleichen. In erster Annäherung können wir so vorgehen, aber wir werden noch sehen, dass dieser Vergleich in gewisser Weise auch hinkt. Aber gut, ein Vergleich kann ja sowohl Gemeinsamkeiten als auch Unterschiede zutage fördern.

Wenn wir den Vergleich auf dieser Ebene durchführen, sehen wir auf jeden Fall, dass man die Spalten einer Relation als deren Attribute auffassen kann und die dann durchaus mit den Attributen einer Objektklasse strukturparallel sind. Die Attribute sind sowohl in einer Klasse als auch in einer Relation als Formen definiert, denen dann konkrete Werte zugewiesen werden. Im objektorientierten Modell geschieht das in einem konkreten Objekt und im relationalen Modell in einer einzelnen Tabellenzeile.

Beim Vergleich von Klasse und Relation fällt aber auch sofort ein Unterschied auf. Klassen haben eigene Methoden, Relationen nicht. Und mehr noch: die Methoden haben in einer Klasse sogar einen klaren Vorrang vor den Attributen, denn die Attribute können nur über die festgelegten Methoden geändert und ausgelesen werden. Die Attribute einer Klasse und damit auch eines Objekts sind also von gewissen Methoden abhängig.

Mit der Abhängigkeit der Attribute von den Methoden hängt noch ein weiterer wesentlicher Unterschied von Klassen und Relationen zusammen. Die Attribute von Klassen und damit deren konkrete Werte in Objekten sind nach außen gekapselt und nur über wohldefinierte Schnittstellen zugänglich. Eine solche Kapselung gibt es bei Relationen nicht.

Aber es gibt noch einen weiteren wesentlichen Unterschied, der jedoch subtiler und damit schwerer zu erkennen ist. Wir wollen aber gerade deshalb besonderes Augenmerk auf ihn lenken, weil die Gefahr groß ist, dass man ihn übersieht. Die Spalten einer Tabelle sind im Grunde die Form einer Relation, die dann in den Zeilen der Tabelle mit konkreten Werten belegt werden. In analoger Weise ist die Klasse mit ihren Attributen und Methoden die Form der Objekte, in welchen dann die Attribute konkrete Werte bekommen. Eine Relation ist aber nicht nur eine Form, sondern sie fasst auch die einzelnen Zeilen zu einer Einheit zusammen. Eine Klasse hingegen ist zwar die Form ihrer Objekte, aber nicht deren Einheit, und somit kann man eine Klasse auch nicht wirklich als Menge ihrer Objekte bezeichnen, wie wir das zunächst getan hatten. Wir müssen diese Aussage also hier wieder zurücknehmen und werden noch sehen, dass dies von entscheidender Bedeutung ist.

Die Quintessenz unseres Vergleichs zwischen Objekt und Relation können wir nun wie folgt zusammenfassen. Sowohl ein Objekt - als Exemplar einer Klasse - als auch eine Relation sind durch Eigenschaften (Attribute) bestimmt. Die Eigenschaften eines Objekts sind aber nur über genau festgelegte Methoden zugänglich und manipulierbar. Die Koppelung der Eigenschaften an Methoden existiert bei Relationen nicht. Dort sind dafür die konkreten Wertbelegungen der Eigenschaften - die sich in den Zeilen der Tabelle manifestieren - zu einer Einheit zusammengefasst, die als Menge der

Zeilen aufgefasst werden kann. Dafür wiederum gibt es bei Objekt und Klasse keine Entsprechung. Es gibt also eine grundlegende Gemeinsamkeit auf beiden Seiten - die Eigenschaften - und jeweils einen grundlegenden spezifischen Aspekt auf jeder Seite, der auf der anderen Seite des Vergleichs fehlt.

Prozesse und Zustände

Nach dem Vergleich der Begriffe "Objekt" und "Relation" wollen wir nun dazu übergehen, den Unterschied der beiden Modelle auf einen logischen Kern zurückzuführen. Hierzu können wir das Augenmerk zunächst noch einmal auf die Koppelung von Attributen an Methoden im objektorientierten Modell lenken und auf unsere Aussage zurückkommen, dass in diesem Modell die Prozesse einen gewissen Vorrang vor den Eigenschaften haben. Zwar haben die Prozesse bzw. Aktionen in diesem Modell auch immer einen Bezug zu Attributen, aber Attribute existieren nicht unabhängig von Prozessen. Die Prozesse sind gewissermaßen die Hauptakteure im objektorientierten Modell. Durch sie werden Daten erzeugt, geändert, gelöscht und mitgeteilt. Es ist also primär die Bewegung der Daten, die im objektorientierten Modell abgebildet wird.

Ganz anders sieht es im relationalen Modell aus. Dort werden vorliegende Daten von den sie erzeugenden Prozessen gerade entkoppelt und in einer bestimmten Ordnung bzw. Anordnung zusammengestellt. Zwar bleiben auch die Daten in einer relationalen Datenbank beweglich und können sich ändern, aber der Fokus liegt vom Modell her auf den Daten in einem bestimmten Zustand, also wenn sie gerade nicht in Bewegung sind. Diese - wenn man so will - eingefrorenen Daten werden im relationalen Modell systematisch angeordnet, so dass sie ganz flexibel zueinander in Beziehung gesetzt und auch umgeordnet werden können. Man versteht dann auch, warum nur hier eine mengenorientierte Sichtweise systematisch sinnvoll ist, denn nur Daten im Sinne von eingefrorenen Zuständen lassen sich zu Mengen zusammenfassen. Eine Menge von Ereignissen oder Prozessen wäre zunächst ein Widersinn. Erst im eingefrorenen Zustand kann man Ereignisse zu einer Menge zusammenfassen.

Den Kern des Unterschiedes zwischen unseren beiden Modellen können wir also so formulieren: das objektorientierte Modell eignet sich in idealer Weise um Prozesse abzubilden und Daten im Moment ihrer Bewegung. Das relationale Modell dagegen eignet sich besonders gut, um Daten im Moment ihrer Unbeweglichkeit abzubilden und in eine festgelegte Ordnung aufzunehmen.

Die Unverträglichkeit der beiden Modelle ist damit in der Tat bestätigt. Aber es zeigt sich auch ihre Komplementarität, denn Daten spielen in beiden Modellen eine entscheidende Rolle, wenn auch gewissermaßen in verschiedenen Existenzweisen. Die Daten sind damit auch als Schnittstelle zwischen beiden Modellen anzusehen und haben eben nun mal die Eigenheit, einerseits innerhalb von Prozesskontexten aufzutauchen und andererseits innerhalb von eingefrorenen Ordnungskontexten. Die Unverträglichkeit ist damit auch kein bloßer Zufall, der auf ungenügendes Modelldesign zurückzuführen wäre, sondern sie liegt in der Verschiedenheit der Kontexte begründet, in denen die beiden Modelle zum praktischen Einsatz kommen.

Aber was haben wir denn nun eigentlich damit gewonnen, dass wir den Unterschied der beiden Modelle auf einen einfachen Kern zurückgeführt haben? Im Grunde kann es doch eigentlich nur falsch sein, wenn man ein so vielfältiges Problem wie den "object-relational impedance mismatch" auf einen Kern zusammenzieht. Ist das nicht ein unzulässiger Reduktionismus? Und ist es nicht andererseits auch nutzlos, so eine Reduktion durchzuführen, weil man die Vielgestaltigkeit des Phänomens dabei aus den Augen verliert und am Ende eher dümmer und nicht klüger ist als vorher? Diese Fragen drängen sich natürlich auf und sie sind durchaus berechtigt. Wenn wir nämlich einfach an dieser Stelle stehenbleiben würden, hätten wir nur die Hälfte geschafft und vielleicht noch nicht einmal das. Was also haben wir bis hierhin erreicht? Und was bleibt noch zu tun?

Indem wir versucht haben, den Unterschied der beiden Modelle auf eine vergleichsweise einfache Kernthese zurückzuführen, haben wir lediglich versucht, einen zentralen Punkt zu finden, von dem aus wir die außerordentliche Vielgestaltigkeit des Problems überschauen können. Das Ziel dieses Vorgehens war es, der erdrückenden Komplexität zu entkommen, mit der uns das Problem im beruflichen Alltag begegnet. Wenn wir aber nur sehen, dass objektorientierte Programme und relationale Datenbanken nicht zusammenpassen und sich dies in zahllosen Problemen bemerkbar macht, dann kann das schon eine sehr frustrierende Situation sein.

Es bleibt aber in der Tat noch etwas wesentliches zu tun, wenn uns unsere Kernthese helfen soll, die Lage besser zu überschauen. Wir müssen nämlich nun die zahlreichen Aspekte der Unverträglichkeit, die uns an der konkreten Oberfläche unseres Berufsalltags begegnen, zu dieser Kernthese in Beziehung bringen bzw. sie daraus ableiten oder in deren Horizont zu verstehen versuchen. Erst dann werden wir wirklich das Gefühl haben, klüger geworden zu sein und die praktischen Probleme auch anders angehen zu können. Diese Aufgabe ist natürlich nicht abschließbar, aber wir werden versuchen, sie an einigen Beispielen durchzuführen, um damit die Plausibilität der Kernthese zu untermauern.

Fortsetzung folgt. Ein paar Stichworte für die Fortsetzung sind folgende:

- Ereignisse (Prozesse) und deren Formen
- Daten (Zustände) und deren Mengen
- Ereignislogik vs. Prädikatenlogik
- Aggregationsattribute