3 Collaborative Modeling

In den letzten Jahren hat eine Gruppe von neuen Workshop-Formaten in der DDD- und der agilen Gemeinde viel Aufmerksamkeit und eine immer weitere Verbreitung gefunden: Methoden, die unter dem Label **Collaborative Modeling** (kurz auch **CoMo**) zusammengefasst werden.

Collaborative Modeling steht hinter vielen erfolgreichen Methoden, die im Problemraum auf die »Anforderungsermittlung« ausgerichtet sind (s. Abb. 3–1).

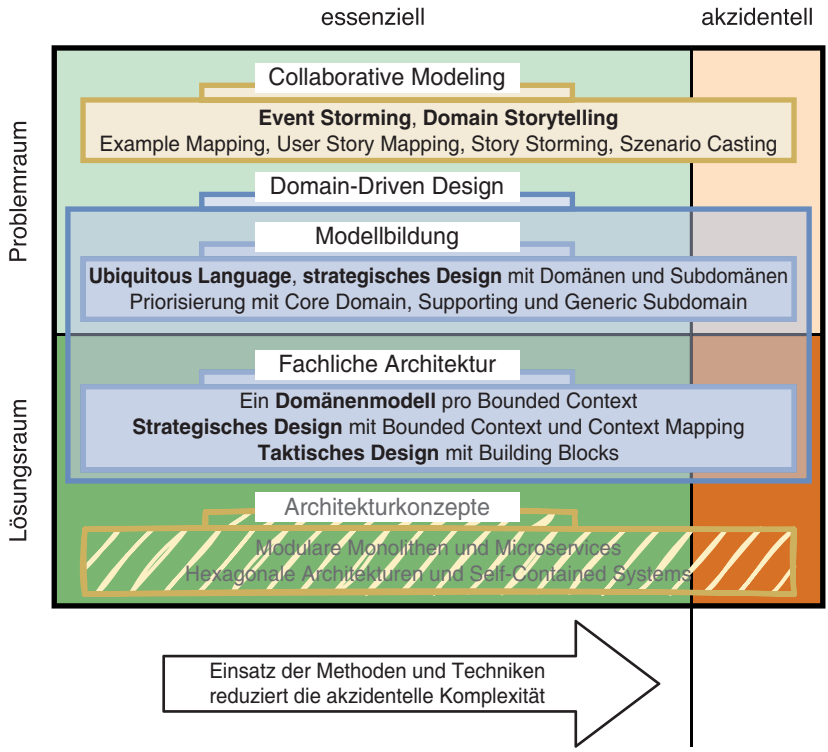

Abb. 3–1 *Komplexität und CoMo*

In Abbildung 3–1 sieht man die folgenden aktuell in der Community eingesetzten CoMo-Methoden:

- Event Storming [Brandolini 2021]
- Domain Storytelling [Hofer & Schwentner 2023]
- User Story Mapping [Patton 2014]
- Example Mapping [Wynne 2015]
- Storystorming [Schimak 2019]
- Scenario Casting [Koch 2023]

Event Storming und **Domain Storytelling** sind in Abbildung 3–1 fett unterlegt, weil wir sie in unserer Transformationsarbeit am häufigsten einsetzen.

Das Besondere an Collaborative Modeling ist, dass diese Methoden auf leichtgewichtige Art, nämlich nur mit Stift und (Klebe-)Zetteln und vielleicht einem Whiteboard, allen Beteiligten die direkte Teilhabe an der Anforderungsermittlung ermöglichen (vgl. [Floyd 1993]). Fachexperten und Anwender haben gleichermaßen Zugang zum Modell, denn sie erstellen es gemeinsam und nutzen es später auch zusammen. Die Einstiegshürde ist für Fachexperten und Anwender niedrig, weil Notationen verwendet werden, die in fünf Minuten erklärt sind. Das Schöne an den verschiedenen CoMo-Methoden ist, dass sie alle auch für IT-ferne Personen gut verständlich sind (s. Abschnitt 3.3).

Durch die Leichtgewichtigkeit bekommt man mit diesen Methoden schnell Antworten auf beispielsweise die folgenden Fragen:

- Was ist in einer Domäne eigentlich wirklich los?
- Welche Rollen haben die verschiedenen Fachexperten?
- In welchen Begriffen sprechen die Fachexperten?
- Wie arbeiten die Fachexperten und welche fachlichen Prozesse führen sie dabei durch?

Um neue Software zu bauen und alte zu verstehen und zu verbessern, müssen die IT-Experten die Domäne, also die Aufgaben und Arbeitsabläufe der Fachexperten, durchschauen. Umgekehrt sollen Fachexperten verstehen, welche Möglichkeiten Software eröffnet und wie sich Digitalisierung auf ihre tägliche Arbeit auswirken wird. Entwickler und Fachexperten benötigen eine *gemeinsame Sprache* (vgl. Abschnitt 2.2.1), um über die Domäne und die daraus erwachsenden Anforderungen an die Software sprechen zu können. Nur so wird die Software das widerspiegeln, was die Entwickler von den Anforderungen aus der Domäne verstanden haben und was die Fachexperten von einer Digitalisierung der fachlichen Prozesse erwarten.

3.1 Die richtigen Leute zusammenbringen

Mit Collaborative Modeling bringen wir die Fachleute und die technischen Menschen zusammen und sorgen dafür, dass sie sich direkt austauschen:

- Diejenigen, die etwas lernen wollen (typischerweise die Entwickler), und
- die Wissensträger (typischerweise Anwender, Domänenexperten, Business-Analysten).

CoMo bringt das Fachwissen aus den Köpfen der Fachexperten in die Köpfe von Entwicklern, Testern, Product Ownern, Produktmanagern und Business-Analysten – zu jedem, der an der Softwareentwicklung beteiligt ist. Im Endeffekt unterstützt Collaborative Modeling durch direkte Kommunikation und gemeinsames Modellieren den Lernprozess, der für das Verständnis einer Domäne unerlässlich ist (s. Abb. 3–2).

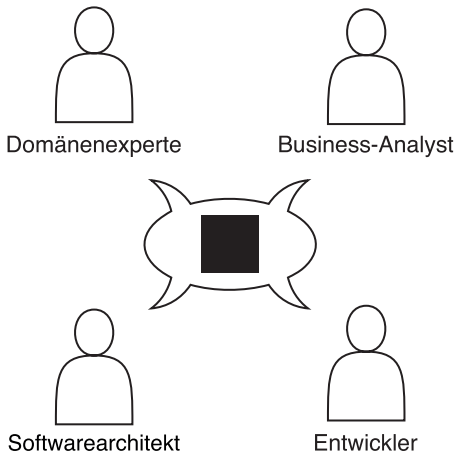

Abb. 3–2 *Collaborative Modeling: Direkte Kommunikation führt zu gemeinsamem Verständnis.*

Bei klassischen Requirements-Engineering-Methoden werden Interviews geführt und im Anschluss Anforderungen daraus abgeleitet. Im Gegensatz zu den CoMo-Methoden sitzen die verschiedenen Beteiligten, die alle ihre eigene Sicht auf die Domäne und/oder die eingesetzte Software haben, beim klassischen Requirements Engineering nicht gemeinsam in einem Workshop. Wird kein gemeinsames Verständnis in großer Runde erzielt, dann entstehen oft Stille-Post-Effekte, die zu vielfältigen Missverständnissen führen (s. Abb. 3–3).

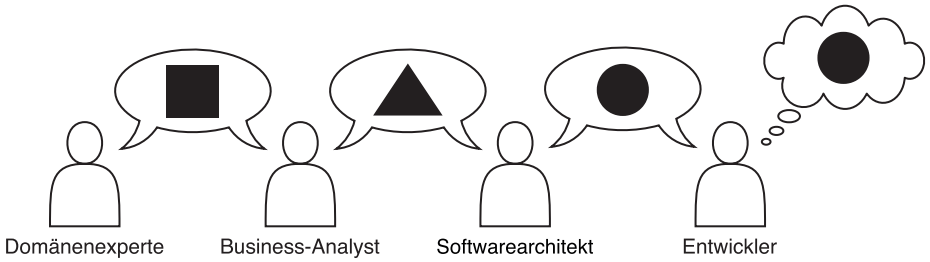

Abb. 3–3 ***Kein*** *Collaborative Modeling: Stille Post führt zu Missverständnissen.*

In den durchaus unterschiedlichen CoMo-Methoden lassen sich ähnliche Konzepte entdecken, die den Geist von CoMo, in offenen und respektvollen Diskussionen durch geschickte Fragestellung und das Kreuzen der verschiedenen Perspektiven auf dasselbe Problem iterativ neue Erkenntnisse zu gewinnen, sichtbar machen. Schauen wir uns diese Konzepte einmal an.

3.2 Grundlegende Konzepte des Collaborative Modeling

Die verschiedenen Techniken des Collaborative Modeling bauen auf den folgenden gemeinsamen Konzepten auf (s. [Hofer et al. 2020]):

- **Gruppenarbeit aller Beteiligten**
  Grundlegend ist, dass die verschiedenen Beteiligten auf Anwendungs- und Entwicklungsseite gemeinsam Arbeitsprozesse und Anforderungen klären, statt dass einzelne Spezialisten auf der Basis von Interviews und Dokumentanalysen diese Themen spezifizieren. Alle Ansätze sehen gemeinsame Workshops vor, manchmal in großen Gruppen, manchmal mit wenigen Vertretern der Gruppen.
- **Geschichten erzählen**
  In den Workshops diskutieren die Beteiligten über konkrete Szenarien statt über abstrakte Beschreibungen. Szenarien werden wie Geschichten erzählt. Sind die Szenarien von allen verstanden, dann werden die für die Softwareentwicklung nötigen Abstraktionen und »vollständigen« Abbildungen von Regeln und Fallunterscheidungen aufgebaut. Unterschiede gibt es darin, ob für jeden Fall ein eigenes Bild erzeugt wird oder alle relevanten Fälle in einem Bild landen.
- **Über eine gemeinsame Sprache ein wechselseitiges Verständnis aller Beteiligten anstreben**
  Oft wird beklagt, dass die Fachabteilungen und die IT in ihren eigenen Welten leben und aneinander vorbeireden. Alle CoMo-Ansätze stellen dagegen die Sprache der Anwender in den Vordergrund; die verschiedenen Workshop-Techniken sollen den Beteiligten helfen, sich besser zu verstehen.

- **Durch visuelle Mittel Verständnis aufbauen**
  Das direkte Gespräch und Zuhören war, ist und wird auch in Zukunft der beste Weg sein, um miteinander zu kommunizieren. Oft wird beim Kommunizieren ohne Visualisierung der Beitrag des Gegenübers missverstanden. Die Visualisierung eröffnet einen zusätzlichen Feedback-Kanal und hilft, die Diskussion wirklich auf den Punkt zu bringen und unterschiedliche Ideen zu betrachten.
- **Gemeinsame Modelle des Anwendungsbereichs erstellen**
  Unsere Erfahrungen mit CoMo zeigen, dass die gemeinsame Arbeit ein gegenständliches Ergebnis haben sollte. Die verschiedenen Modelle aus Zeichnungen, Karteikarten oder Post-its sind mehr als ein Wegwerfprodukt. Sie zeigen allen Beteiligten den Stand der Diskussion, ihre grafische Darstellung bietet eine neue Sicht auf das Thema und markiert am Ende eines Workshops ein handfestes Ergebnis, das oft in der weiteren Arbeit genutzt werden kann.
- **Exploratives Lernen beim Modellieren**
  Das Modellieren findet in einem zunächst unbekannten Erkenntnisraum statt. Wir wissen vorher nicht, wer die richtigen Fragen stellt oder wer die entscheidenden Antworten gibt. Es wird auch nicht den einen Experten geben, der sich in allen Bereichen der Fachwelt perfekt auskennt, sondern in der Regel hat jeder Teilnehmer ein unterschiedlich tiefes Verständnis verschiedener Teile der Fachlichkeit. Noch dazu ist es bei der klassischen Softwareentwicklung häufig so, dass das Entwicklungsteam das Feedback der Anwender erst spät im Projektverlauf bekommt. Damit die nötigen Diskussionen schon zu Beginn eines Projekts geführt werden, ist Collaborative Modeling ein sehr guter Ansatzpunkt.

Beim Collaborative Modeling ist das Problem für alle sichtbar am Whiteboard, an der (Lein-)Wand oder der Tafel, und die Teilnehmer arbeiten mit- und nicht gegeneinander an einer Lösung.

3.3 Modellierungswerkzeuge und das Modellmonopol

Der entscheidende Punkt bei allen Methoden für Collaborative Modeling ist, dass die Nicht-Softwareentwickler in der Diskussion über die Fachlichkeit keine Probleme mit dem Verständnis der Notation haben sollen. Wählt man für die Analyse der Fachdomäne mit den Fachexperten eine fachfremde Notation, wie UML, BPMN oder EPK[1], so schlägt ein Phänomen zu, das in der Soziologie als Modellmonopol bezeichnet wird [Bråten 1973].

Ein **Modellmonopol** liegt vor, wenn für ein Modell eine Notation eingesetzt wird, die eine Gruppe in einer Diskussion oder in einem Verständigungsprozess besser beherrscht als eine andere Gruppe. Fachexperten haben in der Regel keine Ausbildung

1. *https://de.wikipedia.org/wiki/Unified_Modeling_Language*, *https://de.wikipedia.org/wiki/Business_Process_Model_and_Notation*, *https://de.wikipedia.org/wiki/Ereignisgesteuerte_Prozesskette*.

in technischen Notationen, sodass sie bei einer Analyse auf Basis dieser Notationen einen deutlichen Nachteil gegenüber Softwareentwicklern oder Business-Analysten haben. Ist man sich dieser Problematik nicht bewusst und setzt eine Notation ein, die für die Fachexperten eine Verständnishürde bedeutet, dann wird man weniger Feedback und korrigierende Anmerkungen bei der Erarbeitung und Präsentation der Ergebnisse bekommen.

Im Endeffekt läuft man mit einer für die Fachexperten schwer verständlichen Notation in eine frustrierende und teure Falle bei der Softwareentwicklung: Die Fachexperten stimmen den Ergebnissen der Analyse ihrer Geschäftsprozesse ohne viele Anmerkungen zu, weil sie Verständnisschwierigkeiten mit der Notation haben und die Expertise der anderen Seite nicht infrage stellen können/wollen. Die Software wird auf Basis dieser Analysen und des daraus entstandenen Pflichtenhefts (weiter-)entwickelt. Aber bei der Auslieferung wird klar: So ist die Software oder ihre Erweiterung nicht zu gebrauchen. Die Anwender benötigen ein etwas anderes Domänenmodell für ihre Arbeit und bei den in der Software festgeschriebenen Abläufen fehlen Schritte bzw. die Reihenfolge der Schritte ist teilweise nicht sinnvoll. Es müssen viele Workarounds eingebaut werden und das gewählte Domänenmodell muss überarbeitet werden. Insgesamt ist der Umbau der Software in die fachlich sinnvolle Richtung zum Zeitpunkt der Auslieferung sehr aufwendig und zum Teil nicht mehr realisierbar. Hätte man doch nur bei der Analyse der Fachdomäne eine Notation gewählt, die die Fachexperten verstehen, und so das wertvolle Feedback vor der Implementierung bekommen!

Selbstverständlich sind die im Software Engineering entwickelten Modellierungswerkzeuge und Notationen, wie BPMN, UML und weitere, wichtig und verwendbar, aber eben nicht für die Kommunikation mit Fachexperten. Diese Werkzeuge und Notationen sind als Kommunikationsmittel zwischen Entwicklern gedacht. Bei der Sourcecode-nahen Kommunikation zwischen Entwicklern ist formale Korrektheit wichtig. Beim Collaborative Modeling wird bewusst darauf verzichtet.

3.4 Collaborative Modeling und DDD

Im »Blauen Buch« von Evans gibt es mehrere Kapitel, in denen er von etwas spricht, das er *Knowledge Crunching* nennt, also das Wissen knacken. Konkrete Methoden, wie man das Wissen knacken könnte, fehlen sowohl in seinem Buch als auch im »Roten Buch«. Diese Lücke füllt Collaborative Modeling.[2]

Erfreulicherweise lässt sich CoMo mit DDD sehr gut kombinieren – Fachsprache, Ereignisse, Handlungen, Arbeitsmittel und Strukturen der Domäne bilden das sogenannte Domänenmodell, das die Entwickler in der Software abbilden. Ein valides Do-

2. Wie DDD-Gründervater Eric Evans in seinen Keynotes immer wieder sagt (z.B. Explore DDD 2018 in Denver, DDD Europe 2019 in Amsterdam): »DDD isn't done! Domain-Driven Design is not a dogma, but a method that must always be extended, refined and improved.« Er hat dazu aufgerufen, weitere gute Ansätze und Methoden in DDD zu integrieren und auf diese Weise die Softwareentwicklung besser zu machen. Genau das ist mit den Methoden zum Collaborative Modeling passiert.

mänenmodell lässt sich nur gemeinsam von Entwicklern und Fachexperten erstellen. DDD-Experten, wie Paul Rayner, sehen Collaborative Modeling deshalb mittlerweile als eine wichtige Säule von DDD an (s. Abb. 3–4).

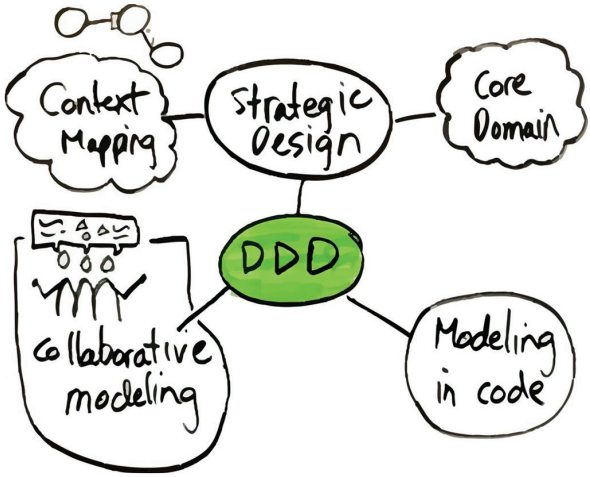

Abb. 3–4 *Collaborative Modeling als Säule von DDD*[3]

In unserem Vorgehen *Domain-Driven Transformation* setzen wir beim Zerlegen von Legacy-Systemen auf zwei Methoden aus dem Collaborative Modeling: Domain Storytelling und Event Storming.

3.5 Domain Storytelling

Domain Stories [Hofer & Schwentner 2023] kombinieren eine einfache grafische Notation mit einem Workshop-Format, um fachliche Prozesse zu erfassen. Dabei versuchen wir die grundsätzliche Frage zu klären:

Wer macht **was womit wozu**?

Die Notation für Domain Stories gibt zwei Arten von Icons und eine Art von Pfeil vor:

- **Akteur** (engl. **Actor**, Icon) – eine handelnde Person, Organisationseinheit (oder eventuell ein handelndes IT-System).
- **Arbeitsgegenstand** (**Work Object**, Icon) – ein fachliches Ding, mit dem etwas getan wird.
- **Aktivität** (**Activity**, Pfeil) – eine Handlung, die ein oder mehrere Akteure an oder mit einem oder mehreren Arbeitsgegenständen vornehmen.

3. Wir danken Paul Rayner für die Erlaubnis, diese Grafik hier abzubilden. Quelle: Workshop auf der Explore DDD 2018. Foto: Martin Schimak.

Mit diesen Symbolen werden Sätze gebildet. Will man eine Domain Story lesen, folgt man den **Sequenznummern** (engl. **Sequence Numbers**) an den Sätzen. Alles, was sich nicht mit diesen Bausteinen ausdrücken lässt, wird in **Notizen** (engl. **Annotations**) aufgeschrieben.

Domain Stories entstehen, indem wir uns von den Fachexperten erzählen lassen, was in der Domäne geschieht. Während der Erzählung wird die Geschichte so, wie wir sie verstehen, als Diagramm aufgezeichnet. Auf diese Weise geben wir direkte Rückkopplung über das Verständnis und mögliche Missverständnisse werden schnell aufgeklärt. Dieser Punkt ist besonders wichtig, wenn man schnell vorankommen will. Der schriftliche Austausch von Informationen über eine Domäne ist um Längen weniger effektiv als der direkte Austausch. Die Dokumentation des gemeinsamen Verständnisses ist für den fortschreitenden Lernprozess sehr wichtig – dafür sind Domain Stories eine für alle Beteiligten verständliche Lösung.

Im Programmkino-Beispiel führen wir unseren ersten Workshop zur Analyse von CineSmall mit den Fachanwendern und Entwicklern durch. Das erste Ergebnis unseres Workshops ist eine Domain Story, die als Überblick beschreibt, welche Arbeitsschritte der Kinomanager und die Kassenmitarbeiter heute mit CineSmall erledigen und wie sie dabei zusammenarbeiten.

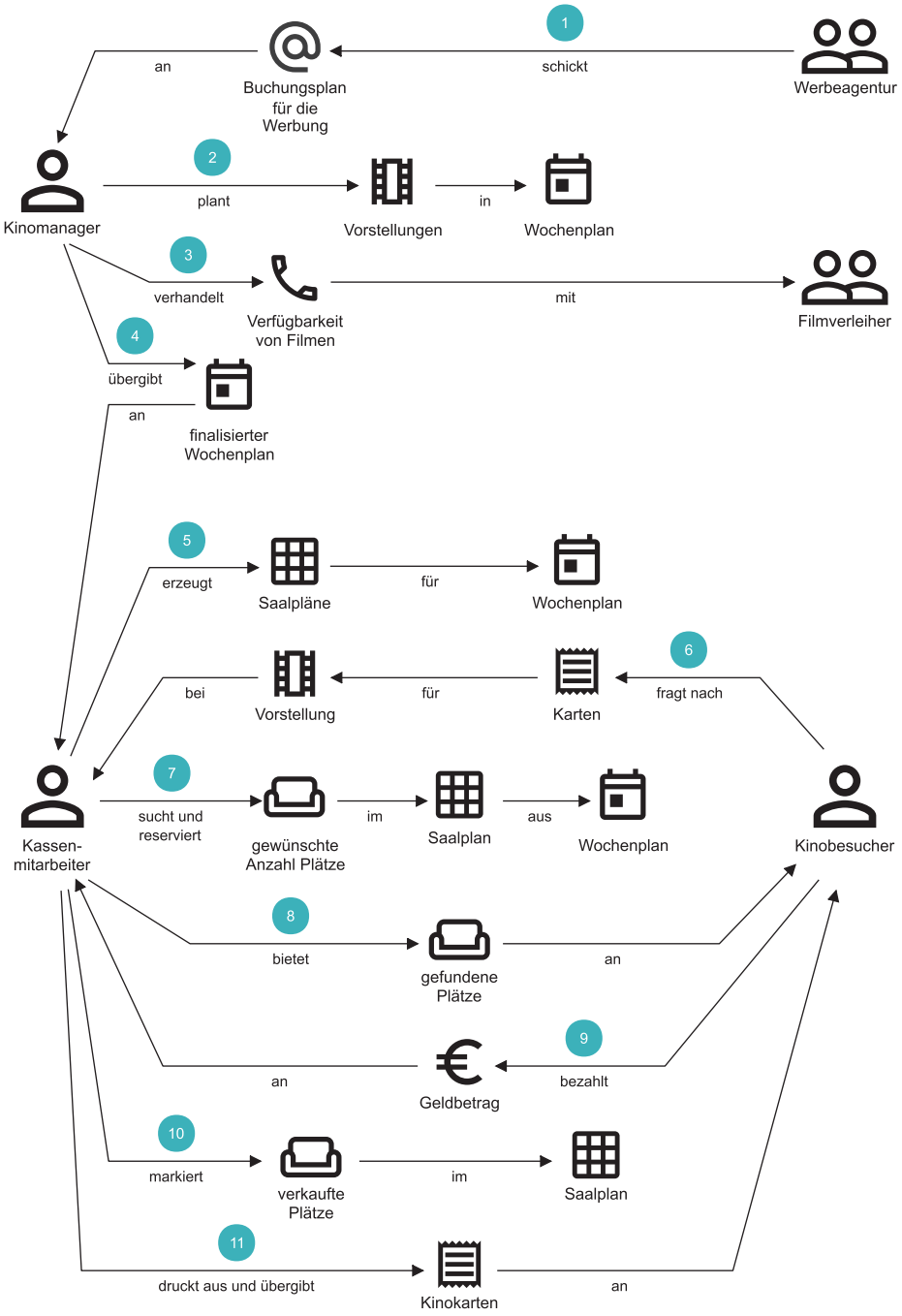

Abb. 3–5 *Gesamtprozess im Programmkino*

Abbildung 3–5 zeigt eine Domain Story für unser kleines Programmkino: Der Kinomanager links oben bereitet den Wochenplan (Schritt 2) für die kommende Woche in Zusammenarbeit mit der Werbeagentur (Schritt 1) und dem Filmverleiher (Schritt 3) vor. Sobald der Wochenplan fertig ist, finalisiert er ihn (Schritt 4) und gibt den Plan dadurch für den Kartenverkauf frei. Der Kassenmitarbeiter kann nun dem Kinobesucher Kinokarten verkaufen (Schritt 5 bis 11), indem er in den Saalplänen nach passenden Plätzen sucht (Schritt 7) und sie dem Kinobesucher anbietet (Schritt 8). Der Kinobesucher zahlt schließlich (Schritt 9) und zieht mit seinen ausgedruckten Kinokarten (Schritt 11) zufrieden davon.

3.5.1 Ein Bild – eine Geschichte

Domain Storytelling ist ein szenariobasiertes Vorgehen, was bedeutet, dass jedes Diagramm nur einen Fall darstellt. So bleibt das einzelne Bild verständlich. Für weitere (interessante) Fälle werden weitere Domain Stories gezeichnet.

Im Kino haben wir bisher den »Gutfall« betrachtet. Als Nächstes werden Domain Stories für die relevanten Sonderfälle gezeichnet, die wir hier aus Platzgründen weglassen.

3.5.2 Grenzen ziehen

Domain Stories bieten als weiteres Element der Bildsprache **Gruppierungen** (engl. **Groups**) an. Diese kann man u. a. dazu verwenden, Subdomänen sichtbar zu machen.

In diese Domain Story vom Programmkino zeichnen wir die gleichen Subdomänen ein, die wir in Abschnitt 2.2.2 bereits anhand der Kernbegriffe gefunden hatten: die Subdomäne »Wochenplanung« und die Subdomäne »Kartenverkauf«.

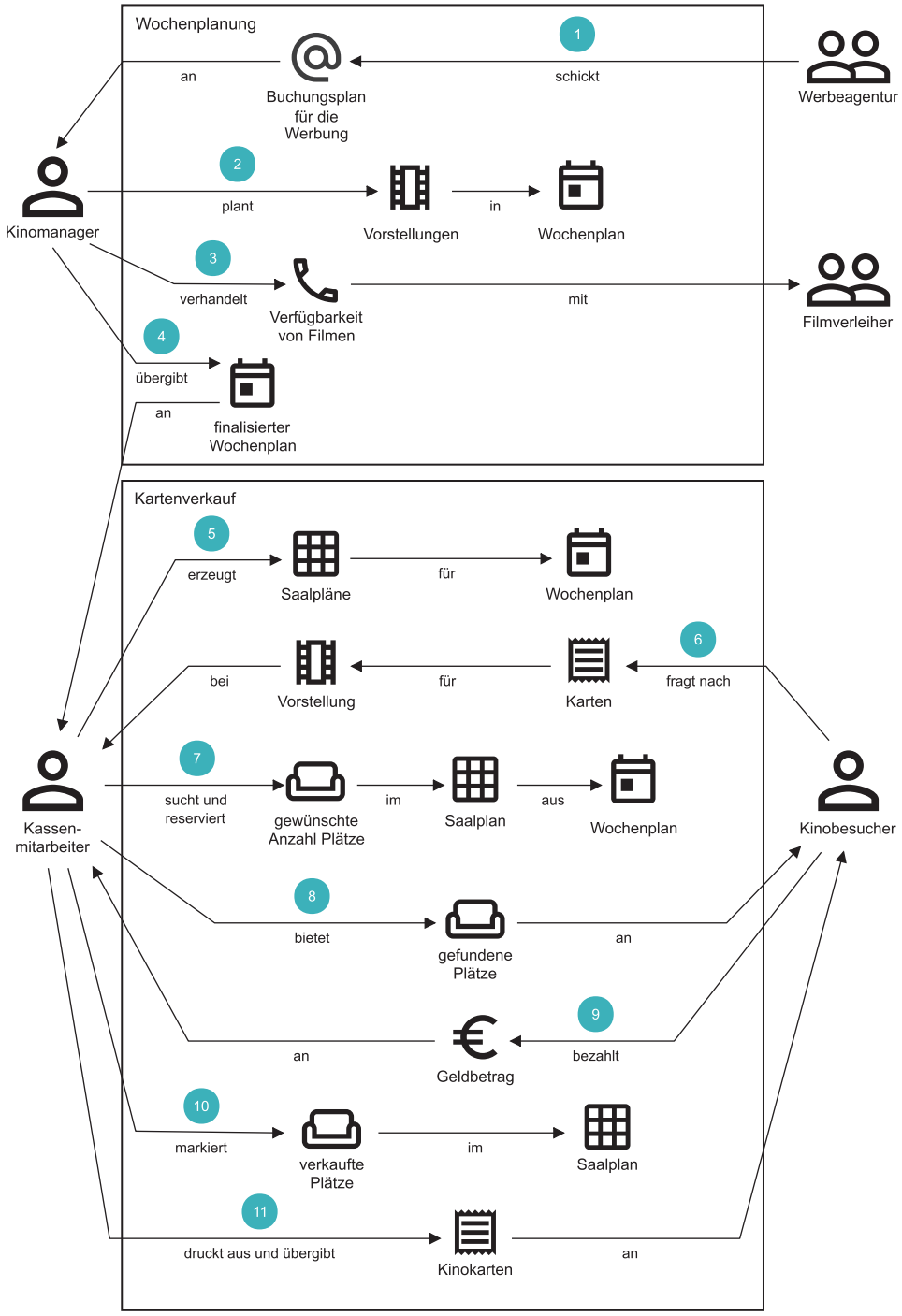

Abb. 3–6 *Gesamtprozess im Programmkino mit Subdomänen*

Mehr zum Thema Subdomänen im Arbeitsprozess finden mit Domain Stories zeigen wir in Abschnitt 9.7.

Neben der grundlegenden Technik, Domain Stories zu entwickeln und sie in Subdomänen aufzuteilen, führen Henning[4] und sein Co-Autor Stefan in ihrem Buch verschiedene Arten von Domain Stories ein, die anhand der sogenannten Scope-Faktoren kategorisiert werden.

3.5.3 Scope-Faktoren für Domain Stories

Domain Stories haben unterschiedliche Ausprägungen, die vom sogenannten Scope bestimmt werden. Der Scope einer Domain Story setzt sich zusammen aus den folgenden Faktoren:

- **Granularität**
  Domain Stories können grobgranular, feingranular und alles dazwischen sein. Das Spektrum ist kontinuierlich. Wichtig ist, einen einheitlichen Detaillierungsgrad in einer Domain Story zu erreichen. Die Vermischung von feingranularen und grobgranularen Aktivitäten ist verwirrend und weist in der Regel daraufhin, dass einige Wissensträger beim Erstellen der Domain Story gefehlt haben.
- **Domain Purity**
  Domain Stories können rein fachlich modelliert werden, sodass nur menschliche Akteure und keine Softwaresysteme in ihnen auftauchen. Solche rein fachlichen Domain Stories sind »rein« bzw. »pur«. Tauchen ein oder mehrere Softwaresysteme in der Domain Story auf, bezeichnet man sie als »digitalisiert«.
- **Zeitpunkt**
  Domain Stories können für die aktuelle Situation, das »Ist«, oder für die Zukunft, das »Soll«, modelliert werden. Je nachdem ob man eine Fachdomäne zuerst einmal so verstehen will, wie sie ist, oder ob man den nächsten Schritt in der Zukunft plant, hat die Domain Story einen anderen Zeitpunkt.

Es sollte für jede Domain Story ein Tripel aus diesen drei Faktoren angegeben werden und welche Ausprägung sie jeweils haben (X, Y, Z). Für Abbildung 3–5 würde dieses Tripel folgendermaßen aussehen: (grobgranular, pur, Ist). Die Scope-Faktoren werden uns später bei der Domain-Driven Transformation helfen, die Fachlichkeit herauszukristallisieren und auf dieser Basis das Altsystem zu zerlegen.

Der Unterschied zwischen reinen und digitalisierten Domain Stories lässt sich an den beiden Domain Stories, die nur den Verkaufsprozess in unserem Kinobeispiel zeigen, gut nachvollziehen (s. Abb. 3–7 und 3–8). In der digitalisierten Story in Abbildung 3–7 ist der Verkaufsprozess zu sehen, in dem der Kassenmitarbeiter dem Kinobesucher Kino-

4. Um nicht einfach nur das Buch [Hofer & Schwentner 2023] nachzuerzählen und um einen weiteren Blick auf die Methode zu geben, wurde dieser Abschnitt von Carola geschrieben.

karten für eine Vorstellung verkauft. Der Kassenmitarbeiter setzt dabei das Softwaresystem CineSmall ein, das links in der Domain Story eingezeichnet ist.

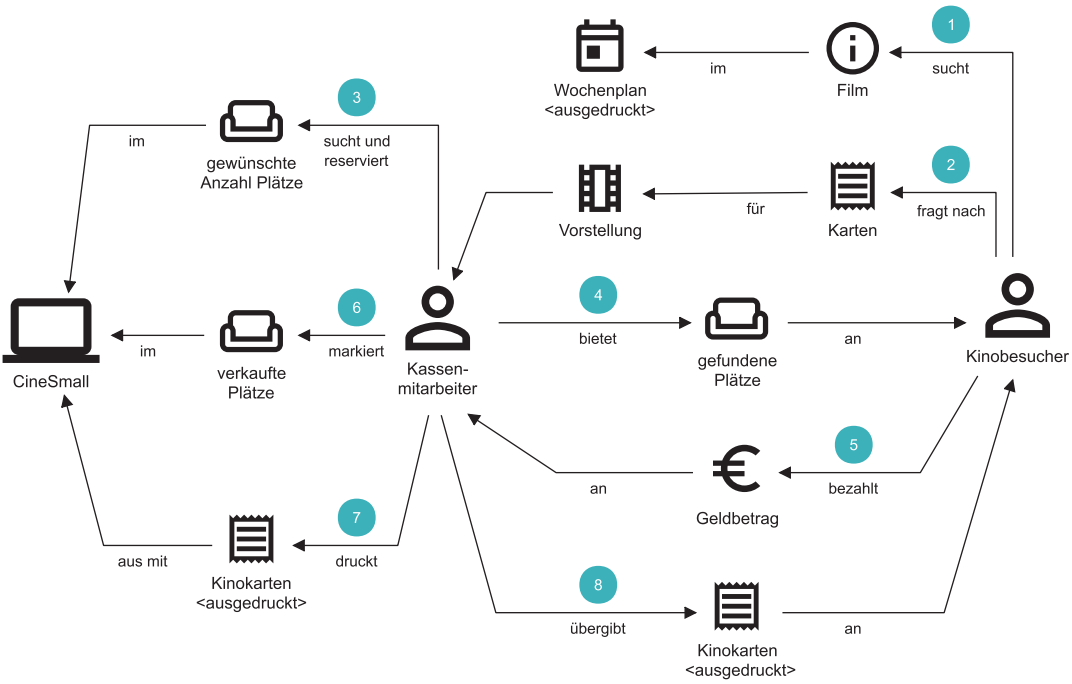

Abb. 3–7 *Verkaufsprozess im Kino mit einem System*

Vergleicht man diese Domain Story mit dem Grobprozess aus Abbildung 3–5, dann sieht man, dass der Saalplan hier gar nicht existiert.

So etwas passiert oft, wenn man einen bereits digitalisierten Prozess mit den Fachexperten aufnimmt. Die Fachexperten sind so an die Arbeit mit ihrem Softwaresystem gewöhnt, dass sie die ursprünglichen fachlichen Konzepte nicht mehr präsent oder sogar vergessen haben. Unsere Aufgabe besteht dann darin, den fachlichen Prozess herauszudestillieren.

Schaut man sich nun die nächste, von Technik »bereinigte« Story in Abbildung 3–8 an, so sieht man, dass CineSmall nicht mehr vorkommt, dafür aber der Saalplan als fachliches, IT-gestütztes Artefakt auftaucht. Dieses für das Kino doch sehr wichtige Artefakt wurde in der Analyse der Domäne erst sichtbar, als wir die Fachexperten motivierten, darüber zu sprechen, wie die Dinge ablaufen würden, wenn alle Aktivitäten nur durch die Domäne motiviert wären (und nicht durch die bestehenden Softwaresysteme).

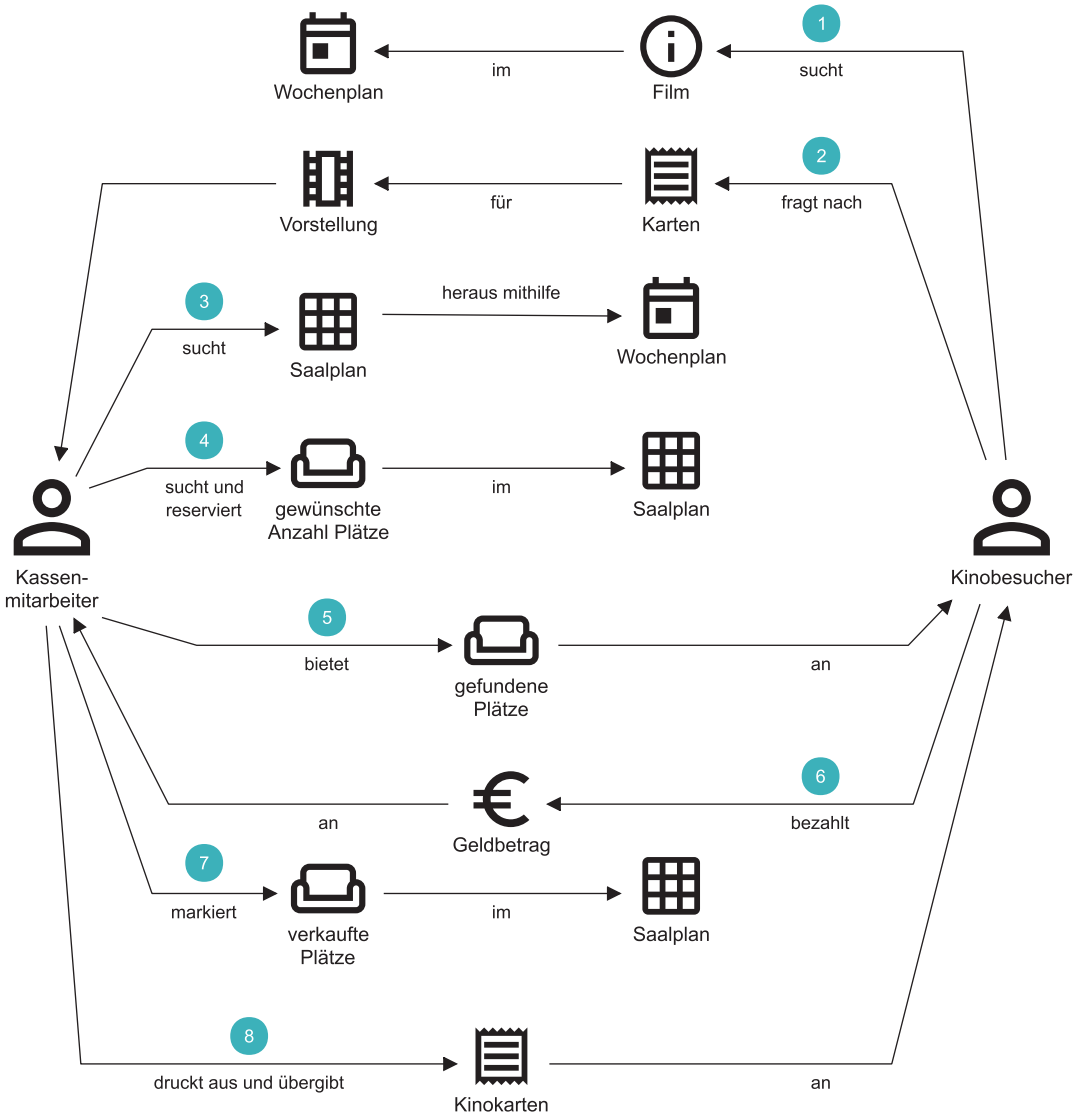

Abb. 3–8 *Verkaufsprozess im Kino ohne System*

3.5.4 Das Domänenmodell finden

Wendet man Domain Storytelling so an, dass man pure Domain Stories hat, dann sollten diese Stories ausschließlich Fachworte und keine technischen Worte enthalten. Haben wir das geschafft, dann haben wir uns wirklich auf die Fachlichkeit fokussiert und viele Begriffe für die Ubiquitous Language und damit für das Domänenmodell gefunden.

In der puren Domain Story vom Verkaufsprozess im letzten Abschnitt (s. Abb. 3–8) und auch im Gesamtprozess weiter oben (s. Abb. 3–5) sind wir ausschließlich in der Fachsprache des Kinos unterwegs. Alle die dort verwendeten Begriffe sind Kandidaten für die Ubiquitous Language des Kinos: »Wochenplan«, »Vorstellung«, »Karte«, »Saalplan«, »Sitzplatz« usw. Dabei interessieren uns nicht nur die Substantive, sondern auch die Verben: »anbieten«, »bezahlen«, »planen« usw.

Mit diesen Begriffen und Verben können wir direkt in die Modellierung gehen und aus der Domain Story das Domänenmodell ableiten. Wir entwerfen zwei Klassen `Saalplan` und `Sitzplatz` und notieren an der Klasse `Saalplan` die Methode `markiereAlsVerkauft()`, um einen typischen Umgang des Kassenmitarbeiters mit dem Saalplan zu modellieren (s. Abb. 3–9).

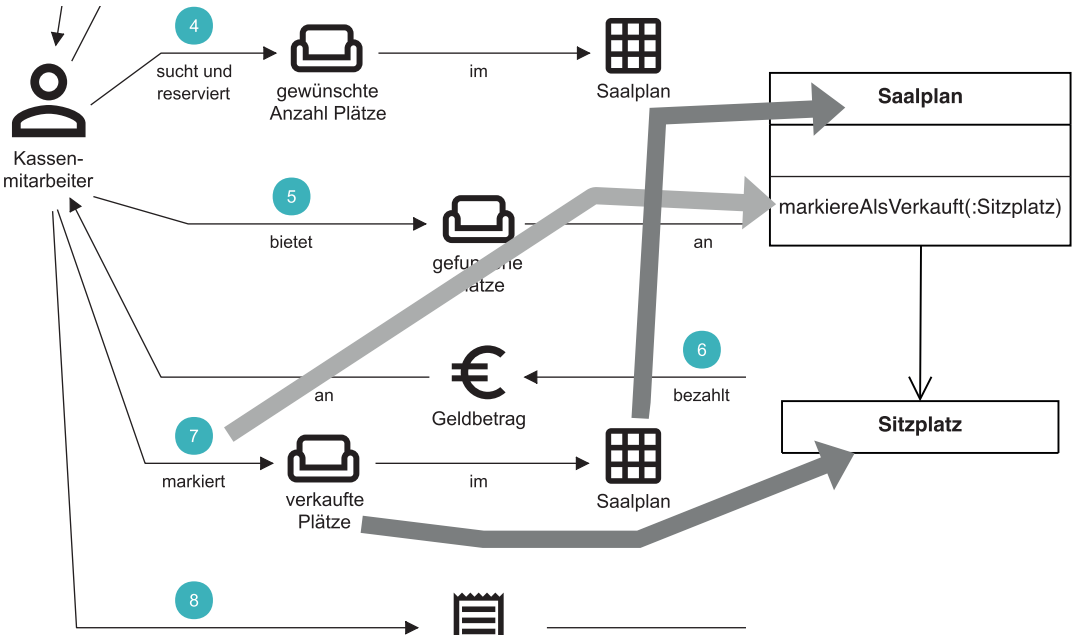

Abb. 3–9 *Von der Domain Story zum Domänenmodell*

3.5.5 Ausblick

Falls Sie sich die gesamte Technik Domain Storytelling aneignen wollen, dann verweisen wir Sie auf das gleichnamige Buch von Stefan Hofer und Henning [Hofer & Schwentner 2023] und die Webseite von Domain Storytelling[5].

5. *https://domainstorytelling.org*

3.6 Event Storming

Event Storming ist eine Technik, um in Workshops alle Beteiligten zur Zusammenarbeit zu bewegen. Erfunden wurde Event Storming von Alberto Brandolini (s. [Brandolini 2021]). Beim Event Storming wird die Frage etwas anders als beim Domain Storytelling gestellt. Beim Domain Storytelling war die Frage: »Wer macht was womit wozu?«. Bei Event Storming lautet die Frage:

»Welche **Ereignisse** spielen sich in der Fachlichkeit ab?«

Beim Event Storming wird ein gemeinsames Verständnis der Domäne anhand von fachlichen Schlüsselereignissen gefunden (also **Domain Events,** wie sie uns schon in Abschnitt 2.3.7 begegnet sind). Die Domain Events werden mit Klebezetteln sichtbar und greifbar gemacht, die in einem Workshop gemeinsam aufgeklebt werden.

Die Teilnehmer an einem Event-Storming-Workshop versammeln sich in einem Raum vor einer breiten Wand[6], an der als **Modellierungsfläche** (engl. **Modeling Surface**) ca. 10 m einer Papierrolle mit Kreppband angebracht sind. Tische und Stühle sind beiseite geräumt oder ganz entfernt. So wird sichergestellt, dass alle gleichberechtigten Zugang zur Wand haben.

Der Workshop wird von einem sogenannten **Facilitator** moderiert. Auf einem Pult, Stehtisch o.Ä. legt der Facilitator das benötigte Modellierungsmaterial bereit: Klebezettel in unterschiedlichen Farben (s. Abschnitt 3.6.4) und Marker, um auf die Klebezettel zu schreiben. Jeder der Teilnehmer bekommt am Anfang einen Stapel mit orangefarbenen Stickies und einen Marker in die Hand.

3.6.1 Storming-Phase

Zunächst schreiben alle Teilnehmer als Ereignisse auf, was in der Domäne passiert. Ähnlich wie beim Brainstorming geht es zunächst darum, Wissen und Ideen zu sammeln, und noch nicht darum, andere zu korrigieren. Das Ganze geschieht eher leise.

Die Teilnehmer schreiben **Domain Events** auf orangefarbene Stickies und kleben diese auf die Modellierungsfläche. Zur Erinnerung (vgl. Abschnitt 2.3.7): Ein Domain Event ist ein Ereignis, das relevant für die Fachexperten ist. Die Vorgabe ist, dass die Domain Events in der Vergangenheitsform aufgeschrieben werden, wie »Sitzplätze reserviert« oder »Wochenplan finalisiert«. Manche Stickies erscheinen doppelt und dreifach; oft in leicht unterschiedlicher Formulierung. Das ist okay und Grundlage für die Diskussionen, die jetzt anfangen.

6. Manchmal steht keine Wand zur Verfügung, die genug Platz bietet. In diesem Fall können längliche Tische in einer Reihe aufgestellt werden, auf denen die Papierrolle ausgerollt werden kann. Idealerweise stehen die Tische dann so, dass die Modellierungsfläche von beiden Seiten gut zugänglich ist.

Eine weitere Beispieldomäne: Bankwesen

An dieser Stelle führen wir unser nächstes Beispiel, das Bankwesen, ein. In dieser Bank gibt es ein Hostsystem, mit dem alle Prozesse in der Bank unterstützt werden. Dieses Hostsystem soll abgelöst werden, aber es ist allen Beteiligten klar, dass das nicht auf einmal passieren kann. Es müssen Teilbereiche gefunden werden, die man durch neue Software ersetzen kann.

Wir schlagen dem Management der Bank ein Event Storming vor, um die Domäne in Subdomänen zu zerlegen. Dazu laden wir Fachexperten aus verschiedenen Abteilungen und die Entwicklungsteams ein. Gemeinsam finden wir zunächst die in Abbildung 3–10 dargestellten Events.

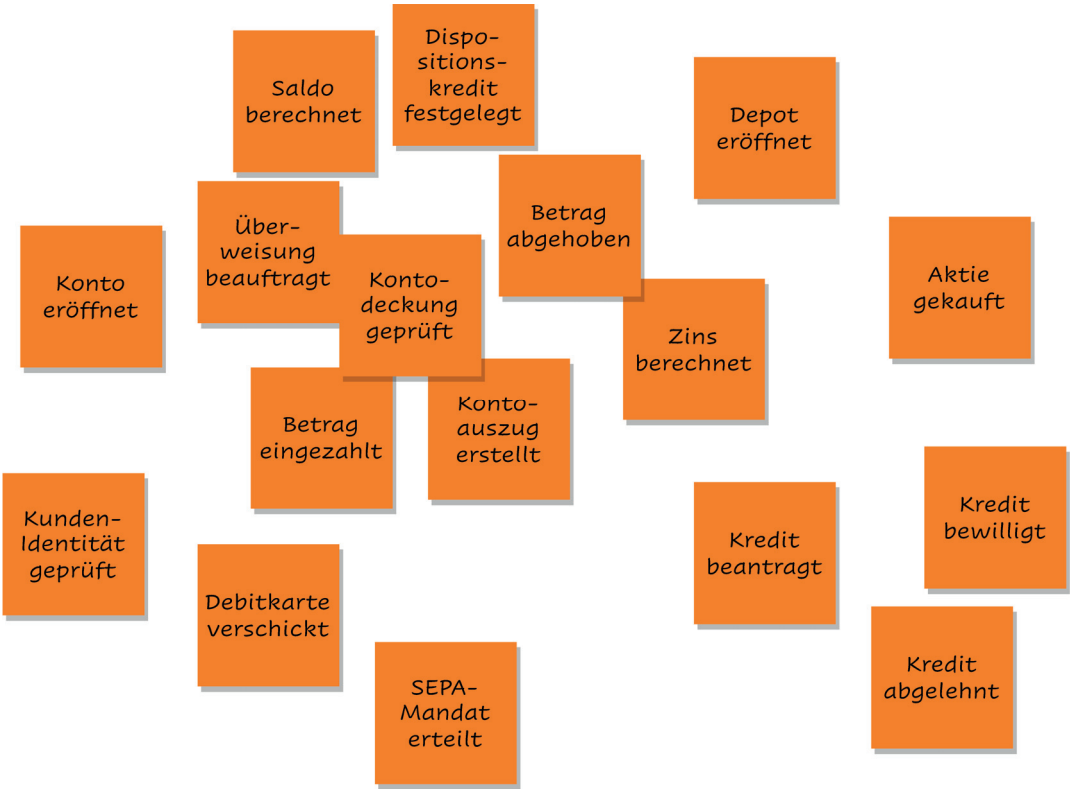

Abb. 3–10 *Storming-Phase: Alle Teilnehmer lassen ihr Wissen und ihre Ideen hervorsprudeln.*

3.6.2 Zeitliche Ordnung herstellen

Um etwas Struktur in das sich entwickelnde Modell zu bringen, werden die Ereignisse in eine grobe zeitliche Reihenfolge von links nach rechts gebracht. Dadurch werden viele Stickies umgehängt und die Diskussion wird breiter und tiefer.

Die Events in eine zeitliche Reihenfolge zu bringen, macht es einfacher, Lücken in der Geschichte aufzudecken, in denen weitere Ereignisse fehlen. Doppelte Stickies werden jetzt entfernt, schlechtere Varianten verworfen. Ein gutes Event Storming zeichnet sich u. a. dadurch aus, dass in dieser Phase viele Zettel von der Wand entfernt und zusammengeknüllt auf den Boden geworfen werden.

Während der Workshop zu Beginn einigermaßen still verlief, wird es jetzt lauter. Anfangs ist jeder noch mit seinen eigenen Ideen beschäftigt, jetzt wird diskutiert. Typische Sätze, die man in dieser Phase hört, sind:

- »Passiert dieses wirklich vor jenem?«
- »Ach, so macht ihr das!«
- »Ich habe mich schon immer gefragt, warum wir diese Mail verschicken!«

In dieser Phase haben viele Teilnehmer Aha-Erlebnisse. Sie bauen ein gemeinsames Verständnis von dem auf, was eigentlich geschieht.

Wenn mehrere Menschen zusammenarbeiten, um zu einer gemeinsamen Erkenntnis oder zu einer gemeinsamen Entscheidung zu kommen, können drei Phasen beobachtet werden. Am Anfang wird der Raum geöffnet, die Teilnehmer starten am selben Punkt, driften oft erst einmal auseinander (Divergenz). Dann fängt es an zu knarzen, Neues wird geboren (Emergenz). Schließlich kommen alle wieder zusammen, es entsteht etwas Gemeinsames (Konvergenz). Genau das ist hier passiert.

In unserer Bank landen wir nach ausführlichen engagierten Diskussionen bei Abbildung 3–11.

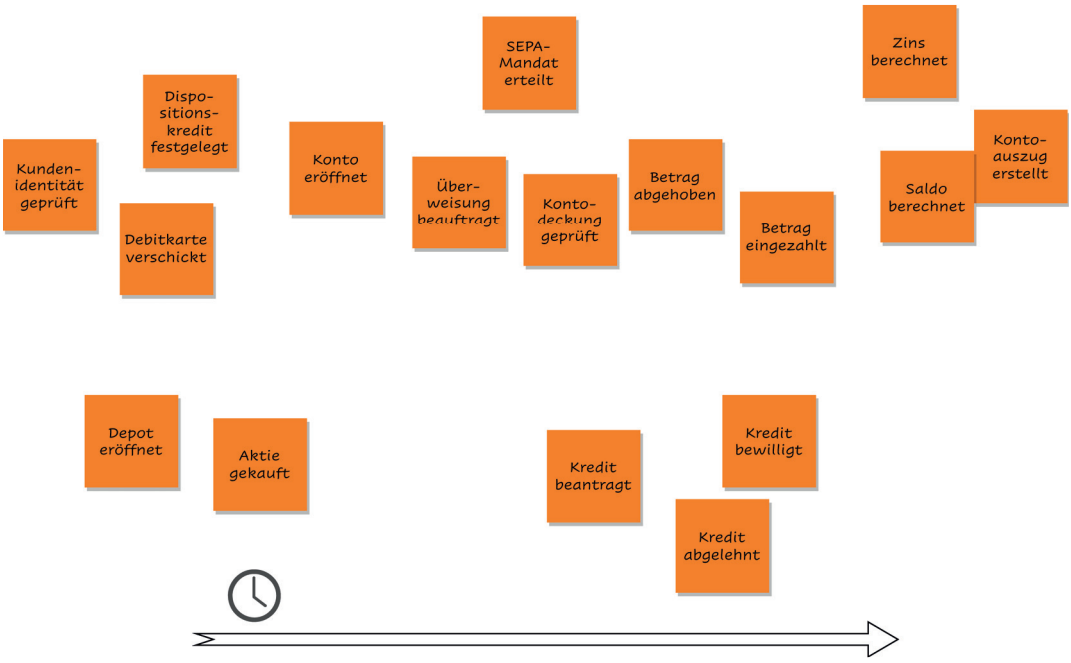

Abb. 3–11 *Zeit: Die Events werden in eine Reihenfolge gebracht.*

Nun ist der richtige Moment gekommen, um Grenzen zu ziehen.

3.6.3 Grenzen ziehen

Als Nächstes teilen wir die jetzt viel besser verstandene Domäne in Subdomänen auf. Diese Subdomänen sind Kandidaten dafür, später zu Bounded Contexts im Softwaresystem zu werden bzw. die Zerlegung des Softwaresystems anzuleiten.

In unserem Bankbeispiel ergeben sich durch die Diskussion mit unseren Fachexperten die Grenzen für Subdomänen, wie sie in Abbildung 3–12 zu sehen sind.

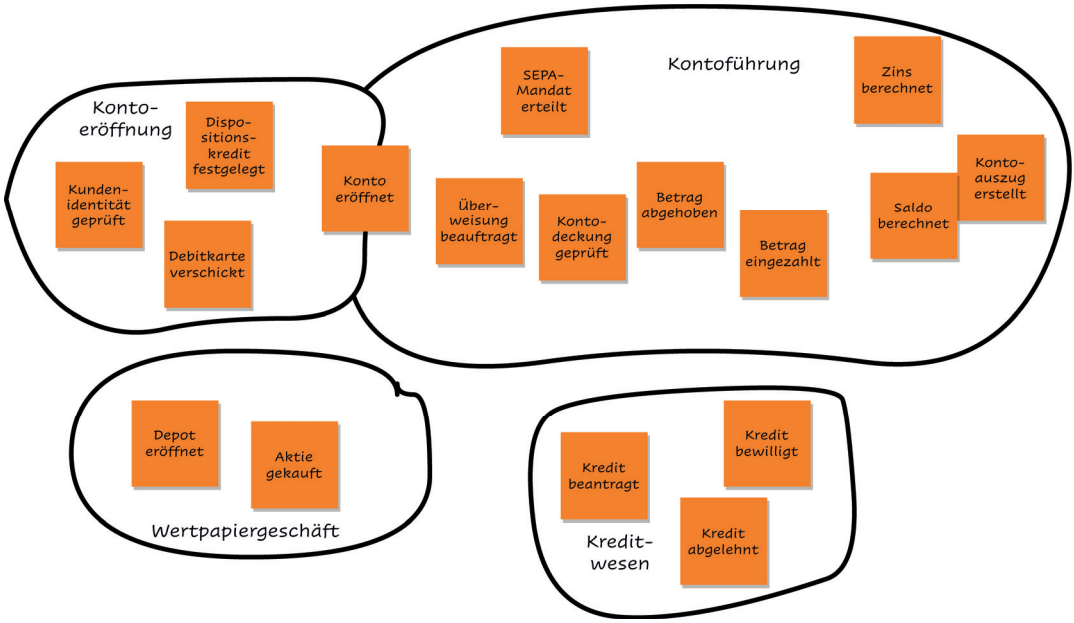

Abb. 3–12 *Grenzen: Die Events werden gruppiert.*

Was wir bisher gesehen haben, war die erste Ebene von Event Storming, mit der alle Event-Storming-Workshops beginnen: Es werden orange Klebezettel für Events beschrieben (»gestormt«) und sortiert. Wenn es erforderlich ist, werden währenddessen oder später weitere Klebezettel mit anderen Farben eingeführt.

3.6.4 Die Farbcodierung

Abbildung 3–13 fasst die Arten von Klebezetteln zusammen. In der Mitte sieht man den schon bekannten orangen Zettel, der für Domain Events verwendet wird. Als Nächstes werden üblicherweise Hot Spots in einem Event Storming eingesetzt:

- **Hot Spots**
  Wann immer offene Fragen oder kontroverse Diskussionen den Modellierungsprozess ausbremsen, markiert man den strittigen Punkt mit einem roten Klebezettel als Hot Spot für eine spätere Klärung und lenkt die Energie auf den Rest des Prozesses.

Häufig erst später interessant werden Akteure, Aggregates und Commands sowie externe Systeme:

- **Akteure** werden mit gelben hochkant ausgerichteten Klebezetteln dargestellt, auf denen der Name der Rolle steht und ein kleines Symbol für eine menschliche Person abgebildet ist.
- Für **Aggregates** werden ebenfalls gelbe Klebezettel verwendet, allerdings im Querformat.
- Eine Aktion eines Akteurs formulieren wir als **Command** auf einem blauen Klebezettel.
- **Externe Systeme** werden als breiter rosa Klebezettel dargestellt.

Beim Domain Storytelling haben wir den Unterschied zwischen Domain Stories mit Softwaresystemen als Akteuren und solchen ohne kennengelernt. Als dann die Scope-Faktoren für Domain Stories (s. Abschnitt 3.5.3) hinzukamen, wurde deutlich, dass es bei der Domain-Driven Transformation unser Ziel ist, pure Domain Stories ohne Softwaresysteme als Akteure zu modellieren. Beim Event Storming können mit rosa Klebezetteln Softwaresysteme modelliert werden, die die Anwender bei ihrer Arbeit verwenden – nichtsdestotrotz ist es auch beim Event Storming für die Domain-Driven Transformation meist sinnvoll, ohne Technik auszukommen.

Abb. 3–13 *Event-Storming-Klebezettel*

Um den Teilnehmern des Workshops zu helfen, werden die verschiedenen Farben der Stickies mit ihrer Bedeutung auf einem Flipchart neben der Modellierungsfläche als **visuelle Legende** zur Verfügung gestellt. Üblicherweise beginnen wir mit Events (orange) und Hot Spots (rot). Dann kommen vielleicht Akteure und Aggregates hinzu, ggf.

auch externe Systeme oder Commands. So wächst die Legende im Laufe der Session um weitere Farben – je nachdem, was benötigt wird.

Die Bedeutung der Farben ist wichtig, aber wenn sie uns in der gegebenen Situation nicht passt, ändern wir sie ab.

3.6.5 Tiefer ins Detail: Design Level Event Storming

Bisher haben wir ein grobes gemeinsames Bild von dem, was in der Domäne passiert, entwickelt. Deshalb wird ein solcher Workshop auch ein **Big Picture Event Storming** genannt. Man verschafft sich gemeinsam einen Überblick über die Ereignisse. Aus dem Big Picture Event Storming heraus entsteht oft der Wunsch, bei einzelnen Abschnitten weiter ins Detail zu gehen und mit einem **Design Level Event Storming** ein Domänenmodell für die Software zu entwerfen.

Für eine Session zur Detailarbeit benötigen wir oft nicht mehr alle Fachexperten gleichzeitig, sondern führen pro Bounded Context/Entwicklungsteam eine eigene Session durch. Innerhalb eines Bounded Context bohren wir ausgehend von den bereits gefundenen Events weiter in die Tiefe. Dabei stellen wir uns als Nächstes die Frage: **Wodurch** wurde das einzelne Domain Event ausgelöst? Das ist dann spätestens der Moment, in dem *Akteure*, *Commands* und *externe Systeme* im Event Storming auftauchen.

Analysieren wir die bereits gefundenen Domain Events mit dieser Verfeinerung, schärfen wir unser Verständnis für die Domäne und finden Lücken in unserer bisherigen Analyse. Häufig entdecken wir in dieser Phase auch noch weitere Events, die wir bei der Big-Picture-Session übersehen haben. Schauen wir uns das in unserem Bankbeispiel an:

In der Bank starten wir am nächsten Tag eine neue Session für ein Design Level Event Storming mit dem Team, das für den Bounded Context »Kontoführung« zuständig ist. (Diesen hatten wir in Abbildung 3–12 gefunden.) Ein Auszug daraus, der die Teilschritte einer Überweisung modelliert, ist in Abbildung 3–14 dargestellt.

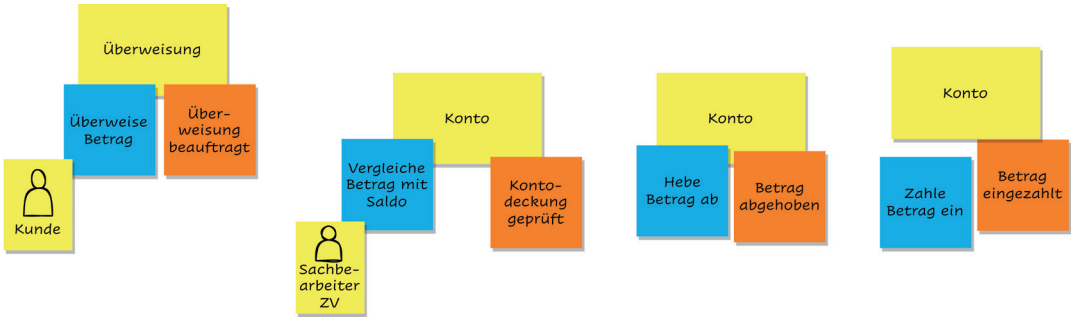

Abb. 3–14 *Design Level Event Storming des Bounded Context »Kontoführung« (Auszug)*

Alle Leser, die sich im Bankgeschäft auskennen, sehen sicherlich sofort, dass dieser Auszug nur für Überweisungen funktioniert, die innerhalb einer Bank ausgeführt werden. Besteht das Konto des Geldempfängers bei einer anderen Bank, so muss die Überweisung über das Clearing in der dafür verantwortlichen Landesbank erfolgen. Ein Prozess, der für dieses Buchbeispiel zu komplex ist.

In Abbildung 3–14 sieht man die Commands, die die Events auslösen, sowie die Akteure »Kunde« und »Sachbearbeiter ZV«, die diese Commands auslösen. Außerdem finden sich die Aggregates »Überweisung« und »Konto«.

3.6.6 Das Domänenmodell finden

Ähnlich wie beim Domain Storytelling (s. Abschnitt 3.5.4) kann man aus dem Strom der Events an der Wand einen ersten Entwurf für die Implementierung des Domänenmodells ableiten: sozusagen ein Klassendiagramm aus Klebezetteln. Im objektorientierten Entwurf werden dabei Aggregates zu Klassen und Commands zu Methoden.[7]

Im Bankbeispiel hängen wir die gelben und blauen Klebezettel um und kommen zu einer ersten Idee, wie eine Klasse `Konto` mit fachlichen Methoden aussehen könnte (s. Abb. 3–15).

Abb. 3–15 *Domänenmodell für das Konto*

7. Funktional modelliert werden die Aggregates zu Typen und die Commands zu Funktionen. Bei einer Event-Sourced-Implementierung (s. [Fowler 2005]) werden auch die Domain Events zu Architekturelementen. Mit Command-Query-Responsibility Segregation (CQRS) gilt das auch für die Commands (s. [Fowler 2011], [Fowler 2017]).

In Kapitel 7 werden wir dieses Konto wiedersehen und verschiedene Möglichkeiten diskutieren, um Domänenklassen fachlich zu stärken.

3.6.7 Weitere Formate und Ausblick

Wir haben gesehen: Event Storming kommt in verschiedenen Formaten daher. Brandolini beschreibt dies mit der Analogie von Pizzas, die alle die gleiche Basis, aber unterschiedliche Beläge haben. Für dieses Buch gilt:

- **Big Picture Event Storming** ist das wichtigste Format für die Zerlegung mit strategischer Domain-Driven Transformation, die das Thema von Teil III des Buches ist.
- **Design Level Event Storming** geht noch tiefer ins Detail und hilft beim Einführen von mehr Fachlichkeit in das Domänenmodell mit taktischer Domain-Driven Transformation, wie wir es in Kapitel 7 beschreiben werden.
- **Weitere Formate** sind in anderen Situationen interessant, dazu verweisen wir auf die Literaturtipps weiter unten.

Die ultimative Referenz zu Event Storming ist [Brandolini 2021]. Praktische Tipps für die Moderation von Event-Storming-Workshops finden sich in [Rayner 2022].

3.7 Wann welche Methode einsetzen?

Wir haben zwei für unsere tägliche Arbeit an Legacy-Systemen wichtige Methoden vorgestellt: Domain Storytelling und Event Storming. Im weiteren Verlauf des Buches werden wir häufiger Domain Storytelling einsetzen, weil es sich für die Darstellung in einem Buch und generell für die Dokumentation besser eignet. In unseren Workshops verwenden wir die Methoden allerdings gleichberechtigt. Wann welche Methode richtig ist, wird in der Community gern diskutiert. Unsere Sicht ist die folgende:

- Event Storming eignet sich insbesondere, wenn wir einen Prozess einer (scheinbar) wenig strukturierten Domäne analysieren.
- Domain Storytelling zeigt seine Stärken, wenn wir Prozesse analysieren, bei denen die Kooperation zwischen unterschiedlichen Rollen im Fokus liegt.
- In Kapitel 13 werden wir zum Abschluss dieses Buches sehen, wie sich unterschiedliche Arten von Domänen verhalten.

Die Effektivität des Werkzeugs hängt auch davon ab, wer es in die Hand nimmt. So haben und entwickeln unterschiedliche CoMo-Praktiker auch unterschiedliche Stile. Als ein interessantes Mashup sei hier Storystorming von Martin Schimak genannt (s. [Schimak 2019]). Außerdem möchten wir noch auf ein Buch hinweisen, das im Herbst 2023 herauskommen soll: *Collaborative Software Design: How to Facilitate Domain Modeling Decisions* [van Kelle et al. 2023] – wir sind sehr gespannt!

3.8 Collaborative Modeling Remote

Kann man die Leute nicht im richtigen Leben zusammenbringen, dann kann mit digitalen Tools, wie Egon.io oder Miro, gearbeitet werden. Hier geht allerdings einiges verloren, weil man z.B. die Körpersprache der anderen Teilnehmer nicht lesen kann. Trotzdem kann sowohl Domain Storytelling als auch Event Storming auch in der Online-Situation funktionieren. Wir raten dabei von hybriden Workshops ab: Entweder sind alle Teilnehmer vor Ort oder alle online in einer Videokonferenz.

3.9 Andere Techniken des Collaborative Modeling

Im Rahmen einer Domain-Driven Transformation kann es sinnvoll sein, auch weitere CoMo-Methoden einzusetzen. Zum Beispiel planen viele Teams Implementierung und Umsetzung innerhalb eines Bounded Context mit User Story Mapping (s. [Patton 2014]). Um das Buch nicht zu überfrachten, gehen wir hier jedoch nicht tiefer auf Methoden ein, die nicht direkt zum Zerlegen von Monolithen verwendet werden. Stattdessen empfehlen wir dem stärker Interessierten einen Blick in [Baas-Schwegler & Rosa 2020] oder den Besuch der Konferenzserie *ComoCamp*[8].

3.10 Zusammenfassung

In diesem Kapitel haben wir Ihnen zwei verschiedene Methoden zur Anforderungsermittlung vorgestellt, die für Anwender leicht verständlich sind: Domain Storytelling und Event Storming. Beide Methoden gehören zum Collaborative Modeling, einer Gruppe von Workshop-Formaten, die sich in den letzten Jahren rund um DDD entwickelt haben. Im dritten Teil dieses Buches werden wir Domain Storytelling wieder aufgreifen.

8. *https://comocamp.org.*