

Vaughn Vernon

# Domain-Driven Design kompakt

→ Aus dem Englischen übersetzt
von Carola Lilienthal und Henning Schwentner

dpunkt.verlag

**Domain-Driven Design kompakt**

**Vaughn Vernon** ist ein »Software Craftsman« mit mehr als 30 Jahren Erfahrung in Softwaredesign, Entwicklung und Architektur. Seine Leidenschaft ist die Vereinfachung von Softwaredesign und die Implementierung mit innovativen Methoden. Er ist Autor der Bücher »Implementing Domain-Driven Design (IDDD)« und »Reactive Messaging Patterns with the Actor Model« (beide Addison-Wesley). Vaughn hat sein Wissen in seinem IDDD-Workshop bereits an Hunderte Softwareentwickler überall auf der Welt weitergegeben und spricht regelmäßig auf Fachkonferenzen. Er interessiert sich vor allem für verteilte Systeme, Messaging, und das insbesondere in Kombination mit dem Aktorenmodell, und ist als Berater für Domain-Driven Design sowie für DDD unter Verwendung des Aktorenmodells mit Scala und Akka tätig. Seine aktuellen Veröffentlichungen sind in seinem Blog *www.vaughnvernon.co* zu finden oder bei Twitter unter *@VaughnVernon*.

**Die Übersetzer:**

**Dr. Carola Lilienthal** (@cairolali auf Twitter) ist Softwarearchitektin bei der WPS – Workplace Solutions GmbH. 2015 hat sie ihre Erfahrungen aus über hundert Architekturanalysen in Java, C#, C++, ABAP sowie PHP in dem Buch »Langlebige Softwarearchitekturen – Technische Schulden analysieren, begrenzen und abbauen« zusammengefasst. Regelmäßig gibt sie ihr Wissen auf Konferenzen, in Artikeln und bei Schulungen weiter.

**Henning Schwentner** (@hschwentner) liebt Programmieren in hoher Qualität. Diese Leidenschaft lebt er als Softwarearchitekt, Berater, Trainer und Entwickler bei der WPS – Workplace Solutions GmbH aus. Seine Projekte sind Domain-Driven, agil und in Programmiersprachen wie Java und C#, aber auch ABAP geschrieben. Er gibt regelmäßig Workshops zum Thema Domain-Driven Design.

**Vaughn Vernon**

# Domain-Driven Design kompakt

Aus dem Englischen übersetzt
von Carola Lilienthal und Henning Schwentner

dpunkt.verlag

Vaughn Vernon · vaughn@forcomprehension.com
Carola Lilienthal · cl@wps.de
Henning Schwentner · hs@wps.de

Lektorat: Christa Preisendanz
Übersetzung: Carola Lilienthal und Henning Schwentner
Copy-Editing: Ursula Zimpfer, Herrenberg
Satz: Birgit Bäuerlein
Herstellung: Susanne Bröckelmann
Umschlaggestaltung: Helmut Kraus, www.exclam.de
Druck und Bindung: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

ISBN:
Print 978-3-86490-439-4
PDF 978-396088-178-0
ePub 978-3-96088-179-7
mobi 978-3-96088-180-3

1. Auflage 2017

Wieblinger Weg 17
69123 Heidelberg

Authorized translation from the English language edition, entitled Domain-Driven Design Distilled, 1st Edition by Vaughn Vernon, published by Pearson Education, Inc, publishing as Addison-Wesley Professional, Copyright © 2016 by Pearson Education, Inc.
ISBN 978-0134434421

5 4 3 2 1 0

✣ Nicole und Tristan ✣

*Wir haben es wieder geschafft!*

# Vorbemerkung der Übersetzer

Obwohl das Buch, in dem Domain-Driven Design (DDD) zum ersten Mal vorgestellt wurde – das »Big Blue Book« [Evans 2004] –, schon 2004 erschienen ist, gab es bisher keine Übersetzung eines DDD-Buches ins Deutsche. Das war vielleicht in Ordnung, weil man sowieso viel Zeit braucht, um sich intensiv mit einem dicken Buch auseinanderzusetzen. Und die Fremdsprache macht die Hürde dann auch nicht entscheidend schwieriger.

Aber ein knappes Werk, wie das vorliegende Buch, soll einfach und schnell zugänglich sein und den Leser nicht dadurch in seinem Lesefluss verlangsamen, dass es ihn dazu zwingt, jedes zweite Wort im Wörterbuch nachzuschlagen. Deshalb haben wir es übersetzt. Wir hoffen, dass es Entwicklern nicht nur hilft, selbst in DDD einzutauchen, sondern auch Manager, *Domain Experts* und sonstige Stakeholder davon überzeugen kann.

Wenn man ein Sachbuch übersetzt, stellt sich die Frage, wie man mit den Fachbegriffen umgeht. Lässt man sie in Englisch, weil sie bereits so in die deutsche Sprache eingeführt wurden? Oder übersetzt man sie, um die Begriffe verständlich zu machen? Wir haben in der deutschsprachigen DDD-Gemeinde eine Diskussion darüber geführt und sind zu folgendem Ergebnis gekommen: In der Praxis ist es – auch in Projekten, in denen Deutsch gesprochen wird – üblich, die DDD-Begriffe auf Englisch zu verwenden. Deshalb lassen wir sie auch hier im deutschen Text auf Englisch. Jeweils beim ersten Auftreten eines Begriffes geben wir eine Übersetzung auf Deutsch an, um die Begrifflichkeiten verständlich zu machen. Weil man ein Sachbuch nicht immer von Anfang bis Ende liest, sondern gerne auch mitten in den Text springt, machen wir das jeweils beim ersten Auftreten eines DDD-Begriffes in jedem Kapitel.

Außerdem finden Sie auf den Umschlaginnenseiten ein kleines Glossar, in dem Sie schnell nachschlagen können, wenn Sie sich z. B. fragen, was das »Bounded« in *Bounded Context* nochmal auf Deutsch heißt.

Für die lebhafte Diskussion »englische vs. deutsche Begriffe« und die tollen Ideen danken wir herzlich (in alphabetischer Reihenfolge): Sebastian Dietrich, Jan Fellien, Oliver Gierke, Marco Heimeshoff, Stefan Hofer, Michael Plöd, Stefan Priebsch, Lars Röwekamp und Eberhard Wolff.

*Carola Lilienthal, Henning Schwentner*
Hamburg, März 2017

# Vorwort

Warum macht das Bauen von Modellen so viel Spaß und warum ist es so eine befriedigende Tätigkeit? Schon als Kind habe ich es geliebt, Modelle zu bauen. Damals habe ich hauptsächlich Modelle von Autos und Flugzeugen angefertigt. Ich frage mich, wo Lego damals war. Im Leben meines Sohnes hat Lego jedenfalls eine große Rolle gespielt, schon seit er sehr klein war. Mit diesen kleinen Bausteinen sich Modelle auszudenken und zu bauen ist sehr faszinierend. Man kann mit kleinen Modellen anfangen, und es scheint, als könnte man seine Ideen nahezu unendlich erweitern.

Sie haben sicherlich auch das eine oder andere Modell in Ihrer Jugend gebaut.

Modelle kommen in unserem Leben in sehr vielen Situationen vor. Wenn Sie Spaß an Brettspielen haben, benutzen Sie Modelle. Wir haben Modelle von Immobilien und Grundbesitzern oder Modelle von Inseln und Schiffbrüchigen oder Modelle von Territorien und Baumaßnahmen oder von wer weiß was allem. Auch Videospiele sind Modelle. Vielleicht modellieren sie eine Fantasiewelt mit bunten Charakteren, die fantastische Rollen spielen. Kartenspiele und deren Varianten bilden Machtspiele als Modell ab. Wir benutzen die ganze Zeit Modelle und wahrscheinlich so oft, dass wir den meisten Modellen nicht die wohlverdiente Anerkennung geben. Modelle sind einfach Teil unseres Lebens.

Aber warum? Jeder Mensch hat einen Lernstil. Es gibt eine Vielzahl von Lernstilen, aber die drei am häufigsten diskutierten sind auditives, visuelles und taktiles Lernen. Das auditive Lernen basiert auf Hören und Zuhören. Die visuellen Lerner verstehen durch Lesen und Sehen von Bildern. Die taktilen Lerner erfassen Neues dadurch, dass sie während des Lernprozesses Dinge berühren. Es ist interessant, dass die verschiedenen Individuen jeweils einen bestimmten Lernstil bevorzugen, und zwar so stark, dass der Lernende Probleme mit den anderen Arten des Lernens hat. Zum Beispiel merken sich taktile Lerner wahrscheinlich, was sie getan haben, haben aber vielleicht Probleme, sich zu erinnern, was während des Prozesses gesagt wurde. Man würde annehmen, dass beim Modellbilden visuelle und taktile Lerner einen riesigen Vorteil gegenüber den auditiven Lernern hätten, weil das Bauen von Modellen vor allem visuelle und taktile Stimulationen zu enthalten scheint. Wenn ein Team von Modellentwicklern allerdings in seinem

Prozess der Modellbildung mündliche Kommunikation einsetzt, sieht es gleich ganz anders aus. Mit anderen Worten bietet das Bilden von Modellen die Möglichkeit, die Lernstile der meisten Individuen zu berücksichtigen.

Software beeinflusst und unterstützt unser Leben immer stärker. Wenn wir schon so eine angeborene Neigung dazu haben, durch das Bilden von Modellen zu lernen, wäre es da nicht nur natürlich, auch Software auf Basis eines Modells zu entwickeln? Tatsächlich scheint Software aus Modellen zu bauen irgendwie, nun ja, menschlich zu sein. Also sollten wir Modelle als Basis von Software verwenden. Mir scheint, dass Menschen von Natur aus ein Talent zum Bilden von Softwaremodellen haben.

Ich möchte Ihnen unbedingt helfen, beim Modellieren von Software diese angeborenen Talente so gut wie möglich zu nutzen, indem Sie einige der besten verfügbaren Softwaremodellierungstechniken verwenden können. Diese Techniken sind unter dem Namen »Domain-Driven Design« oder kurz DDD als ein Werkzeugkasten zusammengefasst. Dieser Werkzeugkasten, eigentlich eine Menge von Mustern, wurde zuerst von Eric Evans in seinem Buch *Domain-Driven Design: Tackling Complexity in the Heart of Software* [Evans 2004] aufgeschrieben. Meine Vision ist es, möglichst jedem DDD zugänglich zu machen. Um es zugespitzt zu formulieren: Ich will DDD den breiten Massen bekannt machen. DDD verdient es, überall bekannt zu sein. Es ist der Werkzeugkasten, den modellorientierte Menschen für ihre Arbeit verdient haben, um ihre fortschrittlichsten Softwaremodelle zu erschaffen. Mein Ziel ist es, mit diesem Buch das Lernen und Anwenden von DDD so einfach wie möglich zu machen und DDD dem größtmöglichen Fachpublikum näherzubringen.

Für auditive Lerner hält DDD die Aussicht bereit, durch die Kommunikation im Team das Entwickeln eines Modells auf Basis einer *Ubiquitous Language* (dt.: *allgegenwärtige Sprache*) zu lernen. Für visuelle und taktile Lerner können die DDD-Techniken sehr anschaulich und anfassbar eingesetzt werden, wenn das Team sowohl strategisch als auch taktisch modelliert. Das gilt insbesondere für das Zeichnen von *Context Maps* (dt.: *Kontextlandkarten*) und das Modellieren des Geschäftsprozesses mit *Event Storming*. Deshalb glaube ich, dass DDD jedem helfen kann, der durch das Bilden von Modellen lernen und Großartiges erreichen will.

## Für wen ist dieses Buch?

Dieses Buch ist für jeden, der daran interessiert ist, die wichtigsten DDD-Aspekte und -Techniken möglichst schnell zu lernen. Die typischen Leser sind Softwarearchitekten und Softwareentwickler, die DDD in Projekten in der Praxis einsetzen wollen. Softwareentwickler erkennen die Schönheit von DDD in der Regel sehr schnell und werden von seinen machtvollen Techniken angezogen. Aber ich habe das Thema auch schon Vorständen, Fachexperten, Managern, Business-Analysten, Informationsarchitekten und Testern vermittelt. Tatsächlich ist die Gruppe derer, die vom Lesen des Buches profitieren können, nicht auf die IT-Industrie und auf Forschungs- und Entwicklungsabteilungen beschränkt.

Wenn Sie Berater sind und mit einem Kunden arbeiten, dem Sie die Verwendung von DDD empfohlen haben, bietet dieses Buch eine Möglichkeit, die wichtigsten Beteiligten schnell mit Wissen zu versorgen. Wenn Sie Entwickler in Ihrem Projekt haben – vielleicht von Junior-, mittlerem oder sogar Seniorlevel –, die nicht vertraut mit DDD sind, aber es sehr bald verwenden müssen, stellen Sie sicher, dass sie dieses Buch lesen. Durch das Lesen dieses Buches werden alle Projektbeteiligten und Entwickler auf jeden Fall das Vokabular lernen und verstehen, wie man die grundsätzlichen DDD-Techniken einsetzt. Dies ermöglicht allen Beteiligten, sinnvoll Wissen auszutauschen, während sie das Projekt vorantreiben.

Was immer Ihre Erfahrung und Rolle ist, lesen Sie dieses Buch und verwenden Sie dann DDD in einem Projekt. Danach lesen Sie das Buch noch einmal und schauen, was Sie aus Ihren Erfahrungen lernen können und wo Sie sich in der Zukunft verbessern können.

## Was dieses Buch enthält

Das erste Kapitel, »DDD für mich«, erklärt, was DDD für Sie und Ihre Organisation tun kann, und bietet einen detaillierteren Überblick darüber, was Sie lernen werden und warum das wichtig ist.

Kapitel 2, »Strategisches Design mit Bounded Contexts und der Ubiquitous Language«, führt das strategische Design von DDD ein und lehrt die Eckpfeiler von DDD, *Bounded Contexts* (dt.: *begrenzte Kontexte*) und die *Ubiquitous Language*. Kapitel 3, »Strategisches Design mit Subdomains«, erklärt *Subdomains* (dt.: *Teildomänen*, *Subdomänen*) und wie man sie verwenden kann, um mit der Komplexität bei der Integration von existierenden Altsystemen umzugehen, während man seine neuen Anwendungen modelliert. Kapitel 4, »Strategisches Design mit Context Mapping«, stellt die verschiedenen Wege vor, mit denen Teams strategisch zusammenarbeiten können, und vermittelt Wege, wie die Software verschiedener Teams miteinander integriert werden kann. Das nennt man *Context Mapping* (dt.: *Abbilden von Kontexten*).

Kapitel 5, »Taktisches Design mit Aggregates«, lenkt Ihre Aufmerksamkeit auf das taktische Modellieren mit *Aggregates* (dt.: *Aggregate*). Ein wichtiges und mächtiges Werkzeug für taktisches Modellieren sind *Domain Events* (dt.: *Domänenereignisse, fachliche Ereignisse*), die das Thema von Kapitel 6, »Taktisches Design mit Domain Events«, sind.

In Kapitel 7, »Beschleunigungs- und Managementtechniken«, hebt das Buch schließlich einige Techniken für die Beschleunigung und das Management von Projekten hervor. Sie können den Teams helfen, eine hohe Schlagzahl (engl.: cadence) zu erreichen und zu erhalten. Diese beiden Themen sind in anderen DDD-Quellen bisher selten, wenn überhaupt, diskutiert und werden dringend benötigt, wenn man beschlossen hat, DDD in der Praxis umzusetzen.

## Konventionen

Es gibt nur wenige Konventionen, die man beim Lesen im Gedächtnis behalten sollte. Alle DDD-Begriffe, die ich verwende, sind kursiv gesetzt. Zum Beispiel werden Sie von *Bounded Contexts* und *Domain Events* lesen[1]. Eine weitere Konvention ist, dass sämtlicher Quellcode in nichtproportionaler Schrift gesetzt ist.

Was in diesem Buch besonders stark betont wird und was Ihrem Gehirn hoffentlich gefällt, ist visuelles Lernen durch zahlreiche Diagramme und Abbildungen. Sie werden bemerken, dass es keine Abbildungsnummern im Buch gibt, weil ich Sie nicht verwirren wollte. Die Abbildungen und Diagramme stehen immer vor dem Text, der sie erklärt, was bedeutet, dass die Grafiken die Gedanken einführen, während man sich durch das Buch arbeitet. Das heißt, immer wenn Sie Text lesen, können Sie die vorangehende Abbildung zur visuellen Unterstützung verwenden.

1. Anm. d. Übersetzer: Wie in der Vorbemerkung beschrieben, verwenden wir im deutschen Text – wie in der Praxis üblich – die englischen DDD-Begriffe. Jeweils beim ersten Auftreten eines Begriffes geben wir eine Übersetzung auf Deutsch, um den Begriff verständlich zu machen. Zum schnellen Nachschlagen gibt es das Glossar auf den Umschlaginnenseiten.

# Danksagung

Dies ist nun mein drittes Buch, das im hochgeschätzten Addison-Wesley-Verlag erscheint. Es ist auch das dritte Mal, dass ich mit meinem Lektor Chris Guzikowski und meinem entwicklungstechnischen Lektor Chris Zahn zusammenarbeite, und erfreulicherweise kann ich sagen, dass das dritte Mal genauso angenehm ist wie die beiden ersten Male. Nochmals vielen Dank für die Entscheidung, meine Bücher zu veröffentlichen.

Wie immer kann ein Buch nicht ohne kritisches Feedback erfolgreich geschrieben und veröffentlicht werden. Dieses Mal habe ich mich an eine Reihe von DDD-Anwendern gewandt, die DDD nicht unbedingt selbst lehren oder darüber schreiben, aber dennoch an Projekten arbeiten und anderen dabei helfen, diesen leistungsstarken Werkzeugkasten zu nutzen. Ich hatte das Gefühl, dass dieser Fokus auf DDD-Anwender entscheidend war, um sicherzustellen, dass dieser stark komprimierte Text genau das sagt, was notwendig ist, und es auf die richtige Weise sagt. Es ist so ähnlich wie: Wenn Sie wollen, dass ich für 60 Minuten spreche, geben Sie mir 5 Minuten und ich bin bereit; wenn Sie wollen, dass ich für 5 Minuten spreche, dann brauche ich mehrere Stunden, um mich vorzubereiten.

In alphabetischer Reihenfolge der Nachnamen haben mir am meisten geholfen: Jérémie Chassaing, Brian Dunlap, Yuji Kiriki, Tom Stockton, Tormod J. Varhaugvik, Daniel Westheide und Philip Windley. Vielen Dank!

# Inhaltsverzeichnis

**1 DDD für mich** **1**

Wird DDD wehtun? . . . . . 2
Gutes, schlechtes und effektives Design . . . . . 3
Strategisches Design . . . . . 7
Taktisches Design . . . . . 8
Lernprozess und Wissensvertiefung . . . . . 9
Legen wir los! . . . . . 9

**2 Strategisches Design mit Bounded Contexts und der Ubiquitous Language** **11**

Domain Experts und Geschäftstreiber . . . . . 17
Fallstudie . . . . . 20
Grundlegendes strategisches Design ist notwendig . . . . . 24
Infragestellen und Vereinheitlichen . . . . . 28
Eine Ubiquitous Language entwickeln . . . . . 34
Szenarios umsetzen . . . . . 37
Und auf lange Sicht? . . . . . 39
Architektur . . . . . 40
Zusammenfassung . . . . . 42

**3 Strategisches Design mit Subdomains** **43**

Was ist eine Subdomain? . . . . . 44
Arten von Subdomains . . . . . 44
Mit Komplexität umgehen . . . . . 45
Zusammenfassung . . . . . 48