Geleitwort von Vaughn Vernon

In meiner »Signature Series«[1] stehen organisches Wachstum und Weiterentwicklung im Vordergrund, die ich im Folgenden genauer beschreibe. Doch zuerst erzähle ich eine Geschichte vom organischen Wachstum, die mit diesem Buch verbunden ist.

Mein Buch *Implementing Domain-Driven Design*, auch bekannt als »das rote Buch«, erschien zu einem wichtigen Zeitpunkt. Vor dem roten Buch gab es nur eine Handvoll Menschen, die Domain-Driven Design (DDD), diesen fortschrittlichen Ansatz zur Softwareentwicklung, wirklich verstanden hatten – diejenigen, die man als Vorreiter bezeichnen könnte. Außerdem gab es zu dieser Zeit nur wenige DDD-Meetups und kaum hilfreiche Werkzeuge, die Praktiker bei der Anwendung von DDD unterstützten. Ich war Mitbegründer des DDD-Meetup in Denver im Jahr 2011, lange bevor mein rotes Buch 2013 veröffentlicht wurde. Damals gab es vielleicht insgesamt fünf Meetups zu diesem Thema. Im Jahr 2021 konnte man weltweit 142 Meetup-Gruppen zu DDD mit 93.171 Mitgliedern finden, und es werden immer mehr. Als zunächst 10, dann 20 und dann 25 Gruppen erreicht waren, hätte da jemand gedacht, dass es einmal fast 150 sein würden? Infolge dieses organischen Wachstums wuchs auch die Zahl der Vorreiter und der Werkzeuge für DDD. Timing ist alles, und ich freue mich, dass ich zu den richtigen Zeitpunkten Impulse geben konnte, die zur Verbreitung von DDD beigetragen haben. Bevor ich erkläre, wie dieses Buch und seine Autoren an diesem organischen Wachstum beteiligt sind, möchte ich zunächst auf die Reihe eingehen, in der es erscheint.

Meine »Signature Series« verhilft den Leserinnen und Lesern zu Fortschritten in der Reife der Softwareentwicklung und führt sie zur erfolgreichen Anwendung geschäftsorientierter Praktiken. Die Reihe legt den Schwerpunkt auf *organisches Wachstum und Weiterentwicklung* mit einer Vielzahl von Ansätzen – reaktive, objektorientierte und funktionale Architektur und Programmierung, Domänenmodellierung, Services in der richtigen Größe, Patterns und APIs – und behandelt die beste Nutzung der zugrunde liegenden Technologien.

1. Die englische Originalfassung dieses Buches ist in der Reihe »*The Pearson Addison-Wesley Signature Series*«, die von Vaughn Venon herausgeben wird, erschienen.

Ab hier konzentriere ich mich auf nur zwei Wörter: *organische Weiterentwicklung*.

Das erste Wort, *organisch*, benutzte ein Freund und Kollege kürzlich, um Softwarearchitektur zu beschreiben. Ich habe das Wort *organisch* schon oft im Zusammenhang mit Softwareentwicklung gehört und verwendet, aber ich habe nicht so genau darüber nachgedacht, bis ich die beiden Wörter zusammen gehört habe: *organische Architektur*.

Man denke an die Wörter *organisch* und *Organismus*. Meistens werden sie im Zusammenhang mit Lebewesen verwendet, aber auch, um unbelebte Dinge zu beschreiben, die in einigen Eigenschaften Lebewesen ähneln. Der Begriff *Organismus* stammt aus dem Griechischen. Etymologisch bezieht er sich auf einen Teil eines lebendigen Ganzen, hat aber noch weitere Bedeutungen: ein Mittel zur Herstellung von etwas im Sinne von Gerät, (Musik-)Instrument oder Werkzeug.

Es fällt uns nicht schwer, uns Beispiele für organische Objekte im Sinne von lebenden Organismen vorzustellen von den ganz großen bis zu mikroskopisch kleinen, einzelligen Lebensformen. Bei der zweiten Verwendung von Organismus müssen wir schon etwas länger nachdenken. Ein Beispiel ist eine Organisation, die mit dem gleichen Präfix wie *organisch* und *Organismus* beginnt. In dieser Bedeutung beschreibt *Organismus* etwas mit bidirektionalen Abhängigkeiten: Eine Organisation ist ein Organismus, weil sie organisierte Teile hat. Diese Art von Organismus kann ohne die Teile nicht überleben, und die Teile können ohne den Organismus nicht überleben.

Diese Denkweise können wir auch auf unbelebte Dinge anwenden, denn sie weisen Eigenschaften von lebenden Organismen auf. Nehmen wir z. B. das Atom. Jedes einzelne Atom ist ein System für sich, und alle Lebewesen bestehen aus Atomen. Die Atome selbst sind jedoch anorganisch und vermehren sich nicht. Trotzdem kann man sich Atome als Lebewesen in dem Sinne vorstellen, dass sie sich ständig bewegen und aktiv sind. Atome gehen sogar Verbindungen mit anderen Atomen ein. Wenn dies geschieht, ist jedes Atom nicht nur ein einzelnes System für sich, sondern wird zusammen mit anderen Systemen auch zu einem Subsystem. Zusammen mit anderen solchen Subsystemen ergibt deren gemeinsames Verhalten ein größeres Gesamtsystem.

Alle Konzepte, die sich auf Software beziehen, sind also insofern organisch, als nicht lebende Dinge durch Aspekte lebender Organismen »charakterisiert« werden. Wenn wir softwaretechnische Modelle anhand konkreter Szenarien diskutieren, ein Architekturdiagramm zeichnen oder einen Unit Test und die dazugehörige fachliche »Unit« programmieren, beginnt die Software lebendig zu werden. Sie ist nicht statisch, denn wir diskutieren immer wieder über Verbesserungen und entwickeln sie weiter, wobei ein Szenario zum nächsten führt, was sich wiederum auf die Architektur und das Domänenmodell auswirkt. Wenn wir weiter iterieren, führt der steigende Wert der Weiterentwicklungen zu einem inkrementellen Wachstum des Organismus. Mit der Zeit entwickelt sich die Software weiter. Wir kämpfen mit der Komplexität und bewältigen sie durch nützliche Abstraktionen, und die Software wächst und verändert ihre Form, alles mit dem ausdrücklichen Ziel, die Arbeit für echte, lebende Organismen auf globaler Ebene zu verbessern.

Leider tendieren Softwareorganismen dazu, eher schlecht als gut zu wachsen. Selbst wenn sie ihr Leben bei guter Gesundheit beginnen, neigen sie dazu, Krankheiten zu bekommen, sich zu verformen, unnatürliche Auswüchse zu entwickeln, zu verkümmern und zu verfallen. Noch schlimmer ist, dass diese Symptome durch gut gemeintes, aber schiefgegangenes Verfeinern verursacht werden. Das Schlimmste daran ist, dass die fehlgeschlagene Weiterentwicklung nicht zum Tod dieser auf komplexe Art kranken Softwareorganismen führt. (Oh, wenn sie doch nur sterben könnten!) Stattdessen müssen wir sie töten, und das erfordert Nerven, Geschick und das Durchhaltevermögen eines Drachentöters. Nein, nicht eines, sondern Dutzender von kräftigen Drachentötern. Oder besser gesagt: Dutzende von Drachentötern mit richtig viel Verstand.

Genau hier kommt diese Buchreihe ins Spiel. Sie soll ihnen dabei helfen, sich weiterzuentwickeln und mit verschiedenen Ansätzen – reaktive, objektorientierte und funktionale Architektur und Programmierung, Domänenmodellierung, Services in der richtigen Größe, Muster und APIs – erfolgreich zu sein. Außerdem deckt die Buchreihe die optimale Nutzung der zugrunde liegenden Technologien ab. Das ist nicht mit einem Schlag erledigt. Es erfordert eine organische Weiterentwicklung mit Engagement und Geschick. Die anderen Autoren und ich sind da, um zu helfen. Zu diesem Zweck haben wir unser Bestes gegeben, um unser Ziel zu erreichen.

Deshalb habe ich dieses Buch, *Domain Storytelling*, ausgewählt, um es in meine Signature Series aufzunehmen. Die bereits erwähnte organische Ausbreitung von DDD hat zu neuen Praktikern und Vorreitern sowie zu Innovationen bei den Werkzeugen geführt, die DDD unterstützen. Collaborative Modeling mit diesem brillanten Werkzeug ermöglicht eine visuelle Exploration von Domänen, fachliches Wissen zu entdecken und Nutzungsszenarien mit der nötigen Klarheit zu modellieren. Domain Storytelling sollte nicht als Ersatz für bisherige Werkzeuge gesehen werden, sondern als Chance, den Werkzeugkasten der Wissensgewinnung zu erweitern. Neuartige und anspruchsvollere Modellierungssituationen erfordern eine Reihe nützlicher Werkzeuge, die gemeinsam eingesetzt werden können.

Mit diesem neuen Buch etablieren sich Stefan Hofer und Henning Schwentner als zwei der neuen Vorreiter. Sie haben uns mit dem Domain Storytelling ein zusätzliches Werkzeug an die Hand gegeben, mit dem wir uns mit den komplexeren Problemen auseinandersetzen können, mit denen wir heute konfrontiert sind und in den nächsten Jahren konfrontiert bleiben werden. Das ist organisch.

Vaughn Vernon

Geleitwort von Nick Tune

Im Jahr 2004 veröffentlichte Eric Evans das zeitlose Buch *Domain-Driven Design*, das zu einem Klassiker der Softwareentwicklungsliteratur geworden ist. Evans entwarf darin seine Vision von Softwareentwicklern, die eng mit Fachexperten zusammenarbeiten, um iterativ fachliche Probleme für die Nutzer zu lösen. Damals grenzte dies an Ketzerei gegenüber den gängigen Praktiken, die orientiert waren an Datenmodellen und umfangreicher Vorausplanung und in denen Programmierer als bloße Befehlsempfänger gesehen wurden.

Der Text von Evans war ein Meisterwerk, aber trotzdem fehlte ihm etwas. Ein Jahrzehnt lang wurde DDD vom Mainstream als eine Sammlung von Entwurfsmustern wahrgenommen und als Over-Engineering abgestempelt. In Evans' Buch war häufig die Rede davon, dass Fachexperten und technische Experten gemeinsam Domänenwissen erarbeiten. Anders als bei den Entwurfsmustern gab das Buch den Leserinnen und Lesern jedoch nicht genügend Anleitung, wie man die Zusammenarbeit praktisch umsetzt.

Mitte der 2010er-Jahre begann eine DDD-Renaissance, die bis heute anhält. Das Buch *Implementing Domain-Driven Design* von Vaughn Vernon trug entscheidend dazu bei, viele Missverständnisse zu korrigieren und DDD zugänglicher zu machen. Und eine neue Generation von Praktikern, angeführt von Alberto Brandolini und zu der auch Stefan und Henning gehören, fügte das fehlende Teil des DDD-Puzzles hinzu, indem sie neue kollaborative Modellierungstechniken in die Community einführte. Die Mainstream-Wahrnehmung von DDD ist heute genauso sehr »Klebezettel an der Wand« wie die Implementierung von Entwurfsmustern. Evans' Prophezeiung aus dem Jahr 2004 ist nun Realität.

Domain Storytelling zeichnet sich durch seinen piktografischen, strukturierten und szenariobasierten Charakter aus. Aber dieses Buch ist weit mehr als ein Leitfaden für Domain Storytelling. Stefan und Henning sind leidenschaftliche, intelligente und erfahrene Domänenmodellierer. Dieses Buch nimmt Sie mit in ihre Denkmuster und tief in die Prinzipien des Collaborative Modeling und der Workshop-Moderation. Dieses Werk wird Ihnen nützen – unabhängig von der Technik, für die Sie sich entscheiden, und unabhängig davon, wie viel Sie über DDD wissen. Möglicherweise inspiriert es Sie sogar dazu, die nächste Generation von Collaborative-Modeling-Techniken zu erfinden.

Ich hatte das Glück, Stefan und Henning auf verschiedenen Konferenzen zu treffen. Eine ist mir besonders in Erinnerung geblieben: Explore DDD 2018 in Denver. Durch ihren Enthusiasmus für Domain Storytelling fesselten sie die Zuhörer (mich eingeschlossen) mit ihrem Vortrag. Sie stellten eine Fallstudie aus dem Hamburger Hafen vor. Das i-Tüpfelchen war ihr Rollenspiel mit Requisiten, in dem sie den Domain-Storytelling-Workshop nachspielten. Ich durfte auch an ihrem Workshop teilnehmen, bei dem ihre Leidenschaft die Teilnehmenden (mich eingeschlossen) dazu brachte, das Erlebnis »Kino« mit Begeisterung als Domain Story zu modellieren.

Ich erinnere mich auch gerne an den Abend vor der Hauptkonferenz. Eric Evans hielt am Abend eine Keynote zum offiziellen Start der Explore DDD, und ich war mit Stefan und Henning beim anschließenden Social Event. Sie bezogen die anderen Teilnehmerinnen und Teilnehmer in Diskussionen ein, regten Gespräche an und sie brachten uns alle mit ihrem cleveren Humor zum herzhaften Lachen.

Ich hoffe, dieses Buch vermittelt allen Leserinnen und Lesern die Inspiration, den Enthusiasmus und das Lächeln, das Stefan und Henning mir und vielen anderen geschenkt haben.

Nick Tune