The Addison-Wesley Signature Series

A Vaughn Vernon Signature Book

Domain Storytelling

A Collaborative, Visual, and Agile Way to Build Domain-Driven Software

Stefan Hofer
Henning Schwentner

Foreword by Nick Tune

Praise for *Domain Storytelling*

"This book provides a wonderful introduction to an approachable, structured, narrative-based technique for collaborative domain modeling. And for those wanting to go deeper, Stefan and Henning will help you not only to avoid common facilitation pitfalls, but also to integrate the domain knowledge into your everyday development work."

—*Paul Rayner, author of* The EventStorming Handbook

"This book is destined to be the definitive resource on Domain Storytelling for many years."

—*Mike Cohn, co-founder of the Agile Alliance*

"Until now, when people talk about visualization, they usually mean 'words in boxes on a whiteboard.' Representing the user's needs and journeys has been somewhat awkward, with either long form descriptions or series of wireframes. What Stefan and Henning have achieved is a method that shows what's really happening. A Domain Storytelling model shows who's doing what with whom, in what order, and for what purpose, in a clear, truly visual way. It's easy enough to learn how to build these models, but more importantly, an uninitiated reader can understand and critique the models at first sight. That makes Domain Storytelling a powerful communication tool that I believe will become widely used in software product companies and beyond."

—*Mathias Verraes, curator of Domain-Driven Design Europe*

"This is a great addition to any Domain-Driven Design practitioner's bookshelf."

—*Julie Lerman, software coach, The Data Farm*

"All organizations are being disrupted through the rapid advance of change, and my job is to teach people how to apply the Kanban method in their business life. In that context we use Domain Storytelling while exploring and extracting value streams in organizations in a very successful way. With their book, Stefan Hofer and Henning Schwentner explain how collaboration can and does lead the way to transforming our ways of working."

—*Altuğ Bilgin Altıntaş, business agility engineer, accredited Kanban trainer & coach, author of* Kanban Metodu ile Çeviklik*, co-organizer of FlowConf*

"From a story to working software—this book helps you to get to the essence of what to build. Highly recommended!"

—*Oliver Drotbohm*

"This book is a rare achievement, combining a pragmatic guide to a powerful domain modeling technique and a wealth of distilled insights from key aspects of Domain-Driven Design, without being a tome. The authors present a convincing case that conversational stories told and visualized in a natural language pave the fastest path to quality business software. Be prepared for fingers itching to start your own Domain Storytelling while reading the well-curated case studies."

—Xin Yao, chief software architect at Danske Bank

"Practicing Domain Storytelling is a journey towards deep and true understanding of the problem domain you are working on. While discovering subtle inner workings of the business, be prepared for some unexpected solutions to reveal themselves along the way. This book will put you in a position to embark on that journey on your own and will guide you along the way."

—Mufrid Krilic, DDD and Domain Storytelling practitioner

"Domain Storytelling served as a key bridge between our business, products, and technology stacks, and between our past to our future. Using the practice, everyone who participated—from P&L and Operations team leaders to individual engineers and product leads—levelled-up their understanding of where we intended (and needed) to take the business, aligned with each other, and understood how cross-functional product and engineering teams would function within the relevant bounded contexts that collectively represented our future business model. And many (even most!) found it a fun and liberating process.

Domain Storytelling is a practical methodology rooted in the language and context of customers and business, so accessible and valuable to cross-functions (not just engineering) within your business. I recommend the book and, more importantly, the methodology!"

—Jim Banister, chief product officer, Raisin DS GmbH

"As a product manager I really love visualization. Domain Storytelling was one of the techniques I met at the very beginning of my Domain-Driven Design journey (in 2017). I was impressed, amazed, and at the same time surprised in a very positive way that this is exactly what is needed by someone who facilitates the communication between development teams and business. It is very easy to learn and focuses on the pictographic language that makes it possible for literally everyone to understand and take advantage of. I would recommend using it immediately; don't think too much, just start and go with the flow! Believe me it will be worth it. :)"

—Zsófia Herendi, product manager

Domain Storytelling

The Addison-Wesley Signature Series
Continuous Architecture in Practice
Software Architecture in the Age of Agility and DevOps
Murat Erder
Pierre Pureur
Eoin Woods
Strategic Monoliths and Microservices
Driving Change with Purposeful Architecture
Vaughn Vernon
Tomasz Jaskuła
Implementing Strategic Monoliths and Microservices
Patterns and Practices for Continuous Improvement
Vaughn Vernon
Tomasz Jaskuła
Principles of Web API Design
Applying Domain-Driven Design to APIs and Microservices
James Higginbotham

Domain Storytelling

A Collaborative, Visual, and Agile Way to Build Domain-Driven Software

Stefan Hofer
Henning Schwentner

Addison-Wesley

Boston • Columbus • New York • San Francisco • Amsterdam • Cape Town
Dubai • London • Madrid • Milan • Munich • Paris • Montreal • Toronto • Delhi • Mexico City
São Paulo • Sydney • Hong Kong • Seoul • Singapore • Taipei • Tokyo

Library of Congress Control Number: 2021943443

ISBN-13: 978-0-13-745891-2
ISBN-10: 0-13-745891-6

1 2021

To our families.

This page intentionally left blank

Contents

Domain Stories

Series Editor Foreword

My signature series emphasizes organic growth and refinement, which I describe in more detail below. First, there's a story of organic growth connected with this particular book.

My book, *Implementing Domain-Driven Design*, known as "the red book," arrived at an important point in time. Prior to my red book there were only a handful of people with an accurate and thorough understanding of this advanced approach to software development—those who could be described as leaders. Related to this, few Domain-Driven Design (DDD) meetups and tools to help practitioners existed at that time. I co-founded the DDD Denver meetup in 2011, well before my red book was published in 2013, but that was one of perhaps five or so to be found anywhere. By 2021, you could find 142 Domain-Driven Design meetup groups globally, with 93,171 members, and they are still growing. When that number reached 10, and then 20, and then 25, did anyone think that they would eventually find nearly 150? As a result of this organic growth, the number of leaders and supporting tools provided for DDD also grew. Timing is everything, and I am thrilled to have contributed several catalysts at times when they were needed most, which led to the expansion of DDD. Before explaining how this book and its authors are involved in this organic growth, first consider the series that hosts it.

My signature series is designed and curated to guide readers toward advances in software development maturity and greater success with business-centric practices. The series emphasizes *organic growth and refinement* with a variety of approaches—reactive, object, as well as functional architecture and programming; domain modeling; right-sized services; patterns; and APIs—and covers best uses of the associated underlying technologies.

From here I am focusing now on only two words: *organic refinement*.

The first word, *organic*, stood out to me recently when a friend and colleague used it to describe software architecture. I have heard and used the word *organic* in connection with software development, but I didn't think about that word as carefully as I did then when I personally consumed the two used together: *organic architecture*.

Think about the word *organic*, and even the word *organism*. For the most part these are used when referring to living things, but they are also used to describe inanimate things that feature some characteristics that resemble life-forms. *Organic* originates in Greek. Its etymology is with reference to a functioning organ of the body. If you read the etymology of *organ*, it has a broader use, and in fact organic followed suit: body organs, to implement, describes a tool for making or doing, a musical instrument.

We can readily think of numerous organic objects—living organisms—from the very large to the microscopic single-celled life-forms. With the second use of organism, though, examples may not as readily pop into our mind. One example is an organization, which includes the prefix of both *organic* and *organism*. In this use of *organism*, I'm describing something that is structured with bidirectional dependencies. An organization is an organism because it has organized parts. This kind of organism cannot survive without the parts, and the parts cannot survive without the organism.

Taking that perspective, we can continue applying this thinking to nonliving things because they exhibit characteristics of living organisms. Consider the atom. Every single atom is a system unto itself, and all living things are composed of atoms. Yet, atoms are inorganic and do not reproduce. Even so, it's not difficult to think of atoms as living things in the sense that they are endlessly moving, functioning. Atoms even bond with other atoms. When this occurs, each atom is not only a single system unto itself, but also becomes a subsystem along with other atoms as subsystems, with their combined behaviors yielding a greater whole system.

So then, all kinds of concepts regarding software are quite organic in that nonliving things are still "characterized" by aspects of living organisms. When we discuss software model concepts using concrete scenarios, or draw an architecture diagram, or write a unit test and its corresponding domain model unit, software starts to come alive. It isn't static, because we continue to discuss how to make it better, subjecting it to refinement, where one scenario leads to another, and that has an impact on the architecture and the domain model. As we continue to iterate, the increasing value in refinements leads to incremental growth of the organism. As time progresses so does the software. We wrangle with and tackle complexity through useful abstractions, and the software grows and changes shapes, all with the explicit purpose of making work better for real living organisms at global scales.

Sadly, software organics tend to grow poorly more often than they grow well. Even if they start out life in good health, they tend to get diseases, become deformed, grow unnatural appendages, atrophy, and deteriorate. Worse still is that these symptoms are caused by efforts to refine the software that go wrong instead of making things better. The worst part is that with every failed refinement, everything that goes wrong with these complexly ill bodies doesn't cause their death. (Oh, if they could just die!) Instead, we have to kill them and killing them requires nerves, skills, and the intestinal fortitude of a dragon slayer. No, not one, but dozens of vigorous dragon slayers. Actually, make that dozens of dragon slayers who have really big brains.

That's where this series comes into play. I am curating a series designed to help you mature and reach greater success with a variety of approaches—reactive, object, and functional architecture and programming; domain modeling; right-sized services; patterns; and APIs. And along with that, the series covers best uses of the associated underlying technologies. It's not accomplished at one fell swoop. It requires organic refinement with purpose and skill. I and the other authors are here to help. To that end, we've delivered our very best to achieve our goal.

That's why I chose this book, *Domain Storytelling*, to be among mine and others in my series. The previously noted organic expansion of DDD has resulted in new practitioners and leaders, as well as innovation in tools that support it. Collaborative modeling with this brilliant tool enables visual exploration while simultaneously capturing domain-driven discoveries and model usage scenarios with vital clarity that leads to increased success. Domain Storytelling should not be viewed as a replacement for previous tools, but an opportunity to gain a greater variety of instruments of knowledge acquisition. Novel and more challenging modeling situations call for an array of useful tools that can be employed together.

With this new book, Stefan Hofer and Henning Schwentner are establishing themselves as two of the new leaders. They have brought us an additional tool in Domain Storytelling, to be used for wrangling with greater complexities that we face today and will continue to face over the years ahead. That's organic.

—Vaughn Vernon, series editor

This page intentionally left blank

Foreword

In 2004, Eric Evans published *Domain-Driven Design*, a timeless book that has become an all-time classic in software engineering literature. Evans projected his vision of software developers as people who collaborated closely with subject matter experts to iteratively solve domain-related problems for users. At the time, this was heresy against mainstream practices oriented around data models, big up-front planning, and programmers as mere order-takers.

Evans's text was a masterpiece, but still it was missing something. For a decade, DDD was perceived by the mainstream as a few programming patterns and became synonymous with over-engineering. Evans's book spoke frequently about domain experts and technical experts crunching domain knowledge together, but it didn't give readers enough practical guidance in the same way it did for the technical DDD patterns.

The mid-2010s saw a DDD renaissance, which continues to this day. Vaughn Vernon's book *Implementing Domain-Driven Design* was pivotal in correcting many misconceptions and making DDD more approachable. And a new generation of practitioners led by Alberto Brandolini, including Stefan and Henning, emphatically added the missing piece of the DDD puzzle by introducing new collaborative modeling techniques into the community. The mainstream perception of DDD is now just as much sticky notes on the wall as it is programming design patterns. Evans's 2004 prophecy is truly now a reality.

Domain Storytelling stands out for its pictographic, structured, and scenario-based nature. But this book is far more than a guide to Domain Storytelling. Stefan and Henning are passionate, intelligent, and experienced collaborative domain modelers. This book takes you into their brains through their thinking patterns and deep into the principles of collaborative domain modeling and workshop facilitation. This work provides insights that will be useful regardless of the technique you decide to use and regardless of how much you know about DDD. It may even inspire you to invent the next generation of collaborative modeling techniques.

I've had the chance to meet Stefan and Henning at numerous conferences. But one sticks out in my memory more than others: Explore DDD 2018 in Denver. Their enthusiasm for Domain Storytelling captivated the audience (including me) in a talk where they presented a case study of using Domain Storytelling to build systems that prevent ships getting stranded in the port of Hamburg. Their role play with props was the icing on the cake. I also got to attend their hands-on Domain Storytelling workshop where their pure love of the game shone through and had attendees (including me) excitedly modeling the cinema experience.

I also have fond recollections of the night before the main conference. Eric Evans gave an evening keynote to officially kick off Explore DDD, and I was with Stefan and Henning at the social event afterwards. They made an effort to help conference attendees feel included by inviting them into discussions and striking up conversations, and they made us all belly laugh with their clever humor.

I hope this book provides you with the inspiration, enthusiasm, and smiles that Stefan and Henning have brought to me and many others.

—*Nick Tune*

Preface

Misunderstandings between software developers and people from the business departments are a common problem. Bad communication is a plague that makes projects fail. Domain Storytelling is a remedy because this technique transforms domain knowledge into effective business software. Domain Storytelling brings together domain experts, software developers, user experience designers, product owners, product managers, and business analysts on the same page. They learn from each other by telling stories and drawing them as easy-to-understand pictures.

This Book in the Software Development Landscape

This book is meant to be a practical guide. We wanted to keep it concise, although we were tempted to write a book on Domain-Driven Design, domain modeling, requirements, agile, and software development in general. All of these topics are relevant to Domain Storytelling and are referenced in this book, but we tried not to dive too deep into topics that other authors have already addressed brilliantly. If you are curious, here is an (incomplete) list of authors who have influenced us:

- **Heinz Züllighoven et al.:** The *Tools and Material Approach* [Züllighoven 2004] is important for how we look at software development. It promotes user-centric software development and is built around the powerful metaphors *tool* and *material*. The book was first published in the late 1990s and covers topics such as iterative-incremental development, modeling, scenarios, and patterns for software construction.

- **Eric Evans:** *Domain-Driven Design* (*DDD*) [Evans 2004] shares many characteristics with the Tools and Material Approach, although the two were developed independently from each other. The concepts *bounded context* and *ubiquitous language* immediately resonated with us. DDD opened a new area of application for Domain Storytelling, influencing the direction in which we have been pushing the method in the last couple of years.
- **Kent Beck et al.:** The *Agile Manifesto* [Beck et al. 2001] put "individuals and interactions over processes and tools" and "working software over comprehensive documentation." Domain Storytelling is about individuals and their interactions on two levels: (1) domain stories show people and their collaboration, and (2) the workshop in which these stories are told brings together humans to interact with each other. *Extreme Programming* (*XP*) brought the concept of stories into software development and introduced *user stories* to the agile world [Beck 1999, Beck/Andres 2004, and Cohn 2004].
- **Alistair Cockburn:** *Writing Effective Use Cases* [Cockburn 2001] is one of our favorite books on requirements. Even if you do not apply use cases, the book is worth a read. It shows how to follow an agile path from domain to software. We borrowed the idea of *goal levels* and applied it to Domain Storytelling.
- **Gojko Adzic:** *Specification by Example* [Adzic 2011] highlights the importance of conversations in software development. We started to view software development as a series of conversations about the domain and requirements, supported by several "conversation methods" like Domain Storytelling. We share Adzic's views on requirements and the benefits of using examples.
- **Christiane Floyd:** We had the pleasure of learning from Professor Floyd at the University of Hamburg. She researched and taught (among other things) modeling theory, the limits of modeling, sociotechnical aspects of modeling and software development, and participatory design. She published papers in German and English. If you are curious, see for example "Software Development as Reality Construction" [Floyd 1992] and search for papers on her *STEPS* approach, an early example of what later became known as agile software development.

What This Book Covers

This book is divided into two parts: Part I explains the method, and Part II describes how to use and adapt it for specific purposes.

Part I, "Domain Storytelling Explained": This part of the book contains everything you need to know to use Domain Storytelling for learning about a domain.

Chapter 1, "Introduction": This chapter explains what Domain Storytelling actually is. Also, we introduce a case study that will give you a first impression of the method and show you why it is useful.

Chapter 2, "The Pictographic Language": This chapter describes the graphical notation. To record domain stories visually, you need a set of symbols and rules for combining them. This chapter will also show you what we consider good language style and what pitfalls to avoid. If you have never tried Domain Storytelling, you should do so after finishing Chapter 2. Get out a piece of paper and a pen, and model a workflow that you are familiar with. However, before trying Domain Storytelling in a workshop situation, you should continue reading the rest of Part I.

Chapter 3, "Scenario-Based Modeling": This chapter introduces you to one of the major differences between domain stories and other business process modeling languages: Every domain story is about one case. You will learn which cases you should model and how to keep an overview.

Chapter 4, "Scope": This chapter is about the level of detail that stories have, whether they are descriptive or exploratory, and the amount of technical information they contain. You will need to consider all these factors every time you start a new domain story. This chapter will help you to choose the right scope.

Chapter 5, "Modeling Tools": This summarizes our experience with different modeling tools. We recommend which tools are useful in which situations, describe their strengths and weaknesses, and give practical tips that will help you model more effortlessly.

Chapter 6, "The Workshop Format": This chapter shows that Domain Storytelling works best when used for collaborative modeling. You will learn how to prepare, conduct, and follow up on a workshop. This chapter will help you to become a good moderator.

Chapter 7, "Relationship to Other Modeling Methods": This chapter discusses other modeling methods and workshop formats and how to combine Domain Storytelling with them. This chapter will help you to pick the right tool for the right job.

Part II, "Purposes": This part of the book deals with the different problems and purposes for which Domain Storytelling can be used. Even though we use the same example in all chapters of Part II, you do not have to read the chapters in order. Just pick the purposes that you are interested in.

Chapter 8, "Case Study—Alphorn Auto Leasing Inc.": This chapter introduces a second, more comprehensive case study.

Chapter 9, "Learning Domain Language": Speaking the language of the domain experts is the key for effective conversations about business processes and software requirements. This chapter is for you if you are new to a domain, if the software that you work on does not use real terms from the domain and you want to change that, or if you work in an organization where no real domain language exists and you want one to emerge.

Chapter 10, "Finding Boundaries": Many domains are too big to be understood and modeled as a whole. You need to break down a domain into manageable units. Read this chapter if you are struggling with a monolith and want to reorganize it or split it into more manageable parts, if you want to design microservices, or if you want to apply Domain-Driven Design and have difficulties identifying bounded contexts. The chapter will also help if your development team has become too big to work efficiently or if you already have more than one development team and want to find out how you can organize the work for these teams.

Chapter 11, "Working with Requirements": How do you bridge the gap between domain knowledge and requirements? We show you how to derive requirements from domain stories so that you can discuss priorities and viable products. This chapter is for you if you consider yourself a product owner, product manager, business analyst, requirements engineer, or developer in a cross-functional team that does its own requirements analysis.

Chapter 12, "Modeling in Code": If your ultimate goal is to develop software, then, at some point, you need to move from modeling with diagrams and sticky notes to modeling in programming languages. This chapter will show you how to transition from visual modeling to code.

Chapter 13, "Supporting Organizational Change": The goal of a new software system usually is to make work easier, faster, and more efficient (in short: better). This goal will not be reached by digitalizing bad manual processes. Neither will a pile of requirements magically turn into a seamless business workflow. To build good business software, you need to go beyond merely modeling the current situation. You will need to design the future way of working. Domain stories help to do this and visualize how new software will change the way people work. Read this chapter if you want to optimize business processes, roll out new software, or discuss and promote change in business processes.

Chapter 14, "Deciding Make or Buy and Choosing Off-the-Shelf Software": Not every piece of software is custom-built. Many domains are supported by off-the-shelf software. Domain stories can help to decide if a new software system should be developed or bought. If the decision is to buy an existing solution, usually several vendors will offer their products. Here too, domain stories can be useful in making a decision.

Chapter 15, "Finding Shadow IT": When you are trying to consolidate software application landscapes or promote digitalization, shadow IT stands in your way. Every company beyond a certain size uses software that the central IT department is not aware of. All those little solutions that run in business departments and that hardly anyone knows about are often business-critical. Domain experts use shadow IT unconsciously, so it is easily overlooked. Domain stories can help IT and management find this shadow IT and see the whole IT landscape.

Chapter 16, "*Conclusion*": We take a look into the future of Domain Storytelling and sum up its essence.

Conventions

When we define a new term, we set it in bold, e.g., **actor**. Terms that were defined by other authors are set in italics when we use them for the first time, e.g., *bounded context*. Scope factors appear in small caps, e.g., COARSE-GRAINED. Words from case studies are surrounded by quotation marks, like "moviegoer." Code is set in constant width, like the class `MovieTicket`.

Important notes are marked with a lightbulb icon: 💡. When we give concrete advice, our tips are marked with a checkmark icon: ✔.

We provide examples in the form of case studies and "stories from the trenches." Appearances of case study Alphorn Auto Leasing Inc. are indicated with a car icon: 🚗. The stories from the trenches are marked with a sunflower icon: 🌻.

Legend for "Opening Stories"

As modelers, we couldn't resist the temptation to describe Domain Storytelling itself with domain stories. That's why several chapters of this book begin with an "opening story"—a short domain story that explains what the chapter is about. The legend in Table P.1 will help you to interpret the icons that we use in those stories.

Table P.1 *Icons used in "Opening Stories"*

Icon	Meaning
🪪	A domain story as a diagram
⧉	Any other kind of diagram
⊞	Building blocks of a domain story diagram
⋖	Activities of a domain story diagram
📄	A text such as a user story or a source code file, etc.

Supplementary Materials

On www.domainstorytelling.org/book, we created a companion website for this book [DomainStorytelling BookWebsite]. It contains the example domain stories that we show in this book. The examples were created with Egon.io, an open-source modeling tool that our company WPS – Workplace Solutions created [Egon.io Website]. You can import the source files from this book into Egon.io and "replay" the domain stories (instructions are on the website).

The companion website also contains the bibliography of this book, with clickable links to online resources.

Furthermore, on www.domainstorytelling.org, you will find links to useful materials, including a collection of videos and articles [DomainStorytelling Website].

About the Cover

The covers of the books in Vaughn Vernon's Signature Series all use images that have an organic theme. We were happy about this because domain stories develop in an organic way, too. Like the sunflowers you see on the cover, a domain story starts from a small seed, grows, and if everything goes well, eventually blooms into a beautiful blossom. The relationship of domain and domain story is similar to the one between sun and sunflower:

- A sunflower can't exist without sun. Likewise, a domain story will dry up without contact to the domain.

- The sunflower looks like the sun. Likewise, the domain story is formed to resemble the domain.
- During the day, the sunflower turns its head to follow the sun. Likewise, the domain story follows the domain.
- A sunflower often lives in a field together with many other sunflowers. Likewise, a domain story usually doesn't come alone but together with other stories.

All these properties make the sunflower not only a splendid cover picture plant but also a friendly-looking symbol for Domain Storytelling.

With that out of the way, welcome to The Sunflower Book. We wish you happy reading and happy modeling!

Acknowledgments

Many people have encouraged us to write this book and helped us improve it by giving valuable feedback. We would like to thank all of you!

We thank our friends and colleagues for sharing their experience and stories with us, especially (and in alphabetic order) Jörn Koch, Dorota Kochanowska, Carola Lilienthal, Carsten Lill, Kai Rüstmann, Martin Schimak, and Heinz Züllighoven. Heinz deserves an especially big, big *thank you* for his endless support. Isabella Tran and Samaneh Javanbakht helped us with their language skills.

Carola Lilienthal and Guido Gryczan, CEOs of WPS – Workplace Solutions GmbH, have been our managers for more than a decade and a half now. Their continued support made Domain Storytelling possible in the first place.

Despite coining the name and writing the first book about it, we did not invent Domain Storytelling. Rather, it emerged organically out of another modeling method. The seed was planted in the 1990s at the University of Hamburg, and many people have helped to cultivate it. They deserve credit and our gratitude. That is why we dedicated the appendix, "The History of Domain Storytelling," to them.

An earlier version of this book was first published as an e-book on Leanpub, a self-publishing platform. We thank everyone who downloaded our book. Every download increased our confidence and motivated us to turn our original ~40-page booklet into a proper book. We appreciate the useful tips that Cyrille Martraire gave us when we transitioned to Pearson.

This book could not have been written without the power of open-source software. Pandoc and Perl enabled us to write the source in Markdown. We used Inkscape and Draw.io for many illustrations. A hearty thank-you to the authors of these great tools.

We thank Altuğ Bilgin Altıntaş, Jim Banister, Mike Cohn, Oliver Drotbohm, Zsófia Herendi, Mufrid Krilic, Julie Lerman, Paul Rayner, Mathias Verraes, and Xin Yao for contributing endorsements for our book. They represent several software communities that we care about deeply. The community of Domain-Driven Design practitioners has been crucial to the success of Domain Storytelling. We want to thank all the kind and welcoming organizers of conferences and meetups for giving us a platform to discuss our ideas—especially Mathias Verraes for giving us a first international stage at DDD Europe and for producing a printed Domain Storytelling booklet with us.

We are grateful for the trust that Vaughn Vernon has put in us. Being part of his *Addison-Wesley Signature Series* is an honor. Nick Tune wrote an awesome foreword, both inspiring and personal. We hope that our paths will cross many times in the future, like they did in Denver in 2018.

Writing a text is one thing, but making a book out of it is another. That's why we would like to thank the people at Pearson we've had the pleasure to work with. Our executive editor Haze Humbert and content producer Julie Nahil guided us competently through the development and production process. Development editor Adriana Cloud improved the flow and clarity of our text while preserving our voice. Matt Takane provided us with a very helpful review. Copy editor Kim Wimpsett patiently endured our stubbornness and fine-tuned our text. The book was composed by Aswini Kumar and her team.

Writing this book took a lot of time—uncounted evenings and weekends that our loved ones had to spend without us. Stefan sends gratitude and love to his wife, Samaneh. Henning sends gratitude and love to his children, Lennart, Joris, Friedo, Tara, and Aurelia, and to his partner, Julia. Dear families, we thank you for your patience, understanding, and support!

About the Authors

Stefan Hofer is bad at drawing. However, he thinks he can build up domain knowledge by drawing domain stories. Stefan studied software engineering in Austria and earned a PhD in computer science. Since 2005, he has been working for WPS – Workplace Solutions in Hamburg, Germany. His job there is to help teams develop software that does the right job the right way. He maintains domainstorytelling.org. You can reach him on Twitter (@hofstef) or by email to stefan@domainstorytelling.org.

Henning Schwentner is a programmer who has been into computers ever since he got an Amiga 500 in the early 90s. He was lucky enough to turn this passion into a profession and works as a coder, coach, and consultant at WPS – Workplace Solutions. He helps teams to bring structure into their existing software or to build new systems with a sustainable architecture from scratch. Henning is the author of LeasingNinja.io, the German translator of *Domain-Driven Design Distilled*, and co-organizer of CoMoCamp. He tweets as @hschwentner and reads emails addressed to henning@domainstorytelling.org. Henning is the proud father of five children in a very special patchwork situation.

Part I

Domain Storytelling Explained

This part covers everything you need to know to get started. You will learn the following:

- Why Domain Storytelling is useful to organizations, business experts, and development teams
- How to prepare and run workshops in which domain stories are discussed and modeled visually
- How to choose which examples to use for domain stories (and how to deal with everything else that could happen)
- How to decide on the level of detail and other properties of a domain story
- Which options in modeling tools you have
- How to combine Domain Storytelling with other methods and when to choose one over the other

After reading this part and some practice, you will be prepared to moderate Domain Storytelling workshops.

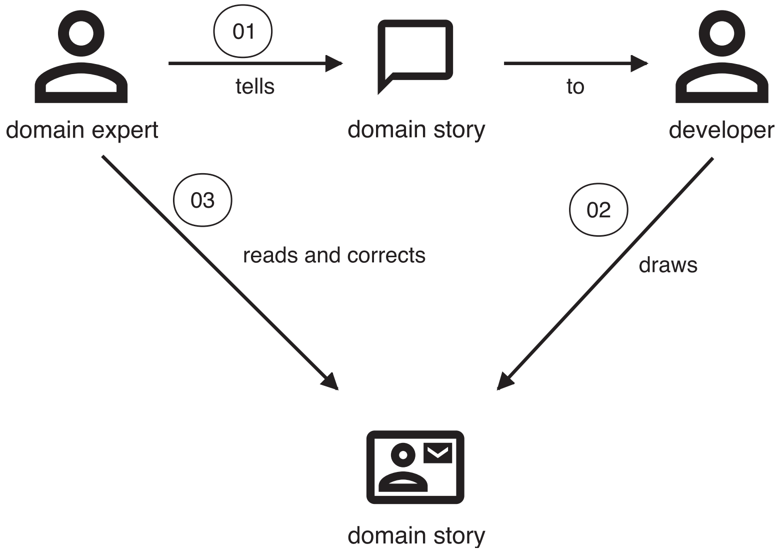
Chapter 1

Introduction

Spoken language is deeply, deeply human. Some things must be said that cannot be written.

—Avraham Poupko [Poupko 2018]

What Is Domain Storytelling?



Domain Storytelling is a collaborative modeling technique that highlights how people work together. Its primary purpose is to transform domain knowledge into business software. This purpose is achieved by bringing together people from different backgrounds and allowing them to learn from each other by telling and visualizing stories.

Telling stories is a basic form of human communication. It is deeply rooted in all of us since the times our ancestors lived in caves.[1] In our modern world, telling a story might seem archaic or childish. How can an activity so informal help us to build business-critical software for domains such as logistics, car manufacturing, e-commerce, and banking?

We believe that conversations cannot be adequately replaced by written, formal specifications. Attempts to do so have even widened the gap between business and software development. But that is not just our personal opinion. Consider software development approaches like *agile*, *Domain-Driven Design*, or *Behavior-Driven Development*. These philosophies focus on feedback and stakeholder involvement. Nevertheless, making great business software is hard, but rarely is this because of technical problems. So why then? Because software developers need to understand how the day-to-day business operates. They need to become domain experts themselves—not for the whole domain but at least for the part they build software for. As Alberto Brandolini put it:

> It's developer's (mis)understanding, not expert knowledge, that gets released in production. [Brandolini 2016]

Telling stories still works in the age of software. In our experience, telling and listening to stories helps with the following:

- Understanding a domain
- Establishing a shared language between domain experts and IT experts
- Overcoming misunderstandings
- Clarifying software requirements
- Implementing the right software
- Structuring that software
- Designing viable, software-supported business processes

Telling stories is a means for transporting domain knowledge from the heads of domain experts into the heads of developers, testers, product owners, product

1. See *The Desirability of Storytellers* [Yong 2017].

managers, business analysts—anyone who is involved in developing software. Of course, we do not sit around campfires in dark and damp caves anymore. We share our stories while we meet in front of a whiteboard in a workshop. The domain experts are our storytellers. We want them to tell us the true stories from the trenches—no abstract "ifs," no hypothetical "coulds." We want concrete and real examples of what actually happens in the domain. We want *domain stories*.

You can learn more from a good example than from a bad abstraction.

Once, storytelling was an oral activity. Domain Storytelling is an oral *and* visual activity, a form of *modeling*: While the domain experts tell their story in spoken language, one of the workshop's participants—the moderator—records the story as a diagram made of simple icons, arrows, and text. This way the participants get another representation of the story, which helps to uncover misunderstandings, contradictions, and plot holes. All participants see how the visual recording evolves together with the story. This makes it easy to give feedback and to contribute.

And that is Domain Storytelling.

Your First Domain Story

Matthew runs a small movie theater for arthouse films—called Metropolis—that enjoys an excellent reputation among cineastes. Local craft beer and organic snacks round off the cinema experience. One day Matthew meets his school friend Anna. When he learns that Anna has been developing apps for almost ten years, he gets an idea.

Movie theater manager Matthew: "My customers like the old-fashioned charm of my cinema. But they do not like my old-fashioned box office. Today's moviegoers are not used to buying tickets in person at the box office anymore. Customers have been asking me to sell tickets online. Can you develop an app for me?"

App developer Anna: "You run just one cinema, and there's only a handful of movies per week, two or three shows a day. Sounds easy."

Matthew: "Great! But one small thing: We also show international movies in foreign languages in our program. Also, in addition to the online sales through the app, I still need the box office for less tech-savvy moviegoers. And I'd like users of the app to be able to sign up for a yearly subscription."

Anna: "Subscriptions? Online and offline sales? Shows in foreign languages? That's more complicated than I thought...."

The Workshop Begins

The next day they meet again in Matthew's office. They are standing in front of a whiteboard, and Anna is holding a marker in her hand.

App developer Anna: "Yesterday you said that the app essentially has three use cases: One: selling standard tickets; two: selling special tickets for foreign-language movies; and three: signing up for a yearly subscription."

Movie theater manager Matthew: "Uh, yes, that's right."

Anna: "I would like to understand how Metropolis operates today. That will help me to develop an app that meets your requirements. Could you please explain to me how you sell tickets at the box office?"

Matthew: "Sure. You sell the tickets and mark the seat in the seating plan and...."

Anna: "Wait a minute. Who sells the tickets?"

Matthew: "I have two students working for me. But sometimes I do it myself."

Anna: "Okay, but what role do you or the students have then?"

Matthew: "Cashier."

Anna draws a stick figure on the whiteboard and writes "cashier" underneath (see Figure 1.1).

Figure 1.1 *The first actor*

Anna: "Who buys the tickets?"

Matthew: "A moviegoer. One without a subscription."

Anna draws a second stick figure and calls it "moviegoer." Next to it, she writes down that the moviegoer has no subscription (see Figure 1.2).

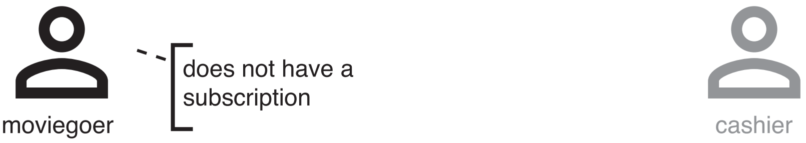


Figure 1.2 *The second actor and the first annotation*

Anna: "What does a moviegoer have to do to buy a ticket?"

Matthew: "They tell the cashier which show they want to see."

Anna: "I will draw a speech bubble as an icon for the show here, because the two of them talk to each other."

Anna continues drawing and numbers the arrow (see Figure 1.3).

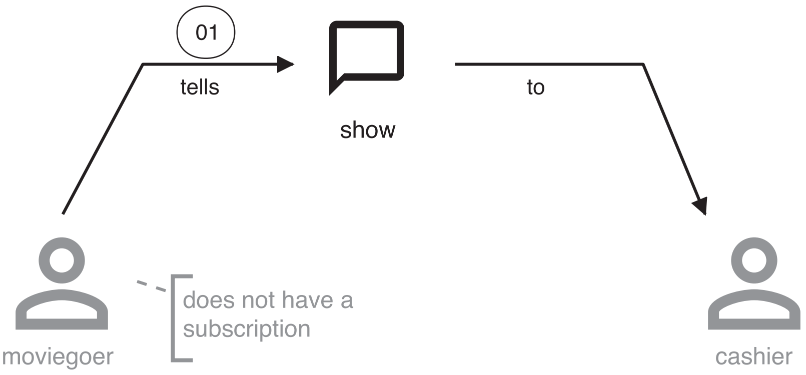


Figure 1.3 *The first activity*

Anna: "And then?"

Matthew: "Usually the cashier suggests the best seat available."

Anna: "Ah, so the moviegoer picks a seat in advance! How does the cashier suggest seats?"

Matthew: "I take the seating plan for the show and search for available seats. In the seating plan, I can see which seats have already been sold and which are still available."

Anna draws and explains the icons.

Anna: "Here, I'm using a film icon instead of the speech bubble to symbolize the show."

Matthew: "The seating plan is a grid. Can you draw a grid?" (See Figure 1.4.)

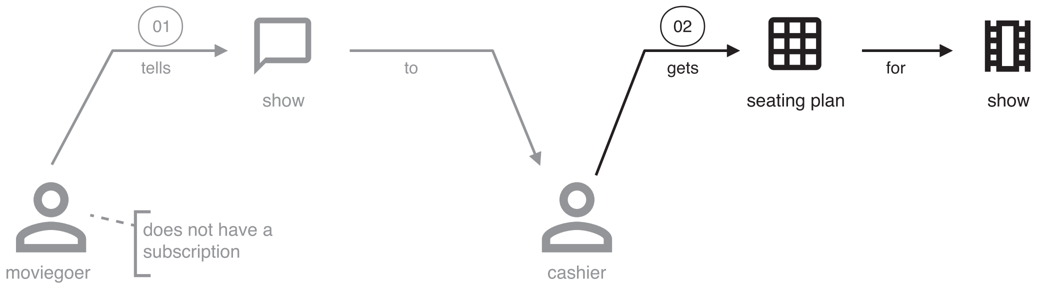


Figure 1.4 *The second activity*

Then, Anna reads back what she has understood.

Anna: "Second, the cashier gets the seating plan for the show. Third, they search for available seats. Is that OK?" (See Figure 1.5.)

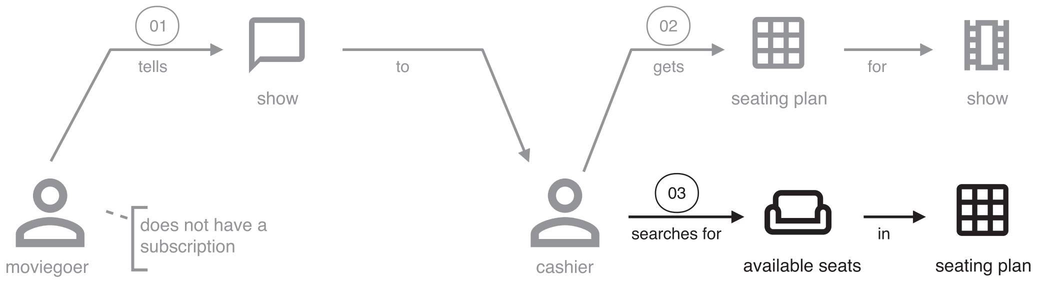


Figure 1.5 *The third activity*

Matthew nods in agreement.

Anna: "And now the cashier suggests the available seats to the moviegoer?"

Matthew: "Exactly."

Anna: "I will move the annotation 'does not have a subscription' up a bit to get enough space for that fourth sentence." (See Figure 1.6.)

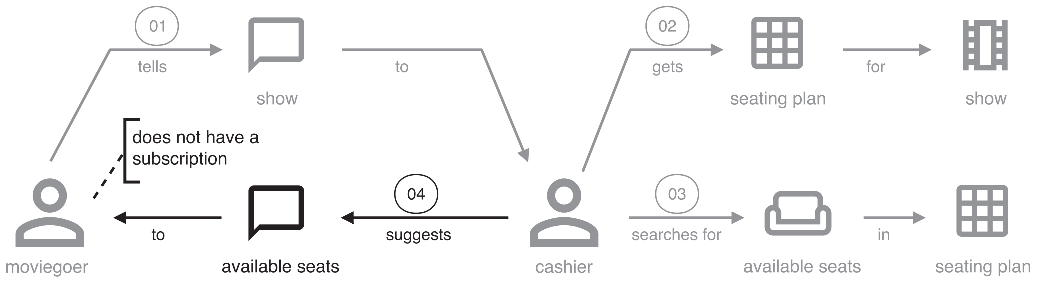


Figure 1.6 *The fourth activity*

The discussion continues….

Retelling the Story

Within a few minutes, the whiteboard is filled with a story about a moviegoer who buys tickets from a cashier at the box office. Icons and arrows are rearranged during the session. Finally, Anna retells the story from the beginning (see Figure 1.7).

Matthew: “Yes, that’s right. But I forgot about the international movies.”

Anna: “You mean the shows in a foreign language? I thought you sell special tickets for those.”

Matthew: “No, no! We usually show the movies in English. When it is a foreign movie, we also show it in its original language. We don’t sell extra tickets; you only have to point out to the moviegoers which language the movie will be shown in.”

Anna: “When does the cashier do that?”

Matthew: “Here.”

Matthew points to the arrow with the number 4. Anna amends the sentence “Cashier suggests available seats to moviegoer” with a comment “and mentions language” (see Figure 1.8).

Anna: “It seems we are finished with our little ‘Going to the movies’ story. Of course, we have looked only at the best possible outcome; I call this the ‘happy path.’ I will ask you about other cases later.”

Matthew: “OK.”

Since Matthew does not have any further remarks, Anna takes a picture of the whiteboard with her smartphone and moves on.

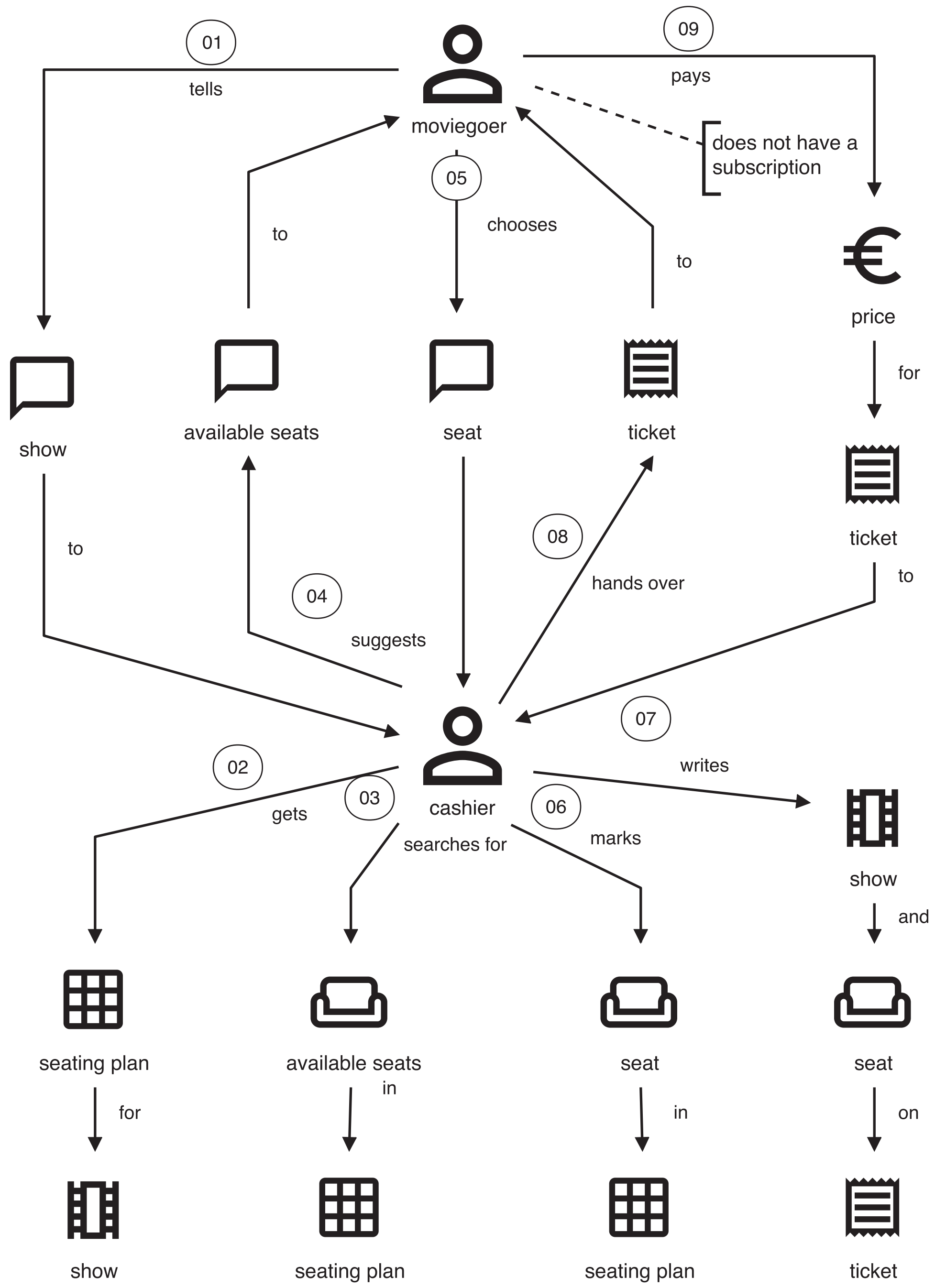


Figure 1.7 *The whole story*

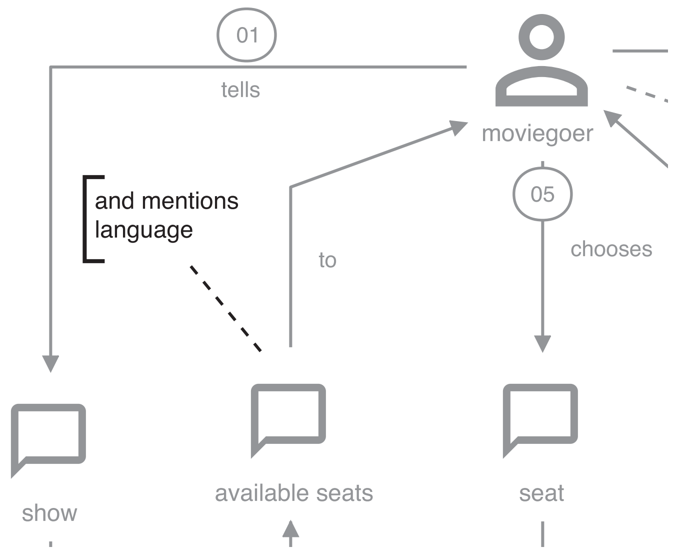


Figure 1.8 *Adding another annotation*

Exploring Further

Anna: "Once a moviegoer has bought a ticket, what do they do with it?"

Matthew: "They go to the entrance of the theater where the usher is waiting and…."

Anna turns the whiteboard around, and Matthew tells her how the usher checks tickets.

After a few FINE-GRAINED Metropolis domain stories, Anna has gained a good insight into the cinema domain. She knows terms like "seating plan," "show," "cashier," "to search for available seats," and "to mark seats." She has an initial understanding of the most important processes.

Anna realizes that it would be helpful to have an overview of the processes—a "big picture" that holds all the stories together. Anna and Matthew decide to model a COARSE-GRAINED "Going to the movies" story (see Figure 1.9).

With the knowledge about the purpose of the app and its context, she can think about how the app will work and how the processes will change.

Summary and Outlook

This section was called "Your First Domain Story"; what you have probably noticed is that this was also your first Domain Storytelling workshop.

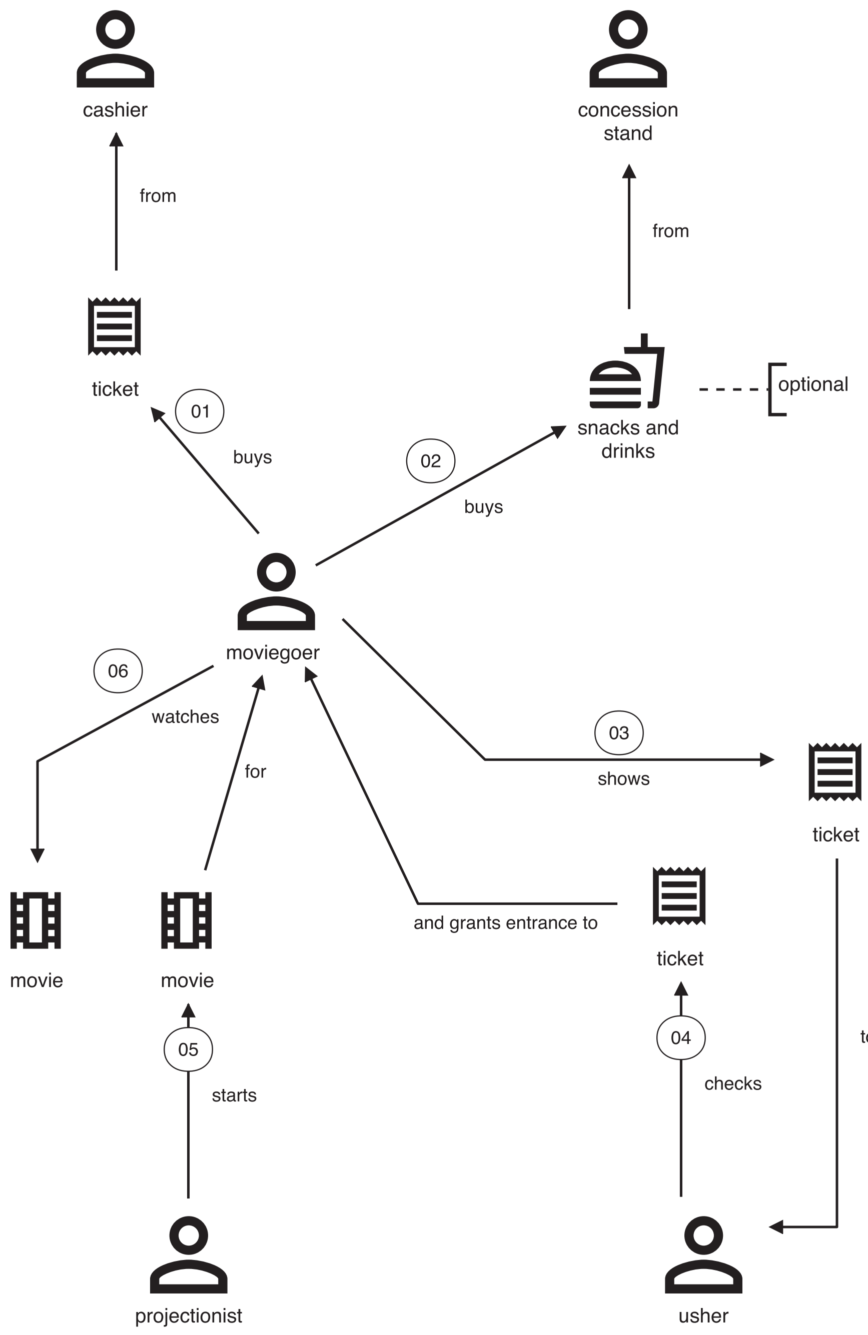


Figure 1.9 *Metropolis 1: Going to the movies*—COARSE-GRAINED

For Anna, this was surely not the first time she applied Domain Storytelling. Why was this technique useful for Anna in this situation? First, Matthew was not able to explain to her what he wanted. He had an idea in his head but had not really thought about how to make it actionable. To start a conversation about Matthew's idea, Anna first had to establish common ground: Which terms does Matthew use when he talks about his business, and what do they mean? Which business processes are relevant, and what are the important steps in those processes? Who is involved in those processes?

From that starting point, they will be able to discuss what's in scope of the project and what's not. Without Domain Storytelling, they likely would have misunderstood each other. For example, Anna initially thought that selling tickets for foreign language movies is a separate process.

Since we will refer to the Metropolis example in the following chapters, we will give the domain stories the names "Metropolis 1" (see Figure 1.9) and "Metropolis 2" (see Figure 1.10) for easier reference.

We want to highlight a few points that came up in this first example:

- Even with just a few domain stories, you can learn a lot about a domain, its language, and its business processes.
- Domain stories use a simple pictographic language to show people, their activities, and their interactions. This helps to uncover hidden assumptions and misunderstandings.
- While a domain story is told, its picture will evolve and change accordingly.
- It's not just about drawing a nice picture; it's also about bringing people together.
- Domain stories can vary in granularity.
- Usually, you will model more than one story in a workshop.
- A domain story has no "ifs" and "ors." Instead, you model only the most important alternatives—each one as a separate domain story.
- A simple whiteboard is a good enough tool for drawing domain stories.

We will cover all these points in detail in the following chapters. The first point we will discuss is the pictographic language.

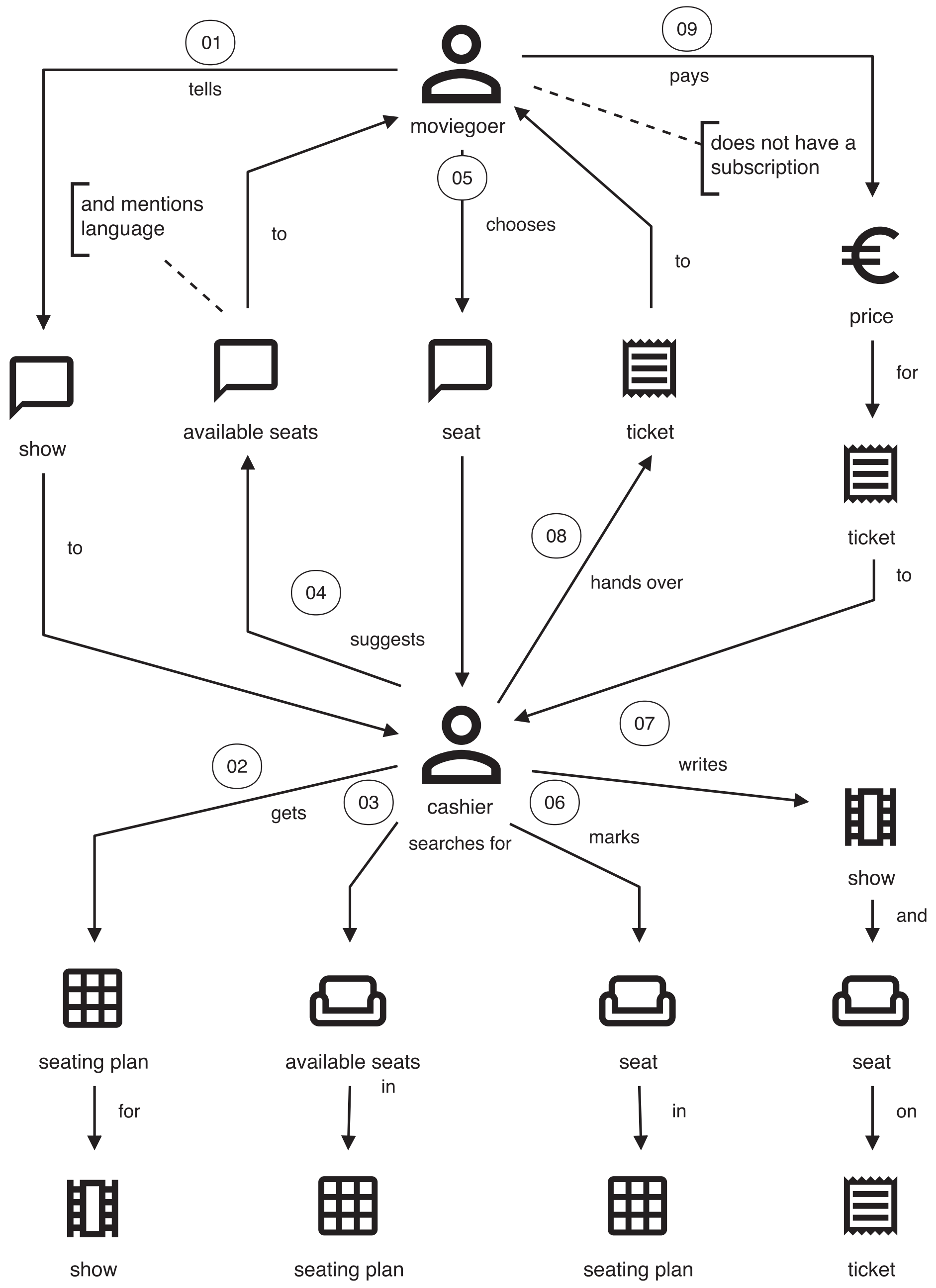


Figure 1.10 *Metropolis 2: Ticket sales, happy path*—FINE-GRAINED

Index