Lab 3: EventStorming

Table of Content

Context and Motivation

Let Alberto Brandolini, the inventor talk first: “EventStorming is a family of workshops, based on collective storytelling with sticky notes on a large modeling surface (usually a paper roll).” (source). He calls out three types/use cases, presented on different levels of detail. We follow the presentation of event storming in “DDD Distilled” in this lab (Vernon 2016).

ChatGPT summarized Brandolini’s original article as this:

> “EventStorming, introduced by Alberto Brandolini in 2013, is a collaborative workshop technique designed to rapidly explore and model complex business domains. By assembling key stakeholders—those with pertinent questions and those with the answers—in an open environment with ample modeling space, participants map out domain events (significant occurrences within the domain) using simple, color-coded sticky notes. This visual and straightforward approach fosters

> engagement, accelerates the modeling process, and aligns seamlessly with Domain-Driven Design principles, particularly benefiting event-sourcing architectures. The method's simplicity and inclusivity make it an effective and enjoyable tool for uncovering domain insights and facilitating shared understanding among diverse team members."

Figure 1 explains the concepts and their relations in the process modeling version of EventStorming along with the color codes for the concepts.

- Domain events constitute the mandatory core (color code: orange)
- Commands (actions) are also part of the mandatory core (color code: blue)
- Actors and query model/information (views and forms) are often stormed as well (dark yellow).
- The figure does now show the data (aggregate/entity) concept, which will be featured in the lab (color code: bright yellow)
- Hot spots are an optional but important and powerful concept; they capture open issues and call to action (color code: red).
- Policy and system are additional optional concepts that will receive less attention in this lab. Their color codes are purple (also called lilac) and pink.

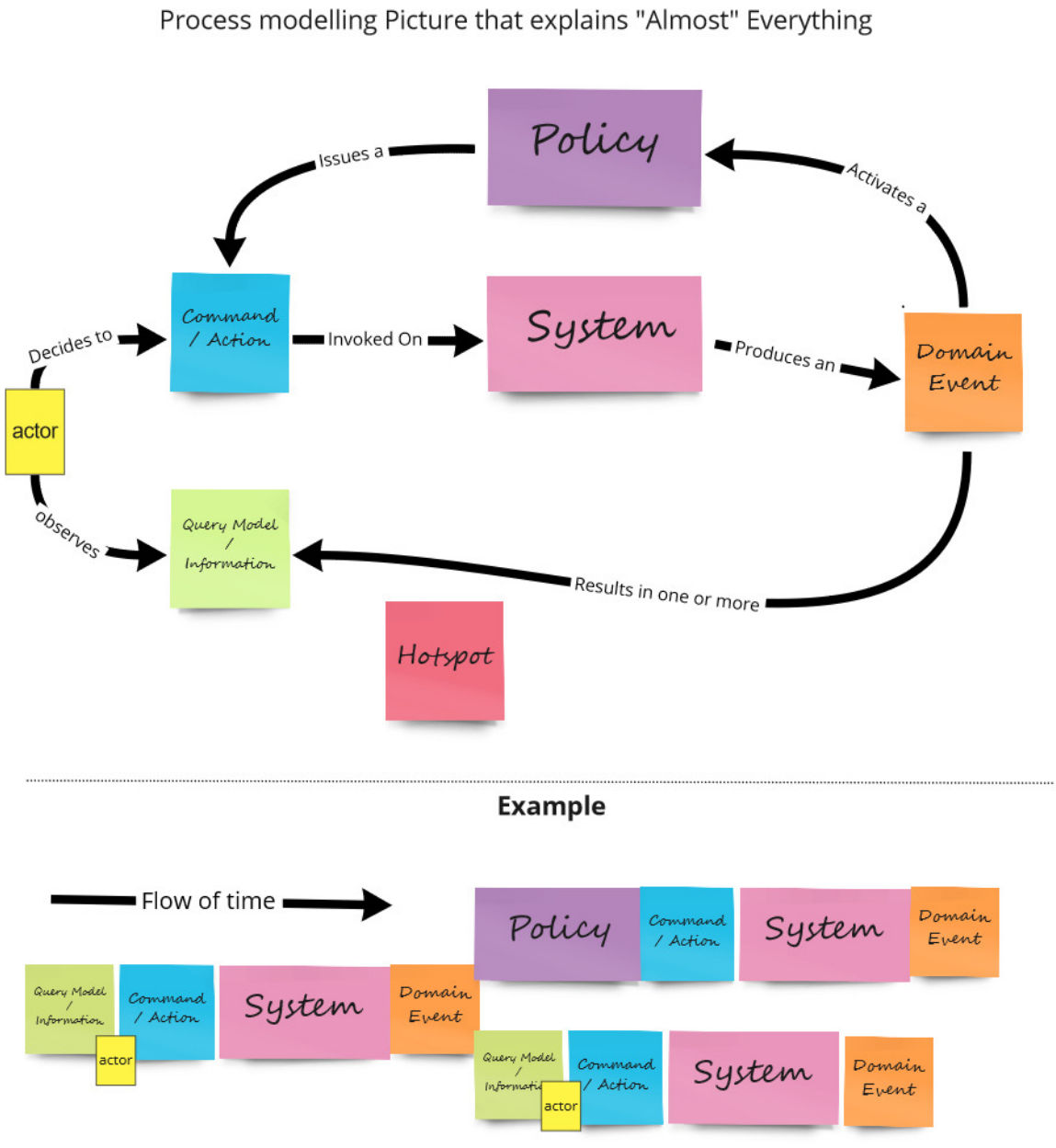

Figure 1: Cheat sheet: concepts and colors in EventStorming (source: ddd-crew on GitHub, CC-BY-SA-4.0 license)

Learning Objectives

Having completed this lab, participants are able to:

- Explain the key concepts and activities in the process modeling use case for event storming.
- Apply event storming to their own cases.
- Review, assess and improve event storming results w.r.t. their use as input for software and other digital design.

Steps Overview

This lab exercise follows the steps proposed in "DDD Distilled", pages 116 to 124 (Vernon 2016):

1. Storm out the business process by creating a series of Domain Events on sticky notes.
2. Create Commands that cause each Domain Event.
3. Associate the Entity/Aggregate on which the Command is executed and that produces the Domain Event outcome.
4. Draw boundaries and lines with arrows to show flow on your modeling surface.
5. Identify the various views that your users will need to carry out their actions, and important roles for various users.

> *Note:* The current setup might be a bit too much for a 90-minute slot, depending on the skills and expectations of participants and desired level of detail/depth. We succeeded with a 75-minute dry run (in MSE CS EVA) and a 90-minute one (in a group meeting). You might want to split the lab into two consecutive sessions. If you need background information, please read parts of Vernon (2013) and Vernon and Jaskula (2022).

Starting Position (Baseline, Initial Position; dt. "Ausgangslage")

We start with parts of the sample solution to Labs 2 and 4 (domain story turned into role-feature-benefit user stories on the epic level):

> 1. As an independent game developer, I want to create game information so that my game can be advertised and sold on the game sharing platform.
> 2. As a business development staff and/or review SME working at Fair Game 3002, I want to review game info drafts so that game developers are treated fairly and our game advertising and sharing policies are enforced.
> 3. As a gamer, I want to discover, buy play and rate games so that I have fun without getting

> addicted or spending more money than I have budgeted.

Note that the lab can also be run through without having done Lab 2; we would start with the product vision from Lab 1 on that case.

Step 1: Storm out the business process (20 mins)

Task: Focus on events to storm out a timeline for things that have to happen to get the product vision to work. Feel free to take one of the roles mentioned in the product vision and/or modeled in Lab 1 (Domain Storytelling), but do not limit yourself strictly to it. On a general level, they map to service provider, service matchmaker/product owner and service consumer.

Hints: This step is described in (Vernon 2016). Some related advice from experience is:

- Focus on pivotal events
- Ask questions like “what’s next?”, “how do/did we get here?”.
- Do not hesitate to move sticky notes around; there is no need to be correct or perfect in this step (and the following ones).
- Leave a lot of space between events because more sticky notes will appear when the next steps are performed!

You might also want to consult the presentations by Paul Rayner (see “More Information” for link to AgileDenver one), with tips such as “focus first on learning and understanding”, “anchor to a concrete business example”, “clarify fuzzy concepts and langauge”, “timebox”.

Questions:

- Are multiple actors involved? Do they all need to see the same information?
- Did you also find hot spots and/or policies (see event storming metamodel)?
- Do business rule variations exist (spoiler alert: we’ll take a closer look in Lab 4 on Story Mapping and Splitting)?

Step 2: Create Commands that cause each Domain Event (20 mins)

Task: Decide where the events from step 1 come from. Give these commands meaningful names. Optionally, identify the user roles (aka actors) performing/executing the commands. This step is also described in (Vernon 2016) (skipping processes, mentioned in the book, is fine.).

> *Hints:* It is ok to vary a bit, for instance command names. If the modeling/storming gets too schematic, where is the point? Note that you may receive different advice here, depending on the experience and position of your instructor (the literature also agrees to disagree).

Questions:

- Which questions did you ask? Did you also find new events?
- Did you find new hot spots and policies in this step?
- Did you identify 1:1 correspondences between events and commands only, or are some events not triggered by a command execution, do some commands emit several events (and so on)?

Step 3: Associate the Entity/Aggregate on which the Command is executed and that produces the Domain Event outcome (20 mins)

Task: This step is also described in (Vernon 2016). Commands operate on data; events report on data changes. This data has to be stored and managed somewhere. Ask which data each command operates on.

> *Hints:* Entity and Aggregate are Domain-Driven Design (DDD) patterns; think data of any kind. You can place the data sticky note as recommended or vary a bit.

Questions:

1. How did you name the data?
2. Is some of the data touched (e.g., created, read, updated, deleted, searched for) by multiple commands?
3. Do you find the entity/aggregate wording hard to comprehend? Where do these terms cone from?

Step 4: Draw boundaries and lines with arrows (15 mins)

Task: Draw boundaries and lines with arrows to show flow on your modeling surface. Identify additional hot spots optionally; for instance, the potential for software systems support (automation) might be worth calling out. Again, this step is described in (Vernon 2016) (“Multiple models …”).

> *Hints:* Lines should receive less attention than the functional decomposition. Do not worry about terms such as subdomain or bounded context at this point. An alternative is to simply put vertical lines where pivotal events are; these lines can be drawn with a marker, but also tape can be used. Your instructor will show you the way! For additional inspiration and interest permitting, please take a look at papers from David Parnas, the coupling criteria in Service Cutter, the Architectural Refactorings in Context Mapper, the service integrators and disintegrators in "Software Architecture: The Hard Parts" (Neal Ford et al. 2021).

Questions:

1. Is it hard to decide? What are the criteria?
2. Do conflicts exist between the criteria?
3. Can the criteria and decision outcome change over time?

Step 5: Identify the various views and forms (aka read models) (15 mins)

Task: Identify the various views and data entry forms that your users will need to carry out their actions, and important roles for various users. This step is also described in (Vernon 2016).

> *Hints:* The term "read model" might be somewhat confusing. Take a system/process user perspective (with external systems, connecting via APIs, also qualifying as users, just like end users). Apply what you learned already, for instance in User-Centered Design (UCD) and database courses.

Questions:

1. Do all commands need views/forms?
2. Can you call out some non-functional requirements that will concern the view/form design?
3. How do events, commands, user roles/actors, read models/views relate to each other?
4. Which optional tasks regarding follow-up do you see?

Tools

- Online whiteboards providing unlimited horizontal space such as miro boards; there is a template in the "miroverse": https://miro.com/miroverse/event-storming/.
- (not tested) draft.io, https://draft.io/example/eventstorming
- (not tested) Figma, https://www.figma.com/templates/event-storming-example/

- (not tested, required login) Qlerify, mentioned in https://medium.com/@tanstorm/event-storming-ia-ddd-turbo-d8f720fccfb0
- Context Mapper for results capturing, tutorial: https://contextmapper.org/docs/event-storming/

Summary and Conclusions

EventStorming is an intense, concept-rich, yet easy to use and relate to, collaborative modeling practice.

Concepts Revisited

- Three usage modes/version of EventStorming: big picture, process modeling, software design.
- Domain Events, pivotal ones in particular; several more key concepts in "grammar" or "meta-model": commands, data, decomposition into subdomains/bounded contexts, hot spots, policies, systems, read models/information (aka views and forms).
- Five storming steps from "DDD Distilled", the book that gave this lab its steps; we discussed that many variations are practiced in industry.

Reflection and Call to Action

Please review and recapitulate what you take away from this lab. Where and when can you apply the taught concepts?

Repetition Questions

1. List the three variants of EventStorming. Which ones was featured in this lab?
2. What are the key concepts in any EventStorming? In the variant shown in this lab?
3. How can the output be used?

Frequently Asked Questions (FAQ)

- *Is EventStorming really new or have we been here before?*
  Answer: The term *event storming* is relatively new (but supported by a rather strong community). That said, following a business event through a scenario or system and asking "what happens next?", "what is needed here?" type of questions is something you find in many consulting methods and personal toolboxes of seasoned architects.

- *Does this work?*
  Answer: Yes. Evidence: see sample solution and experience reports in the public communities.
- *What if I/we do not know enough? For example, if expertise is missing (not all stakeholders present)?*
  Answer: Carefully checking the prerequisites for successful workshop makes sense. On the other hand, there will always be reasons to delay; "just do it" (aka the Nike principle) also makes sense in many cases. You learn as you go and avoid analysis paralysis. Just make sure that everybody who should be involved is at least informed.
- *How to deal with major variations in the modelled process, branches (parallel, alternative), for example error conditions?*
  Answer: Focus on one main modeling thread and analysis/design topic at a time; an event storming does not aim at completeness, consistency or BPMN-type rigor. Analyze the other branches in separate stormings.

More Information

Event storming is featured in many online resources. We found the following ones useful:

- Alberto Brandolini, the creator, writing: "Collaborative Process Modelling with EventStorming" and talking "100,000 Orange Stickies Later"
- Glossary and cheat sheet by DDD Crew on GitHub
- "EventStorming", Paul Rayner, YOW! 2016 presentation video; slides from AgileDenver conference: https://www.slideshare.net/slideshow/event-storming-76390807/76390807 (the presenter also has an eBook on LeanPub)
- "Finding the Turning Points: Pivotal Events in Big Picture EventStorming" by Michael Ploed suggests criteria that make events pivotal (German version: "Identifikation von Pivotal Events im Big Picture EventStorming"). Note that some of the criteria are comparable and rleate dto the 5+2 ones in the ASR Test by ZIOL, taught in AppArch.
- "EventStorming: The Game-Changing Technique You Wish You Knew Sooner!", https://techtrendsetters.org/p/event-storming
- EventStorming introduction and tutorial on Serverless Land (note: big picture mode, different steps)

Books include:

- Chapter 3 in the book "Strategic Monoliths and Microservices" (Vernon and Jaskula 2022), called "Events-First Discovery and Experimentation" covers Big Picture modeling in depth. Figures 3.6 in that chapter introduces the color coding and Figure 3.7 provides a cheat sheet of common modeling patterns.

- The original creator, Alberto Brandolini, has an unfinished but still informative eBook (Brandolini 2021).

For a bigger picture *event modeling* resources can be consulted:

- Event Modeling website. There is a LeanPub book by Martin Dilger called "Understanding Eventsourcing" that has event modeling in its subtitle.
- "Awesome Event Modeling", a collection of resources on GitHub and LinkedIn post with slides "Collaborative Software Design" by Kenny (Baas) Schwegler.
- "Event Modeling: Another Game-Changing Technique You Wish You Knew Sooner!" on techtrendsetters.org.

Finally, more information about related topics is available too:

- Introduction to "Domain Events" by Mathias Verraes.
- Vaughn Vernon's "Design Accelerator" channel on YouTube
- "What do you mean by "Event-Driven"?", Martin Fowler, February 2017.
- Domain Message Flow Modelling is a practice combining event orientation with more traditional, UI- and command-oriented scenario /system walkthroughs.
- Chapter 22 in "Patterns, Principles, and Practices of Domain-Driven Design" by Scott Millett and Nick Tune (Wrox 2015) is an early reference on event sourcing (Millett and Tune 2015); Chapter 24 in that book features CQRS (online update).
- "Implementing Domain-Driven Design" by Vaughn Vernon (Addison Wesley 2013) covers event sourcing and CQRS in Chapter 4 and Appendix A (Vernon 2013).

References

Brandolini, Alberto. 2021. *Introducing EventStorming: An Act of Deliberate Collective Learning*. online: LeanPub. https://leanpub.com/.

Millett, Scott, and Nick Tune. 2015. *Patterns, Principles, and Practices of Domain-Driven Design*. John Wiley & Sons.

Neal Ford, von, Mark Richards, Pramod Sadalage, and Zhamak Dehghani. 2021. *Software Architecture: The Hard Parts: Modern Trade-Off Analyses for Distributed Architectures*. "O'Reilly Media, Inc".

Vernon, Vaughn. 2013. *Implementing Domain-Driven Design*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

———. 2016. *Domain-Driven Design Distilled*. Addison-Wesley. Pearson Education.

Vernon, Vaughn, and Tomasz Jaskula. 2022. *Strategic Monoliths and Microservices: Driving Innovation Using Purposeful Architecture*. Addison-Wesley Signature Series (Vernon). Addison-Wesley Professional.