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*

A Useful Metaphor for Granularity

Distinguishing different levels of granularity with terms like FINE-GRAINED and COARSE-GRAINED has limited expressiveness. In *Writing Effective Use Cases*, Alistair Cockburn introduces a metaphor for granularity that can also help with domain stories [Cockburn 2001]. Cockburn defines so-called *goal levels* by using the ocean as a metaphor (see Figure 4.2). You start at *sea level*. A sea-level goal describes something a user wants to achieve. When you get more COARSE-GRAINED, you ascend to *kite* and *cloud level*, which are summaries of sea-level goals. Going more FINE-GRAINED, you descend below sea level to *fish level*. Finally, at the bottom of the ocean lies the *clam level* (which Cockburn also calls "too low").

Figure 4.2 *Different goal levels*[1]

These metaphors are of course not meant as an exact measure for granularity. But, to give you some idea:

- For Cockburn, a sea-level use case is something that can be achieved by the user in one go. A kite-level use case takes the user more than one sitting with the software.
- For us, the Metropolis 1 example hovers at kite level, and the Metropolis 2 example floats at sea level (see Figure 4.1).
- At cloud level, you might model the whole organization Metropolis as a single actor (not differentiating between departments, roles, and software systems).

1. The icons are taken from Wikipedia [Wikipedia Cockburn-Style] and were created by Menner under Creative Commons CC0.

As a moderator, you can bring these metaphors into play whenever the scope seems unclear. Draw the icons from Figure 4.2 on sticky notes and post them on a wall in the meeting room. The icons serve as a constant reminder: "At what level of detail do we need to discuss this?" Once these metaphors are established, you may hear people saying things like, "Let's bring this kite down to sea level!" or "This is not the time to dive to fish level!" It can make sense to add the goal-level metaphor to a domain story, either as an icon or as part of a story's name.[2]

Since COARSE-GRAINED stories cover a lot of ground, storytellers from different departments or even different organizations should participate in a workshop. Up to 15 storytellers is common in such cases. The number of storytellers in a workshop for FINE-GRAINED stories is usually smaller, i.e., around two to seven people. Often the storytellers are from only one (or a few) business departments. But even if we focus just on one department, there might be different roles that need to be represented.

Point in Time (AS-IS vs. TO-BE)

Both Metropolis 1 and Metropolis 2 (see Figure 4.1) describe a process as it currently is. Domain stories can also describe how it will (possibly) look in the future. This **point in time** is another scope factor of a domain story (see Figure 4.3).

Figure 4.3 *Scope factor point in time*

The current situation is often called the *problem space* because the intent of modeling is to improve something that is bad or to solve a problem. Domain stories that model the problem space are called AS-IS stories. Possible improved situations can also be explored with the help of domain stories. Hence, such TO-BE processes describe the *solution space*.

2. In modern Unicode there are even emoji for it. Probably it's good advice to be careful and not use them in filenames, though.

We chose the word pairs as-is/to-be and *problem/solution space* because they are established in the business process modeling community and in the Domain-Driven Design community (see, for example, *Implementing Domain-Driven Design* [Vernon 2013]). You may have come across other word pairs that have the same intention: *descriptive* versus *exploratory* or *informational* versus *aspirational*.

Usually, as-is means the situation at the time of modeling—at a certain point in time. To-be, on the other hand, is a look ahead. There can be different points in the future that may be interesting to model as separate to-be domain stories (see Figure 4.4). Also, you may look at alternative models that explore the same point in time. This is useful for comparing different possible solutions and finding the best one.

Often, the different points in time are chosen because something will be accomplished or changed by then; e.g., a new software system will be operational. The point in time is sometimes added to the name of a domain story.

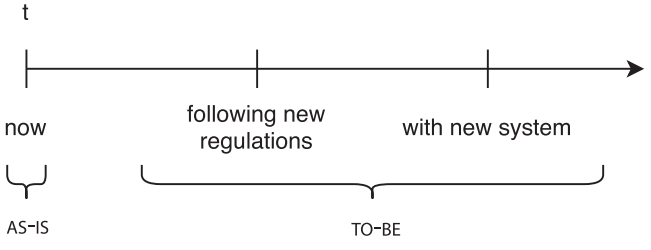

Figure 4.4 *Example points in time of a domain story*

Workshops in which to-be stories are told should involve the future actors—e.g., the (future) users of the software system that is being developed. Other stakeholders such as product owners are also valuable.

Domain Purity (Pure vs. Digitalized)

Looking back at the Metropolis example again, we saw that app developer Anna and movie theater manager Matthew modeled two as-is stories in the first workshop: the coarse-grained (or kite-level) Metropolis 1 and the fine-grained (or sea-level) Metropolis 2 (see Figure 4.1). Did you realize that Anna did not model any software systems that Matthew has used? Surely even a small cinema like the Metropolis must run some kind of IT, even if it is just a spreadsheet. But Anna did not forget to ask Matthew about the software he uses. In fact, Anna made a deliberate choice. When modeling domain stories, you either include or omit (existing or yet to be built)

software. This scope factor is called **domain purity**. We call domain stories without software systems PURE, and those with software systems DIGITALIZED.

PURE domain stories are particularly helpful for building new software systems. They allow you to understand a domain without also internalizing the accidental complexity added by existing software. To lift the veil of software off the domain, the experts should talk about how things *would* be done if all activities were motivated only by the domain (and not by the existing software systems).

For domains that existed before software was ubiquitous, do a thought experiment: Ask for stories that tell how things were done in the pen-and-paper days. Focus on provoking thoughts, not on historical accuracy.

In DIGITALIZED domain stories, the actors can be people or software systems. That means software functionality and its shortcomings are part of the story. In Figure 4.5, for example, the IT system "cinema website" is modeled as an actor.

In many organizations, the domain model is buried under decades of badly modeled software systems. You can use DIGITALIZED AS-IS domain stories to visualize and talk about this mess. Find out which activities are not motivated by the domain but required in order to work with the present software systems. Identify language that was introduced by software developers and is not rooted in the domain. That is where annotations come into play because they help to explain *why* people are doing what they are doing.

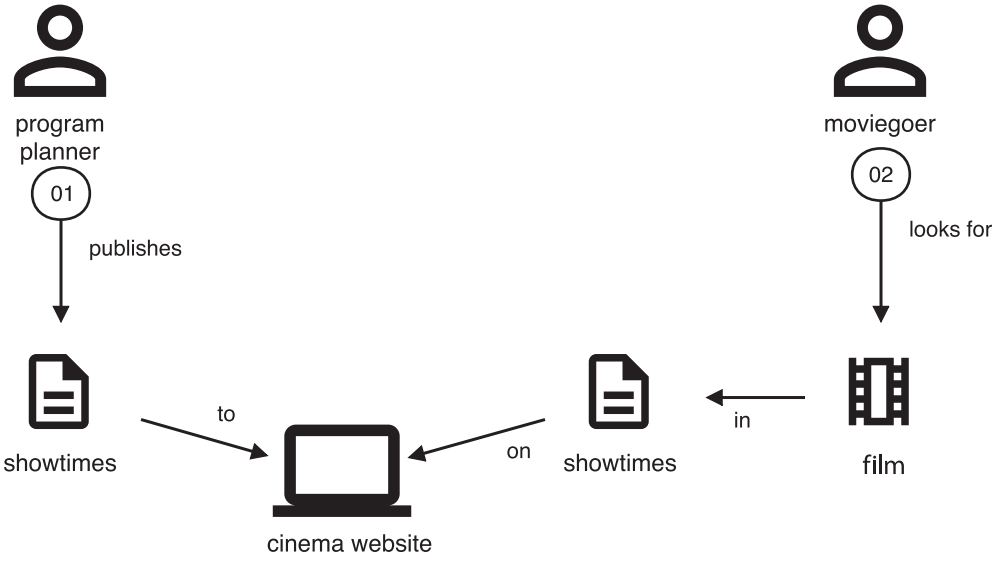

Figure 4.5 *Domain stories with software systems as actors are called* DIGITALIZED

When you want to explore or show how a new software system would change the work, DIGITALIZED TO-BE domain stories will help you. They are often a further development of a PURE AS-IS story—one to which the system is added.

Combining the Scope Factors: A Typical Journey

The factors that determine the scope can be combined in many ways. Mathematically speaking, the scope is a cross product:

$$Scope = Granularity \times DomainPurity \times PointInTime$$

We are going to describe typical scopes of domain stories and how you can travel from one scope to the next:

- COARSE-GRAINED, PURE, AS-IS
- FINE-GRAINED, PURE, AS-IS
- FINE-GRAINED, DIGITALIZED, TO-BE

These are only examples of different types of domain stories, with no strict definitions. Hence, this is not the only possible journey. Feel free to adapt. Also, you do not have to use Domain Storytelling in a sequence from COARSE-GRAINED to FINE-GRAINED. Modeling is usually an iterative process, and you may change the scope as needed. We will show you another example of a journey through different scopes in Part II: "Using and Adapting Domain Storytelling for Different Purposes."

Explore a New Domain (COARSE-GRAINED, PURE, AS-IS)

The first Domain Storytelling workshop in a company, a department, or a development project often takes a COARSE-GRAINED view at the problem space. Stories like that help to get your bearings. Usually, it is helpful to omit software systems to see the naked domain itself.

Since organizations serve a purpose, it is a good idea to look at the primary purpose from the viewpoint of the organization's customers. Hence, COARSE-GRAINED domain stories often illustrate an end-to-end business process. That is why Metropolis 1 (see Figure 1.9) tells a story from buying a ticket for a movie to watching it. If you are uncertain what "end-to-end" really means, you can get the conversation going by explicitly adding a few extra sentences at the beginning and at the end.

Coarse-grained, pure, as-is stories can serve several purposes:

- They are a starting point for exploring the domain (*knowledge crunching*).
- They provide an overview of several fine-grained domain stories (see Chapter 3, "Scenario-Based Modeling").
- They can be analyzed to find boundaries (see Chapter 10) in the domain. You saw a brief example of that in Metropolis 1a (see Figure 2.10), which shows subdomains of the cinema domain. Subdomains can be used as boundaries in teams and boundaries in software—for example, designing a new system in a modular way, splitting a *big ball of mud* [Foote/Yoder 1997] into modules, or splitting a monolith into microservices.

Drill Down into Subdomains (Fine-Grained, Pure, As-Is)

After you've roughly found your way around a domain and established a common understanding of a project's extent, you can drill into the details. Instead of looking at the domain as a whole, you should focus on a selected subdomain. For example, Anna and Matthew drilled down into the ticket sales subdomain when they modeled Metropolis 2 (see Figure 1.10) and Metropolis 3 (see Figure 3.3).

Many companies are organized by subdomains, which means fine-grained as-is stories often take place within one department. However, it can be interesting to add an extra sentence at the beginning and at the end of the story. By doing so, you can take a look at the interface between subdomains or departments. This will help you to understand how people work together across department boundaries.

Fine-grained, pure, as-is stories show *how* people work together *today* (i.e., in the problem space). Such stories are useful in several ways:

- You can find out which business processes could benefit from improved software support. This is often a starting point for conversations about software requirements (see Chapter 11).
- You can distill a technologically untainted domain model from them and implement the model in code (see Chapter 12).
- By comparing as-is with to-be domain stories, you can visualize how work will change (see the next section).

Introduce New Software (Fine-Grained, Digitalized, To-Be)

A common way of using domain stories is to tell how things should change. Stories can be about improved processes, new roles, or new software.