Lab 6: From Collab. Modeling to OOAD/DDD, OOP

Table of Content

Context and Motivation . . . 1
Learning Objectives . . . 1
Steps Overview . . . 2
Starting Position (Baseline, Initial Position; dt. “Ausgangslage”) . . . 2
Step 1: Review/recap/refine domain storytelling and event storming results . . . 3
Step 2: Review story mapping/splitting and results of value-driven analysis and design . . . 3
Step 3: Create OOA-level class diagrams and sequence diagrams . . . 3
Step 4: Transition from CoMo (as a form of OOA) to OOD and Programming . . . 4
Step 5: Create draft EAM models and integration architectures as DDD context maps . . . 4
Summary and Conclusions . . . 5
Concepts Revisited . . . 5
Reflection and Call to Action . . . 5
Repetition Questions . . . 5
More Information . . . 5
Frequently Asked Questions (FAQ) . . . 6
References . . . 7

Context and Motivation

In the previous labs, we primarily worked in analysis mode, investigating the *problem space* (from a software engineering point of view): What are we building and why, and in which business context? We switch from *Object-Oriented Analysis (OOA)* to *OO Design (OOD)* and *OO Programming (OOP)* now, shaping the *solution space*: How do we build, how does the built software look like and work?

Domain-Driven Design (DDD) will stay with us during this transition. Tactic DDD is one form of OOD: we’ll also learn how to leverage strategic DDD to refine the big picture models from previous labs. The Collaborative Modeling Learning Lab (CoMo LL) positions strategic DDD as a pragmatic, agile way of doing/expressing Enterprise Architecture Management (EAM). This is inline with the way DPR (Zimmermann and Stocker 2024) presents Strategic Domain-Driven Design.

Learning Objectives

Having completed this lab, participants are able to:

- Review the output of previous CoMo labs w.r.t. use as input to design (complete and good/mature enough?).
- Transfer results from Labs 1 to 5 into OOAD/UML notation and supporting practices and tools.
- Leverage the created models for design, to justify design decisions, to ensure traceability.

Steps Overview

This lab exercise has the following steps:

1. Review/recap domain storytelling and event storming results (Labs 1 and 2).
2. Review story mapping/splitting and VDAD stakeholder mapping/value impact mapping results (Labs 3 and 4).
3. Create OOA-level class diagrams and sequence diagrams from output of Steps 1 and 2 of this lab: Evolve domain storytelling and event storming results into OOA domain model (UML class diagram).
4. Transition from OOA to OOD and OOP (using the same notation, UML, but on a different level of refinement/detail and with different modeling purpose); proceed from OOA to OOD by selecting strategic and tactic DDD patterns for the solution/software design.
5. Create draft Enterprise Architecture Management (EAM) models and integration architectures from the output of Steps 3 and 4 of this lab. Context maps from strategic domain-driven design will serve that purpose.

Starting Position (Baseline, Initial Position; dt. “Ausgangslage”)

We have already gone through five CoMo Learning Labs up to here:

- One or more domain storytellings have been performed. See “Requirements: From Domain Story to User Story” for background information.
- An EventStorming wokshop has happened.
- User stories have been mapped and split.
- Stakeholders and their values have been identified with the help of Value-Driven Analysis and Design (VDAD), a method and practice collection for ethical software engineering from R&S at OST.
- The values have been prioritized and value requirements been derived from them, again applying concepts from VDAD.

Note that it is possible to perform Lab 6 without having finished all previous ones. The sample solutions can serve as starting points in that case.

Step 1: Review/recap/refine domain storytelling and event storming results

Task: Return to your solutions to CoMo Labs 2 and 3. Review the provided sample solutions too.

> *Hints:* "Domain Storytelling is a useful and versatile tool in a DDD practitioner's tool box" and "Domain-Driven Design (DDD) is an approach that deals with the development of business software. DDD does not describe a single modeling method, but it emphasizes collaborative modeling by domain experts and development teams as the essential part of software development" (source: https://domainstorytelling.org/domain-driven-design).

Questions:

- What do the lab results model?
- How can these results be used for software design and development?
- What would you do differently next time you use domain storytelling and event storming?

Step 2: Review story mapping/splitting and results of value-driven analysis and design

Task: Return to *your* solution to CoMo Labs 4 and 5. Review their sample solutions too.

Questions:

- What do the lab results (i.e., the created models and maps) express?
- How can they be used for business-as-usual design and development?
- What would you do differently next time you use story mapping/splitting, stakeholder mapping, value impact mapping?

Step 3: Create OOA-level class diagrams and sequence diagrams

Task: Represent the results from Labs 2 to 5 in selected diagrams from the Unified Modeling Language (UML).

> *Hints:* Use cases, class diagrams (and sequence diagrams) seem to be particularly suited (and taught elsewhere). Tools can support this step, for instance PlantUML, Context Mapper and MDSL Tools.

Questions:

1. Which classes did you model? Do they feature both data and logic (methods, functions)?
2. Which relationship did you model? Why those? Do others exist?

3. How would you assess the quality of your model? How can it be improved?

Step 4: Transition from CoMo (as a form of OOA) to OOD and Programming

Task: a) Map CoMo concepts to OO/UML and, optionally, DDD: b) Refine the UML from the previous steps, faithful to the original vision of OOAD from the 1990s.[1] Prototype the design that begins to shape in the analysis material produced so far.

> *Hints:* Context Mapper and JHipster are one way of doing so; you might want to try out special-purpose AI too, or a low code/no code tool, for instance one taught in an OST lecture.

Questions and retrospective tasks:

- Reflect on the experience. What did you like, learn, lack, long for (4L)?
- Comment on productivity gain, with and without CoMo practices, with and without RAD tool support.
- Which tools would you like to use?

Step 5: Create draft EAM models and integration architectures as DDD context maps

Task: Use strategic DDD patterns to show the system under construction in the example scenario with its subsystems and external context relationships (provided and consumed interfaces). To do so, draw a Context Map with Bounded Contexts and suited strategic DDD relationship types (e.g., Customer-Supplier, Open Host Service-Conformist or Open Host Service-Anti Corruption Layer, Shared Kernel).

> *Hints:* The Context Mapping project in the ddd-crew organization on GitHub (https://github.com/ddd-crew) provides background information. You might also want to consult the AppArch exercise on strategic DDD for inspiration, which works with two articles from Statoil (Landre, Wesenberg, and Rønneberg 2006), (Wesenberg, Landre, and Rønneberg 2006). The Context Mapper website and its tutorials might be worth visiting too, for instance "Model EventStorming Results in Context Mapper".

Questions:

- What is the motivation for the Bounded Context pattern? How is it defined?
- How do the various context relationship types differ?

[1] See articles and books by Booch (Grady Booch 1982), Larman (Larman 2004), and other OO pioneers. "Object design and analysis methods before and after UML" is informative as well.

- Which features would you like to see in Context Mapper?

Summary and Conclusions

This rather intense lab took us from analysis to design.

Concepts Revisited

- Use cases, UML class diagram to capture CoMo results and transition to OOD and OOP
- Tactic DDD patterns in OOD and OOP
- Strategic DDD patterns as a lightweight form of Enterprise Architecture Management (EAM): bounded context, context map, relationship types

Reflection and Call to Action

Please review and recapitulate what you take away from this lab:

- Reflect on the experience. What did you like, learn, loath, long for (4L)?
- Comment on productivity gain, with and without CoMo practices, with and without RAD tool support.
- Where and when can you apply the taught concepts? Which tools would you like to use?

Repetition Questions

1. What is the difference between analysis and design? How do these concepts relate to each other?
2. What are two synonyms for analysis and design spaces?
3. Does it make sense to use the same notation for analysis and design? Provide one pro argument and one contra argument.

More Information

- From the website supporting the book (Hofer and Schwentner 2021): “Chapter 11 of the Domain Storytelling book describes a recipe for breaking down domain stories and building a backlog of requirements with User Story Mapping.” (source: https://domainstorytelling.org/domain-driven-design)

- "Chapter 09 of the Domain Storytelling book shows how Domain Storytelling can help you to build up domain knowledge which is a prerequisite for establishing ubiquitous languages." and "In chapter 10 of the Domain Storytelling book, we elaborate an important step in strategic design: finding the boundaries between subdomains." and "Chapter 12 of the Domain Storytelling book shows how to turn a domain story into an object-oriented or functional domain model." (source: https://domainstorytelling.org/domain-driven-design)
- For OOAD and OOP, its Wikipedia page is a good start; you can find many other pointers elsewhere in this lab and in other CoMo LL material.
- DDD patterns and definitions are explained in "DDD Reference" by Eric Evans, https://www.domainlanguage.com/ddd/reference/ (free PDF). Two examples of strategic DDD in action are reported in (Landre, Wesenberg, and Rønneberg 2006) and (Wesenberg, Landre, and Rønneberg 2006).

Frequently Asked Questions (FAQ)

- *Do I always need all of this? On the presented level of detail?*
  Answer: No. Methods and practices are always adopted for the current context and culture. They are means to an end not an end. See DPR guidelines for methods engineering and adoption, also featured in a blog post. Examples: "if in doubt leave it out", "do not model without a purpose".
- *Is a general software engineering/software architecture practice collection available online?
  Answer: Yes, DPR, which also comes as an eBook (Zimmermann and Stocker 2024).
- *What is this DPR thing that keeps on getting mentioned?* Answer: Originally created as part of an AppArch script and then evolved on a research project, Design Practice Reference/Repository (DPR) features parts of Olaf Zimmermann's analysis design toolbox grown since 1995. Mirko Stocker contributed part of his toolbox as well.
- *What is the essence of Object-Orientation (OO)?*
  Answer: Three tenets come to mind: 1) Data and logic are kept together, in analysis and design. 2) Information hiding via interfaces. 3) Concepts from the real world/domain as first class citizens in code. Note the absence of inheritance or polymorphism in this list!
- *What are OOA, OOD and OOP? How did OOA, OOD, OOP evolve? What are its predecessors?* Answer: OOA stands for Objected-Oriented Analysis. OOD is OO Design, OOP Programming. Structured Analysis (SA) and Structured Design (SD) are forerunners of OO-anything. Grady Booch recently reviewed the OO roots in a TSE article called "Object-Oriented Development, Revisited" (G. Booch 2025). "Object design and analysis methods before and after UML", a blog post by Rebecca Wirfs-Brock and the background information page of the DPR project provide additional background information.
- *Do I need to use an object-oriented language to use the CoMo results?*
  Answer: No, the decision which language to code in should be driven by other criteria. DDD pat-

terns such as Aggregate and Entity is easier to implement if the chosen language supports objects and classes. Even languages such as Python and Go now support such or similar concepts. OOAD and DDD can be applied and then a different programming paradigm (functional, imperative) switched to. Chapter 12 “Modeling in Code” of the “Domain Storytelling” book covers this topic in depth (Hofer and Schwentner 2021).

References

Booch, G. 2025. “Object-Oriented Development, Revisited.” *IEEE Trans. Softw. Eng.* 51 (3): 725–27. https://doi.org/10.1109/TSE.2025.3536328.

Booch, Grady. 1982. “Object-Oriented Design.” *Ada Lett.* I (3): 64–76. https://doi.org/10.1145/989791.989795.

Hofer, Stefan., and Henning Schwentner. 2021. *Domain Storytelling: A Collaborative, Visual, and Agile Way to Build Domain-Driven Software*. Addison-Wesley Signature Series (Vernon). Pearson Education. https://books.google.ch/books?id=vNnPEAAAQBAJ.

Landre, Einar, Harald Wesenberg, and Harald Rønneberg. 2006. “Architectural Improvement by Use of Strategic Level Domain-Driven Design.” In *Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications*, 809–14. OOPSLA ’06. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1176617.1176728.

Larman, Craig. 2004. *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition)*. USA: Prentice Hall PTR.

Wesenberg, Harald, Einar Landre, and Harald Rønneberg. 2006. “Using Domain-Driven Design to Evaluate Commercial Off-the-Shelf Software.” In *Companion to the 21st ACM SIGPLAN Symposium on Object-Oriented Programming Systems, Languages, and Applications*, 824–29. OOPSLA ’06. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/1176617.1176730.

Zimmermann, Olaf, and Mirko Stocker. 2024. *Design Practice Reference - Guides and Templates to Craft Quality Software in Style*. online: LeanPub. https://leanpub.com/dpr.