# Inhaltsverzeichnis

# Vorwort

*In theory, there is no difference between theory and practice. But, in practice, there is.*

Jan L.A. van de Snepscheut

Eigentlich lässt sich dem Sprichwort noch hinzufügen, dass es nichts Praktischeres gibt, als eine gute Theorie. Das Jahr 1995 war in der Praxis ein sensationelles Jahr. Es war nicht nur die Geburtsstunde des Buches „Design Patterns“ der Gang of Four, nein, es wurden weitere Meilensteine in der Informatikgeschichte gelegt. Mit Windows 95 begann die Ära der 32-Bit-Betriebssysteme, parallel wurde die Pentium-Architektur begründet, die OMG definierte in diesem Jahr die Basis für UML und zu guter Letzt war es auch das Geburtsjahr von Delphi. Ein ereignisreiches Jahr also.

All diesen Ereignissen ging eines voraus: Das Design, das vom Syntaxbaum bei Delphi über das Layout des Pentiums bis hin zum Funktionsmuster des Linux-Kernels reicht. Ich pflege oft zu sagen: „Für jedes Problem gibt es ein Diagramm“, und so will ich es auch halten. Ich gehe davon aus, dass in einigen Jahren auch Politiker, Gewerkschaften oder Ärzte ihre Bedürfnisse oder Lösungen mit UML modellieren wollen, frei nach dem Motto: weniger Gerede, mehr Anwendungsfälle.

## Präzisionssprache

In der Softwareentwicklung stolpert man oft darüber, dass präzise technische Anforderungen mit unscharfer Sprache kollidieren. Dies verdeutlicht z.B. der folgende Satz: „Dieser Abschnitt beschäftigt sich mit den Ereignissen, die diese Botschaften auslösen“. Sind es nun die Ereignisse, welche die Botschaften auslösen oder umgekehrt? Hier kann die Notation einer Grafik eindeutig zu mehr Klarheit führen. Deshalb sind viele der im Buch erstellten Diagramme so genau wie der Code selbst.

Softwaresysteme sind ähnlich wie Sprachen so lange präzise, bis eines Tages ein Fehler aus der morbiden Dunkelkammer der digitalen Ästhetik ans Tageslicht kommt. Vor allem dann, wenn die Ergebnismenge beispielsweise eines Datawarehouse die statistische Aussage ermittelt, dass die meisten Säuglinge ledig sind oder dass es vor allem Frauen sind, die häufig schwanger werden.

So meine ich: ein Diagramm ist schlussendlich mehr Wert als Worte!

## CODESIGN

Die bisherigen Visionen von Entwicklern beschreiben Software gleich Kathedralen aus Licht und pulsierenden Datenbahnen. Wenn Software sichtbar wird, dann seien mit den in diesem Buch vorgestellten Musterlösungen drei Ziele von Bedeutung: Aufbau der Vorstellungskraft, Erlernen des Musterkatalogs und Praktizieren im täglichen Code.

Einer späteren Erleuchtung sei in Anbetracht des Jahres 1995 nichts entgegenzuhalten, da ja jedes erstellte Diagramm seine Reise im morphogenetischen Feld beendet ;).

Im ersten Teil des Buchs werden die Prozesse und Techniken vorgestellt, die sich für Patterns eignen und zur Verwendung kommen. Es sind dies unter anderem OCL, MDA, Refactoring, Interfaces, Listen, Delegation oder Komposition, einfach das solide Rüstzeug, um erfolgreich Patterns in Schulung oder Projekten einzusetzen.

Es folgen im zweiten Teil die 24 Design Patterns in reich bebilderter Delphi- und C#-Notation in Beispiel und Verwendung. Die häufigsten Code Patterns als Idiome zielen auf den Schnelleinsatz ab.

Im dritten Teil runden 15 Architekturmuster das Buch als Kompendium ab. Die meisten der konkreten Verwendungen beleuchten die Crossplattform CLX, sodass die Beispiele auch für C++Builder- oder Kylix-Entwickler von Interesse sind. Ein letzter Schwerpunkt bilden die Komponenten einer verteilten Architektur wie z.B. Web Services oder Provider dies bieten.

Da Patterns für allgemeine Probleme, die immer wieder vorkommen, Lösungen anbieten, liegt es nahe, sie für den gesamten Bereich Softwarebau oder Schulung einzusetzen.

## Dank

Bei Bernhard und Silvia möchte ich mich für die wertvolle Arbeit bezüglich Struktur und Code bedanken, die auch eine nötige Energiequelle darstellt, denn ohne Identifikation auch keine Motivation. Herzlichen Dank auch Beat Strähl für die präzisen Hinweise und der Community für die vielen Ideen und Anregungen, vor allem den Autoren aus aller Welt auf delphi3000.com. Dem Verlag ein Dankeschön, da ich nach mittlerweile 6 Jahren immer wieder Ideen realisieren kann. Schlussendlich gebührt mein Dank wieder Franz Zucol, der Zeit gefunden hat, mit mir über Design zu diskutieren.

Als dann der Bart immer länger wurde, kam mir wieder mal zu Gesicht, ohne Leben kein Tod und ohne Tod keine Ewigkeit, also habe ich meine endliche Zeit auch gründlich einzuteilen.

## Ausblick

Mit der Model Driven Architecture realisiert die Standardisierungsorganisation Object Management Group (OMG) einen weiteren Schritt in Richtung Executable UML.

Kernidee der MDA ist die schrittweise Verfeinerung des Applikationsentwurfs ausgehend von einer Modellierung der Applikations-/Geschäftslogik, die völlig unabhängig von der späteren technischen Implementierung und der umgebenden IT-Infrastruktur ist.

So wird man wohl künftig weniger programmieren und vermehrt einen Modellkatalog wie „Patterns konkret“ konsultieren dürfen. Genauso wenig allerdings wie ein Synthesizer das Komponieren guter Musik garantiert, werden Patterns und Compiler den Entwurf guter Software garantieren. Doch wie in der Musik wird sich auch im Softwarebau die Erkenntnis durchsetzen, auf die Notation kommt es an.

In diesem Sinne, may the source „ein weiteres Mal“ be with you

*Max Kleiner, Dr. Silvia Rothen, Dr. Bernhard Angerer*

Juni 2003