Domain Storytelling

Gemeinschaftlich, visuell und agil zu fachlich wertvoller Software

DAS VORWORT

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

Vorwort

Missverständnisse zwischen Softwareentwicklern und Personen aus Fachabteilungen sind ein häufiges Problem. Schlechte Kommunikation ist eine Plage, die Projekte scheitern lässt. Domain Storytelling bietet Abhilfe, denn diese Technik verwandelt Domänenwissen in effektive Unternehmenssoftware. Domain Storytelling bringt Fachexperten, Softwareentwickler, User Experience Designer, Product Owner, Produktmanager und Business-Analysten zusammen. Sie lernen voneinander, indem sie Geschichten erzählen und sie in leicht verständlichen Bildern darstellen.

Wer sollte dieses Buch lesen?

Das Buch richtet sich an alle, die an der Softwareentwicklung beteiligt oder daran interessiert sind, auch an nicht technische Leserinnen und Leser. Es gibt nur wenige Codebeispiele und der Großteil dieses Buches ist ohne Vorkenntnisse in Softwareentwicklung verständlich. Wo es sinnvoll ist, verweisen wir auf empfohlene Lektüre.

Neben Fachleuten aus der Softwareentwicklung ist das Buch auch für Führungskräfte, Abteilungs- und Teamleiter sowie für Fachexperten relevant. Sie können das Buch nutzen, um ihre Geschäftsprozesse besser zu verstehen und die Kommunikation mit ihren (IT-)Kolleginnen und Kollegen zu verbessern.

Dieses Buch in der Softwareentwicklungslandschaft

Dieses Buch ist als praktischer Leitfaden gedacht. Wir wollten es kurz halten und mussten der Versuchung widerstehen, ein Buch über unsere Sicht auf Softwareentwicklung im Allgemeinen zu schreiben. Themen wie Domain-Driven Design, fachliche Modellierung, Anforderungen und Agilität sind für Domain Storytelling relevant und werden in diesem Buch erwähnt. Wir haben versucht, nicht zu tief in diese Themen einzutauchen, weil andere Autoren das viel besser gemacht haben, als wir es könnten. Wer neugierig geworden ist, findet hier eine (unvollständige) Liste von Autorinnen und Autoren, die uns beeinflusst haben:

Heinz Züllighoven et al.
Der *Werkzeug & Material-Ansatz* beschrieben im *Object-Oriented Construction Handbook* [Züllighoven 2004] ist wichtig für unsere Sichtweise der Softwareentwicklung. Er fördert die anwenderorientierte Softwareentwicklung und basiert auf den starken Metaphern *Werkzeug* und *Material*. Das Buch wurde erstmals in den späten 1990er-Jahren unter dem Originaltitel *Das objektorientierte Konstruktionshandbuch nach dem Werkzeug & Material-Ansatz* veröffentlicht und behandelt Themen wie iterativ-inkrementelle Entwicklung, Modellierung, Szenarien und Muster für die Softwarekonstruktion.

Eric Evans
Domain-Driven Design (*DDD*) [Evans 2004] teilt viele Merkmale mit dem Werkzeug & Material-Ansatz, obwohl die beiden unabhängig voneinander entwickelt wurden. Die Konzepte *Bounded Contexts* und *Ubiquitous Language* haben uns sofort überzeugt. DDD eröffnete einen neuen Anwendungsbereich für Domain Storytelling und beeinflusste die Richtung, in die wir die Methode in den letzten Jahren vorangetrieben haben.

Kent Beck et al.
Das *Agile Manifest* [Beck et al. 2001] schätzt »Individuen und Interaktionen mehr als Prozesse und Werkzeuge« und »funktionierende Software mehr als umfassende Dokumentation«. Bei Domain Storytelling geht es um Individuen und ihre Interaktionen auf zwei Ebenen: (1) Domain Stories stellen Menschen und ihre Zusammenarbeit vor, und (2) der Workshop, in dem diese Domain Stories erzählt werden, bringt Menschen zusammen, um miteinander zu interagieren. *Extreme Programming* (*XP*) brachte das Konzept der Geschichten in die Softwareentwicklung und führte *User Stories* in die agile Welt ein [Beck 2000], [Beck & Andres 2005] und [Cohn 2004].

Alistair Cockburn
Writing Effective Use Cases [Cockburn 2001] ist eines unserer Lieblingsbücher über Anforderungen. Selbst wenn man keine Use Cases anwendet, ist das Buch sehr lesenswert. Es zeigt, wie man einen agilen Pfad von der Domäne bis zur Software beschreitet. Wir haben die Idee der *Goal Levels* entlehnt und sie auf Domain Storytelling angewendet.

Gojko Adzic
Specification by Example [Adzic 2011] hebt die Bedeutung von Gesprächen in der Softwareentwicklung hervor. Wir betrachten Softwareentwicklung als eine Reihe von Gesprächen über die Domäne und die Anforderungen, unterstützt durch verschiedene »Gesprächsmethoden« wie Domain Storytelling. Wir teilen Adzics Ansichten über Anforderungen und die Vorteile der Verwendung von Beispielen.

Christiane Floyd
Wir hatten das Vergnügen, von Professorin Floyd an der Universität Hamburg zu lernen. Sie forschte und lehrte (unter anderem) zu Modellierungstheorie, den Grenzen der Modellierung, soziotechnischen Aspekten der Modellierung und Softwareent-

wicklung und partizipativem Design. Wer neugierig ist, sollte sich z. B. »Software Development as Reality Construction« [Floyd 1992] ansehen und nach Abhandlungen über ihren *STEPS*-Ansatz suchen, ein Vorreiter für das, was später als agile Softwareentwicklung bekannt wurde.

Was dieses Buch behandelt

Dieses Buch ist in zwei Teile gegliedert: Teil I erklärt die Methode, und Teil II beschreibt, wie Domain Storytelling für bestimmte Einsatzzwecke genutzt und angepasst wird.

Teil I, »Domain Storytelling – die Methode«, vermittelt alles, was Sie wissen müssen, um mit Domain Storytelling etwas über eine Domäne zu lernen.

Kapitel 1, »Einführung«, erklärt, was Domain Storytelling eigentlich ist. Außerdem stellen wir eine Fallstudie vor, die einen ersten Eindruck von der Methode bietet und darlegt, warum sie nützlich ist.

Kapitel 2, »Die Bildsprache«, erläutert die grafische Notation. Um Domain Stories visuell festzuhalten, sind mehrere Icons und auch Regeln erforderlich, um diese zu kombinieren. Wir zeigen, was wir für guten Sprachstil halten und welche Fallstricke zu vermeiden sind. Wer noch nie Domain Storytelling ausprobiert hat, sollte das nach Kapitel 2 tun. Dazu nimmt man ein Blatt Papier und einen Stift zur Hand und modelliert einen Arbeitsablauf, mit dem man vertraut ist. Bevor Sie Domain Storytelling jedoch in einer Workshop-Situation ausprobieren, sollten Sie den Rest von Teil I lesen.

Kapitel 3, »Szenariobasiertes Modellieren«, behandelt einen der Hauptunterschiede zwischen Domain Stories und anderen Sprachen zur Geschäftsprozessmodellierung: In jeder Domain Story geht es um *einen* Fall. Wir zeigen, welche Fälle man modellieren sollte und wie man den Überblick behält.

Kapitel 4, »Scope«, zeigt, wie detailliert die Domain Stories sind, ob sie beschreibend oder explorativ sind und wie viele technische Informationen sie enthalten. All diese Faktoren müssen Sie mit jeder neuen Domain Story berücksichtigen. Dieses Kapitel hilft dabei, den richtigen Scope zu wählen.

Kapitel 5, »Modellierungswerkzeuge«, fasst unsere Erfahrungen mit verschiedenen Modellierungswerkzeugen zusammen. Wir empfehlen, welche Werkzeuge in welchen Situationen nützlich sind, beschreiben ihre Stärken und Schwächen und geben praktische Tipps für müheloses Modellieren.

Kapitel 6, »Das Workshop-Format«, zeigt, dass Domain Storytelling am besten funktioniert, wenn es für Collaborative Modeling (also gemeinschaftliches Modellieren von Fachleuten und Entwicklungsteam) eingesetzt wird. Hier lernen Sie, wie Sie einen Workshop vorbereiten, durchführen und nachbereiten. Dieses Kapitel hilft Ihnen dabei, eine gute Moderatorin oder ein guter Moderator zu werden.

Kapitel 7, »Das Verhältnis zu anderen Modellierungsmethoden«, behandelt andere Modellierungsmethoden und Workshop-Formate und wie Domain Storytelling mit ihnen kombiniert werden kann. Dieses Kapitel hilft Ihnen, das richtige Werkzeug für die richtige Aufgabe auszuwählen.

Teil II, »Domain Storytelling für verschiedene Einsatzzwecke nutzen und anpassen«, beschäftigt sich mit den verschiedenen Problembereichen, für die Domain Storytelling genutzt werden kann. Auch wenn wir in allen Kapiteln von Teil II dasselbe Beispiel verwenden, müssen die Kapitel nicht der Reihe nach gelesen werden. Suchen Sie sich die für Sie relevanten Einsatzzwecke aus.

Kapitel 8, *»Fallstudie – Alphorn Auto Leasing GmbH«*, stellt eine zweite, umfangreichere Fallstudie vor.

Kapitel 9, *»Fachsprache lernen«*, zeigt auf, dass die Sprache der Fachexperten zu sprechen der Schlüssel zu effektiven Gesprächen über Geschäftsprozesse und Anforderungen ist. Dieses Kapitel ist interessant, wenn man neu in einer Domäne ist, wenn man in der Software, an der man arbeitet, keine echten Begriffe aus der Domäne verwendet und das ändern will, oder wenn man in einem Unternehmen arbeitet, in dem es keine echte Fachsprache gibt und gewünscht ist, dass sich eine solche entwickelt.

Kapitel 10, *»Grenzen finden«*, beschäftigt sich damit, dass viele Domänen zu groß sind, um als Ganzes verstanden und modelliert werden zu können. Eine Domäne muss in überschaubare Einheiten aufgeteilt werden. Dieses Kapitel ist interessant, wenn Sie mit einem Monolithen zu kämpfen haben und ihn umstrukturieren oder zerlegen möchten oder Microservices entwerfen. Im Domain-Driven Design hilft dieses Kapitel, Bounded Contexts zu identifizieren. Das Kapitel ist auch nützlich, wenn das Entwicklungsteam zu groß geworden ist, um effizient zu arbeiten, oder wenn Sie bereits mehr als ein Entwicklungsteam haben und die Arbeitsaufteilung für diese Teams besser organisieren möchten.

Kapitel 11, *»Mit Anforderungen arbeiten«*, zeigt, wie Sie die Kluft zwischen Domänenwissen und Anforderungen überbrücken. Wir erklären, wie Anforderungen aus Domain Stories abgeleitet werden, um über Prioritäten und tragfähige Produkte diskutieren zu können. Dieses Kapitel ist interessant für Rollen wie Product Owner, Produktmanager, Business-Analyst, Requirements Engineer oder Entwickler in einem cross-funktionalen Team, das seine eigenen Anforderungen analysiert.

Kapitel 12, *»Modellieren in Code«*, beschäftigt sich mit dem Übergang von der Modellierung mit Diagrammen und Klebezetteln zur Modellierung in Programmiersprachen, um Software zu entwickeln. In diesem Kapitel wird vorgestellt, wie man von der visuellen Modellierung zum Code übergeht.

Kapitel 13, *»Organisatorische Veränderungen unterstützen«*, befasst sich mit dem Ziel eines neuen Softwaresystems, das in der Regel darin besteht, die Arbeit einfacher, schneller und effizienter (oder kurz: besser) zu machen. Dieses Ziel wird nicht dadurch erreicht, dass schlechte manuelle Prozesse digitalisiert werden. Genauso wenig wird aus einem Haufen von Anforderungen auf magische Weise ein reibungslos funktionierender Geschäftsprozess. Um gute Unternehmenssoftware zu entwickeln, ist mehr zu tun, als nur den aktuellen Zustand in Software zu implementieren. Man muss auch die zukünftige Arbeitsweise entwerfen. Domain Stories helfen dabei und veranschaulichen, wie die neue Software die Arbeitsweise der Menschen verändern wird. Dieses Kapitel ist interessant, wenn Sie Geschäftsprozesse optimieren, neue Software einführen oder Veränderungen in Geschäftsprozessen diskutieren und vorantreiben wollen.

Kapitel 14, *»›Make or Buy‹-Entscheidungen und Auswahl von Standardsoftware«*, weist darauf hin, dass nicht jede Software individuell entwickelt werden muss. Für viele Domänen gibt es Software von der Stange. Domain Stories können bei der Entscheidung helfen, ob ein neues Softwaresystem entwickelt oder gekauft werden soll. Wenn ein Kauf infrage kommt, muss man sich häufig zwischen Produkten mehrerer Anbieter entscheiden. Auch hier können Domain Stories hilfreich sein, um eine Auswahl zu treffen.

Kapitel 15, *»Schatten-IT finden«*, zeigt auf, dass die Schatten-IT im Weg steht, wenn es darum geht, IT-Landschaften zu konsolidieren oder die Digitalisierung voranzutreiben. Jedes Unternehmen ab einer bestimmten Größe nutzt Software, die der zentralen IT-Abteilung nicht bekannt ist. Diese kleinen Lösungen, die in den Fachabteilungen laufen und wenig beachtet werden, sind manchmal sogar geschäftskritisch. Fachexperten nutzen oft Schatten-IT, ohne sich dessen bewusst zu sein, sodass sie leicht übersehen wird. Domain Stories können der IT-Abteilung und dem Management helfen, diese Schatten-IT zu finden und die gesamte IT-Landschaft zu überblicken.

Kapitel 16, *»Ausblick und Fazit«*, wirft einen Blick in die Zukunft von Domain Storytelling und fasst seine Essenz zusammen.

Konventionen

Wenn wir einen neuen Begriff definieren, setzen wir ihn fett, z.B. **Akteur**. Begriffe, die von anderen Autoren definiert wurden, werden kursiv gesetzt, wenn wir sie zum ersten Mal verwenden, z.B. *Bounded Contexts*. Scope-Faktoren erscheinen in Kapitälchen, z.B. GROBGRANULAR. Wörter aus Fallstudien werden in Anführungszeichen gesetzt, z.B. »Kinobesucher«. Code wird in fester Breite gesetzt, wie die Klasse `Vorstellung`.

Wichtige Hinweise sind mit einem Glühbirnen-Icon gekennzeichnet:  . Wenn wir konkrete Ratschläge geben, sind unsere Tipps mit einem Häkchen-Icon  versehen.

Wir präsentieren zwei Fallstudien und mehrere »Geschichten aus der Praxis«. Abschnitte zur Fallstudie Alphorn Auto Leasing GmbH sind durch ein Icon mit einem Auto gekennzeichnet:  . Die Geschichten aus der Praxis werden in Einschüben erzählt, die mit einem Sonnenblumen-Icon  gekennzeichnet sind.

Legende für die »Opening Stories«

Als Modellierer konnten wir der Versuchung nicht widerstehen, Domain Storytelling selbst mit Domain Stories zu beschreiben. Deshalb beginnen mehrere Kapitel in diesem Buch mit einer »Opening Story« – einer kurzen Domain Story, die erklärt, worum es in dem Kapitel geht. Die Legende in Tabelle 1 soll helfen, die Icons zu deuten, die wir in diesen Geschichten verwenden.

Icon	Bedeutung
	Eine Domain Story als Diagramm
	Jede andere Art von Diagramm
	Bausteine einer Domain Story
	Aktivitäten einer Domain Story
	Ein Text wie eine User Story oder eine Quellcodedatei o. Ä.

Tab. 1 *In »Opening Stories« verwendete Icons*

Ergänzende Materialien

Auf *www.domainstorytelling.org/book* haben wir eine begleitende Website für dieses Buch erstellt [DomainStorytelling BookWebsite]. Sie enthält die Domain Stories, die wir in diesem Buch vorstellen, auf Deutsch und Englisch. Die Beispiele wurden mit Egon.io erstellt, einem Open-Source-Modellierungstool, das unsere Firma WPS entwickelt hat [Egon.io Website]. Man kann die Quelldateien aus diesem Buch in Egon.io importieren und die Domain Stories »abspielen« (Anweisungen finden sich auf der Website).

Die begleitende Website enthält auch die Bibliografie dieses Buches mit anklickbaren Links zu Online-Ressourcen.

Außerdem finden Sie auf *www.domainstorytelling.org* Links zu nützlichen Materialien, darunter eine Sammlung von Videos und Artikeln [DomainStorytelling Website].

Über das Cover

Die englische Ausgabe dieses Buches ist in der Addison-Wesley Signature Series von Vaughn Vernon erschienen. Die Titelbilder der Bücher dieser Buchreihe haben ein organisches Thema. Wir haben uns darüber gefreut, denn auch Domain Stories entwickeln sich auf organische Weise. Wie die Sonnenblumen, die auf dem Cover zu sehen sind, fängt eine Domain Story aus einem kleinen Samen an, wächst heran und blüht, wenn alles gut geht, schließlich wunderschön auf. Die Beziehung zwischen Domäne und Domain Story ist ähnlich wie die zwischen Sonne und Sonnenblume:

> Eine Sonnenblume kann ohne Sonne nicht existieren. Genauso wird eine Domain Story ohne den Kontakt zur Domäne verkümmern.
>
> Die Sonnenblume ähnelt der Sonne, so wie eine Domain Story ein Abbild der Domäne ist.

Während des Tages dreht die Sonnenblume ihren Kopf, um der Sonne zu folgen. Genauso folgt die Domain Story der Domäne.

Eine Sonnenblume lebt oft zusammen mit vielen anderen Sonnenblumen auf einem Feld. Genauso steht eine Domain Story normalerweise nicht allein, sondern in Gesellschaft anderer Stories.

All diese Eigenschaften machen die Sonnenblume nicht nur zu einem optisch ansprechenden Titelbild, sondern auch zu einem Symbol für Domain Storytelling.

In diesem Sinn wünschen wir eine anregende Lektüre des *Sonnenblumenbuches* und viel Spaß beim Modellieren!

Stefan Hofer und *Henning Schwentner*
Hamburg, im März 2023
www.domainstorytelling.org