OST
Eastern Switzerland
University of Applied Sciences

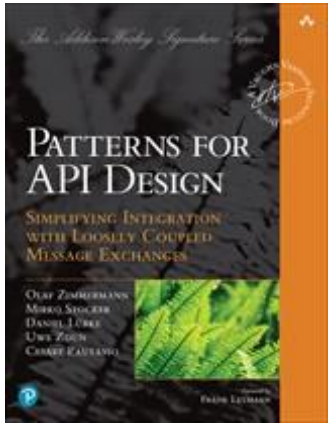
Introduction to DPR, MADR, LASR (What?)

Digital Solution Architecture, Hannover, April 23, 2025

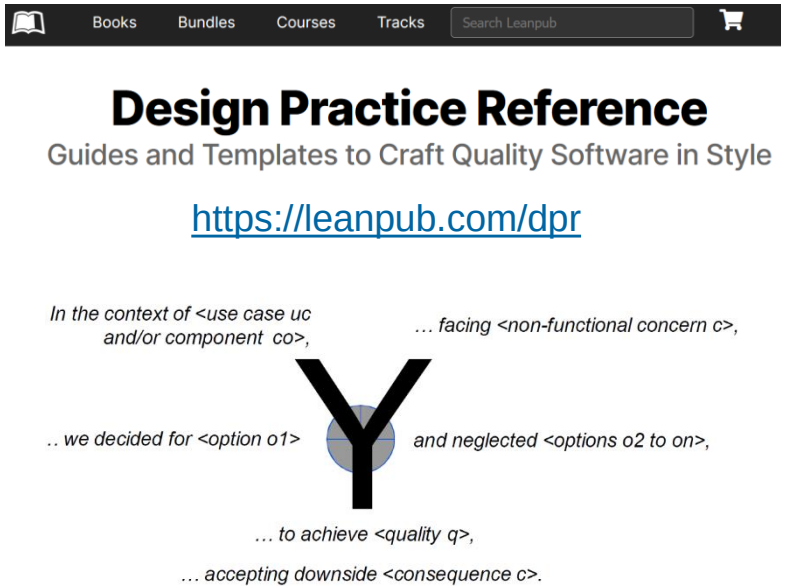
Dr. Olaf Zimmermann (ZIO)
https://ozimmer.ch/

Who is ZIO?

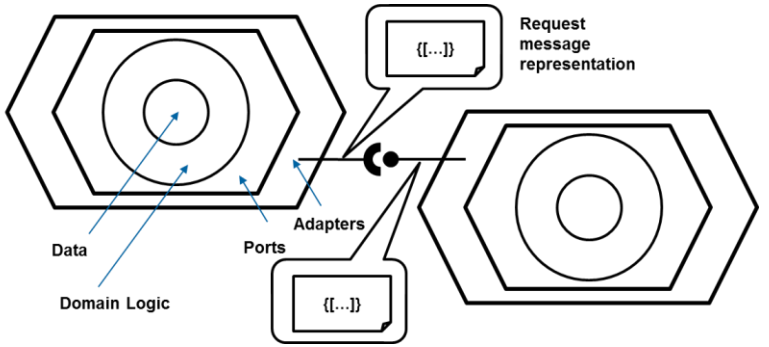
- **Software Architect**
  - Formerly with IBM and ABB
  - The Open Group certified
- **Lecturer, Author, Blogger**
  - Agile architecting
  - All things integration
  - (Micro-)services
- **Open Sourcer**
  - Context Mapper (DDD Tool)
  - Interface Refactoring Catalog
  - API Patterns and MDSL
  - Y-Statements as a compact form of ADRs, Markdown ADRs (MADR)



OST
Ostschweizer
Fachhochschule



https://medium.com/olzzio



Structure and Take-aways of this Talk

1. **Architect deeper: DPR**
2. **Document decisions diligently and selectively (mad-der?): MADR**
3. **Review LASR-sharp**

Pick and choose, make methods and practices your own.

(Almost) AI-free talks exist in 2025.

And I can do without patterns 😉

OST
Eastern Switzerland
University of Applied Sciences

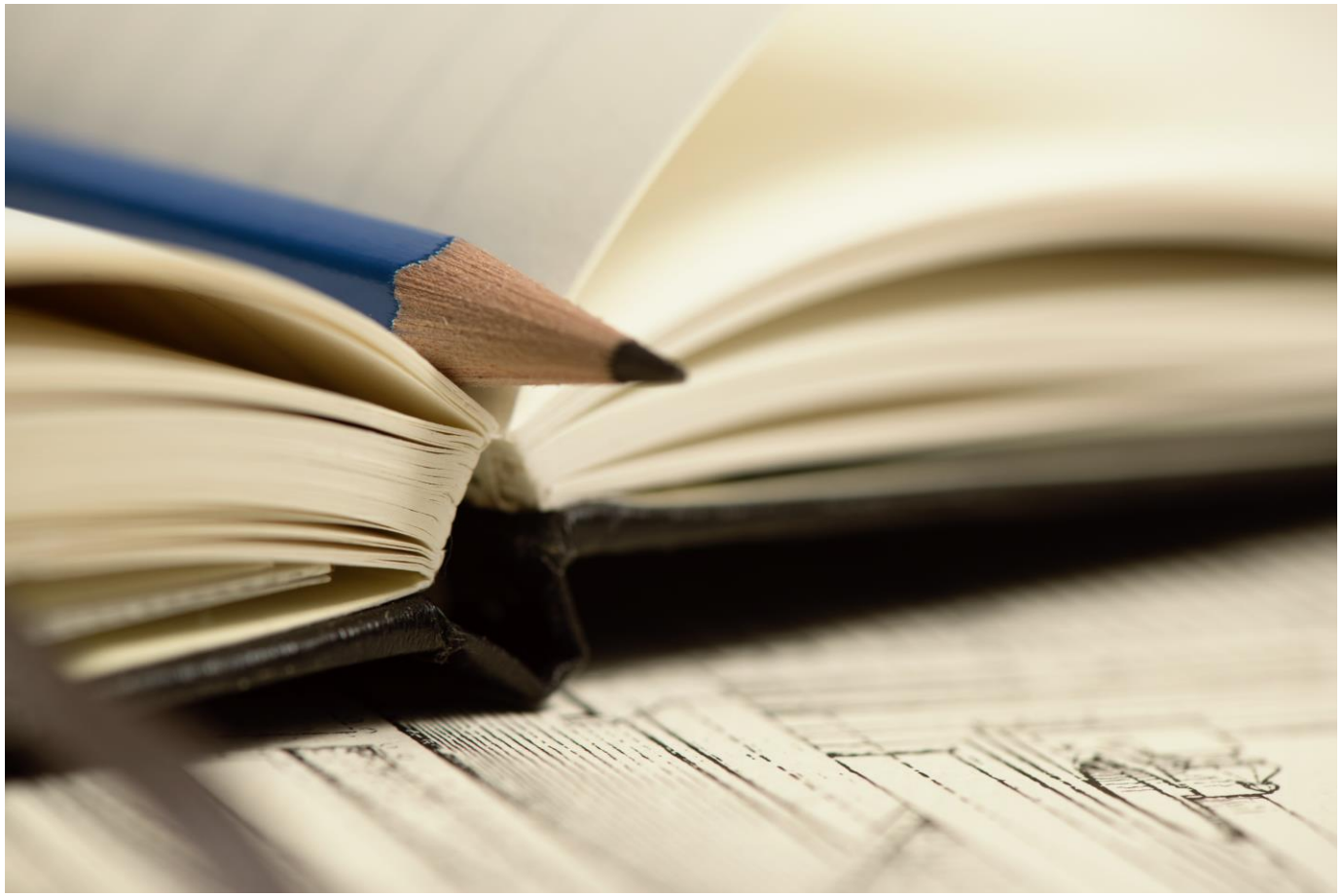
Introduction to DPR, MADR, LASR
Part A: Design Practice Repository and Reference (DPR)

Dr. Olaf Zimmermann

olaf.zimmermann@ost.ch (until 8/25)

Motivating Road Story from Consulting Days

- Once upon a time…



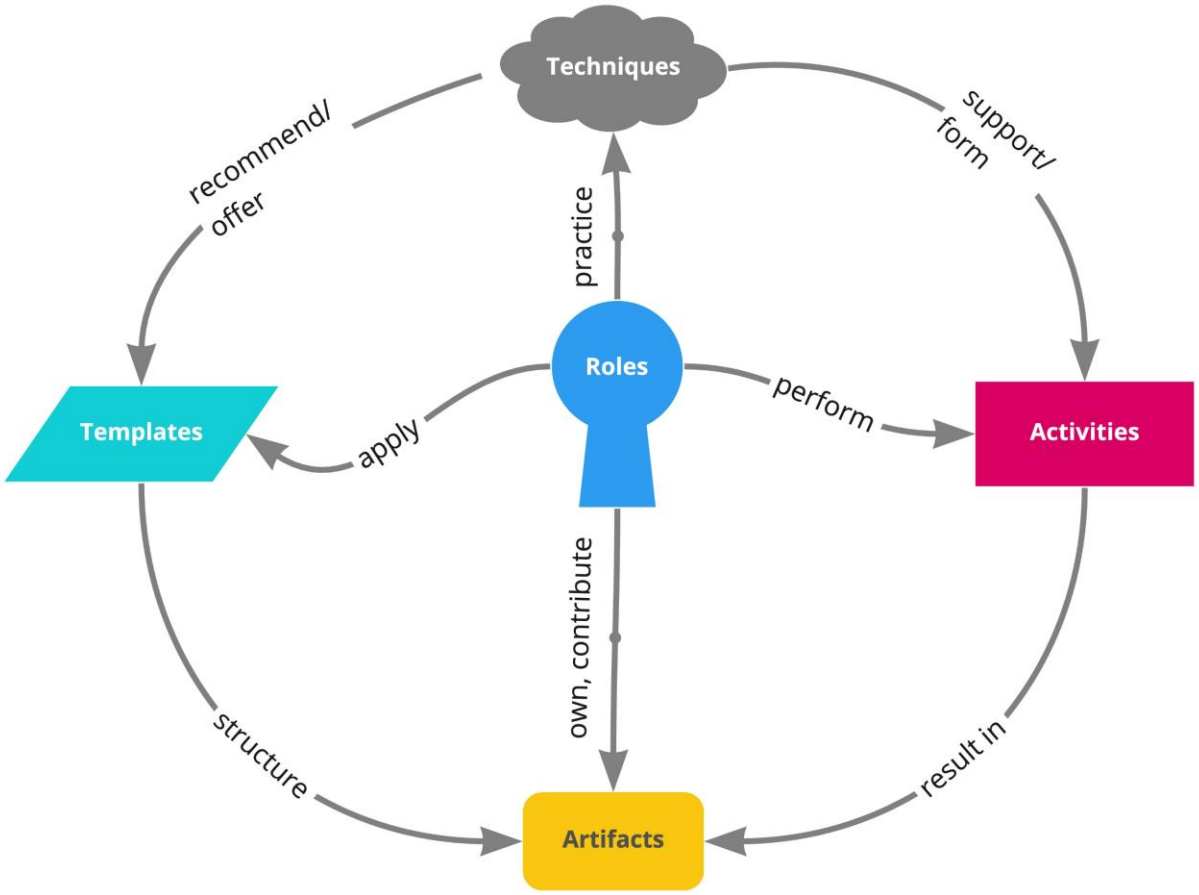
Design Practice Repository/Reference (DPR)

- Use cases and positions
  - Learn from peers and “the elder”, establish toolbox and vocabulary
- FAIR consulting: share knowledge, give something back
  - Be reproducible, understand each other, do not waste time on method wars
- Method can be *lit*!
  - Likable/light, instrumental, trustworthy

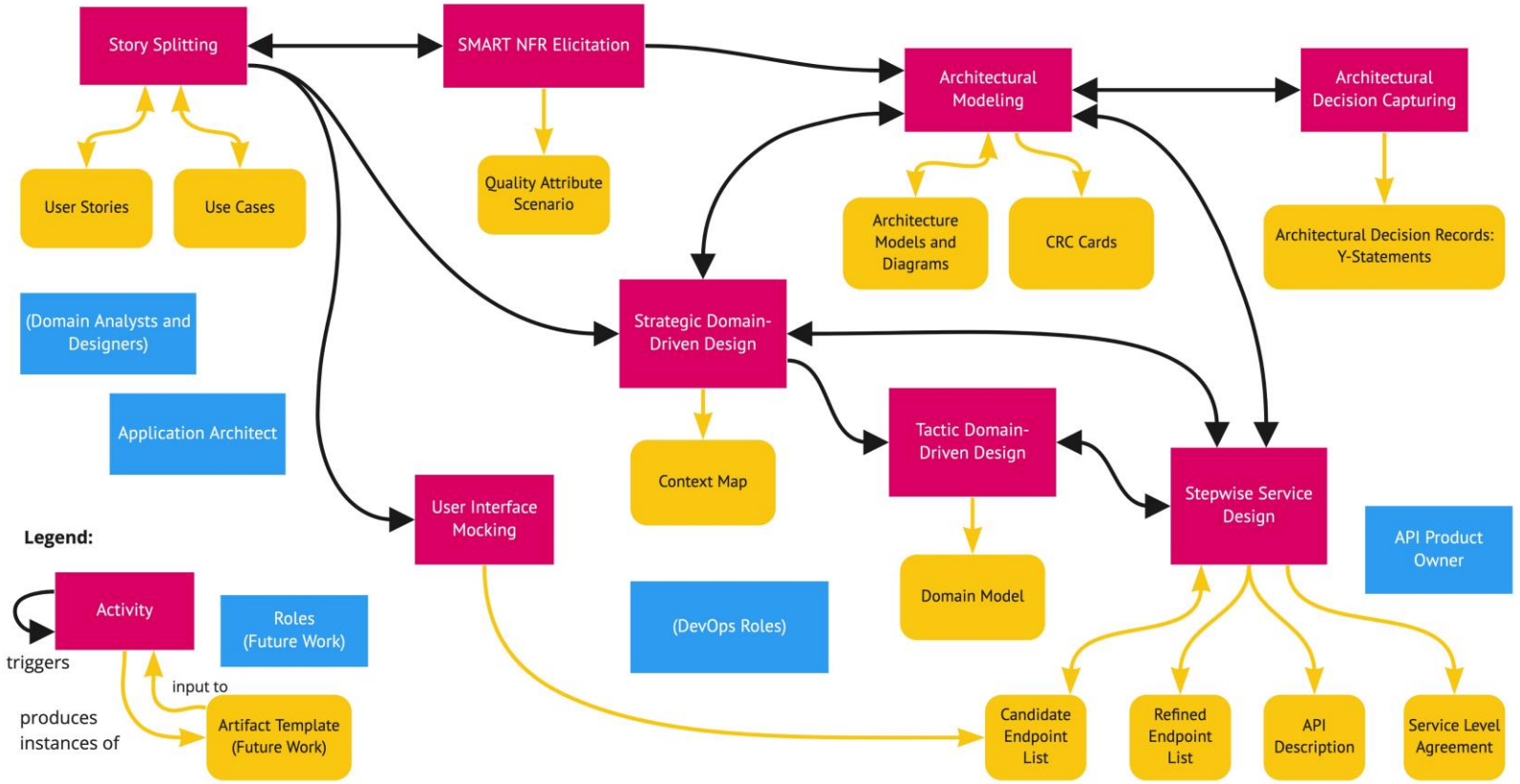
Check out the GitPages website and the blog post(s) on DPR

DPR Concepts

- Metamodel inspired OMG SPEM and others
  - Roles, activities, artifacts
  - Techniques
- Templates inspired by Agile Alliance and others:
  - User story (in context)
  - Concepts
  - Example
  - Origins
  - Hints and pointers



DPR Activities and Artifact Templates

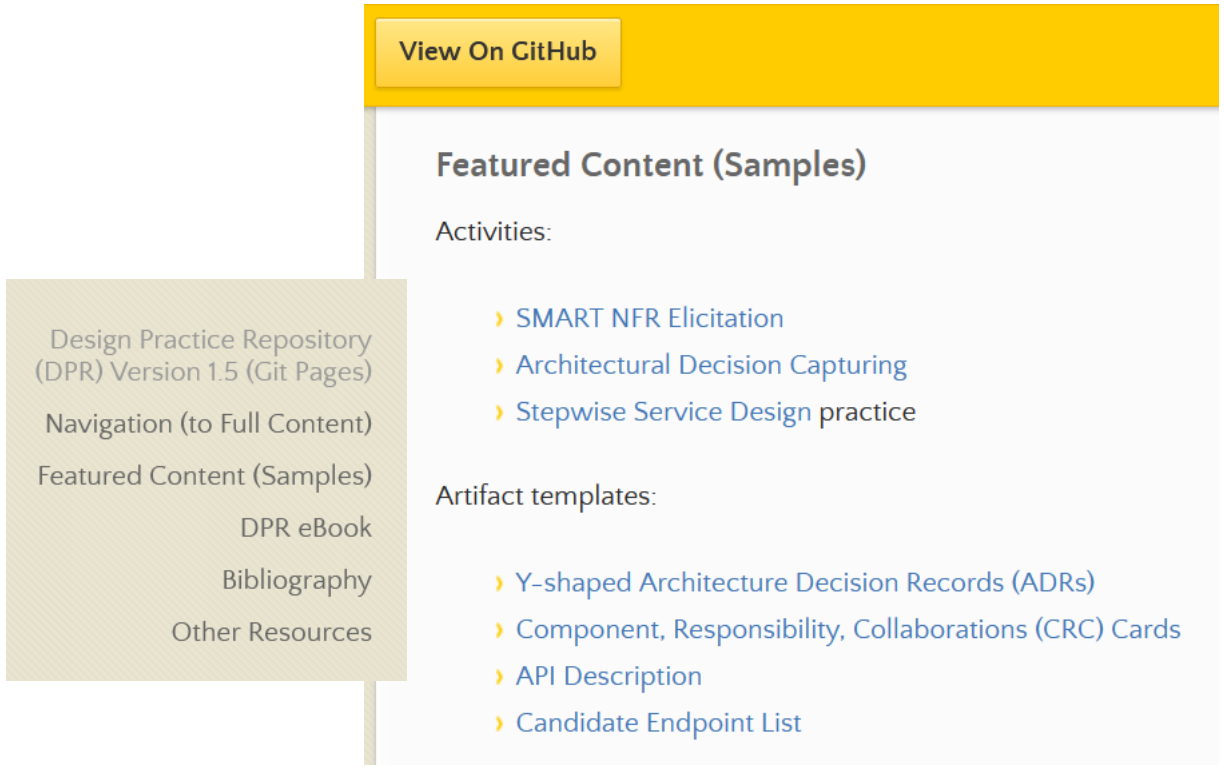


Design Practice Repository (DPR) Version 1.5 (Git Pages)

Summaries of artifacts, templates, practices, and techniques for agile architecting (DPR-mm) and service design (SDPR-nn).

DPR Online

- SMART NFR Elicitation
- User Interface Mocking
- Y-Statements and other ADR templates



https://socadk.github.io/design-practice-repository/

Design Practice Repository (DPR) Version 1.5 (Git Pages)

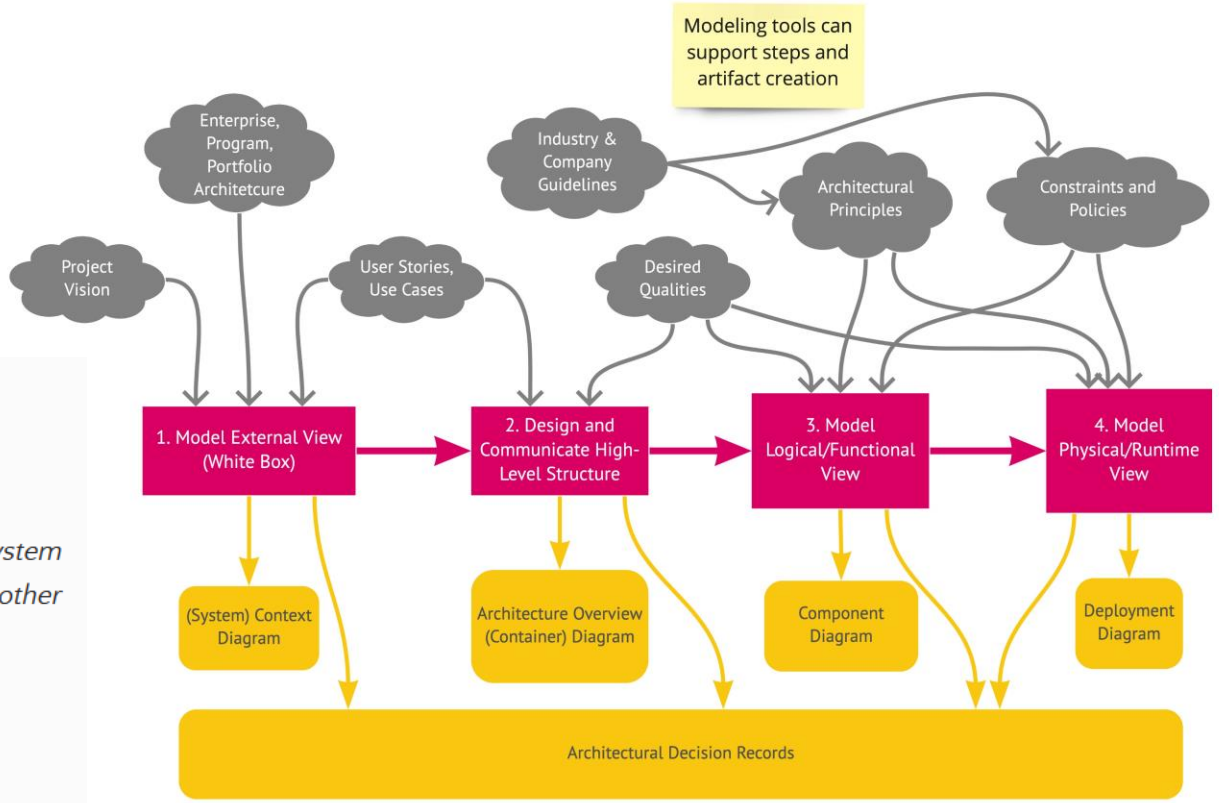
Summaries of artifacts, templates, practices, and techniques for agile architecting (DPR-mm) and service design (SDPR-nn).

Sample Content (1)

Architecture Modeling

Goal and Purpose (When to Use and When not to Use)

> *As a software engineer performing architecture design work,*
>
> *I want to capture my current understanding of the static and dynamic structure of the system under construction (in terms of its components and connectors), share it with peers and other stakeholders, and continuously evolve it*
>
> *so that I can plan ahead (design and implementation work), manage risk, and trace the design back to architecturally significant requirements.*



https://socadk.github.io/design-practice-repository/activities/DPR-ArchitectureModeling.html

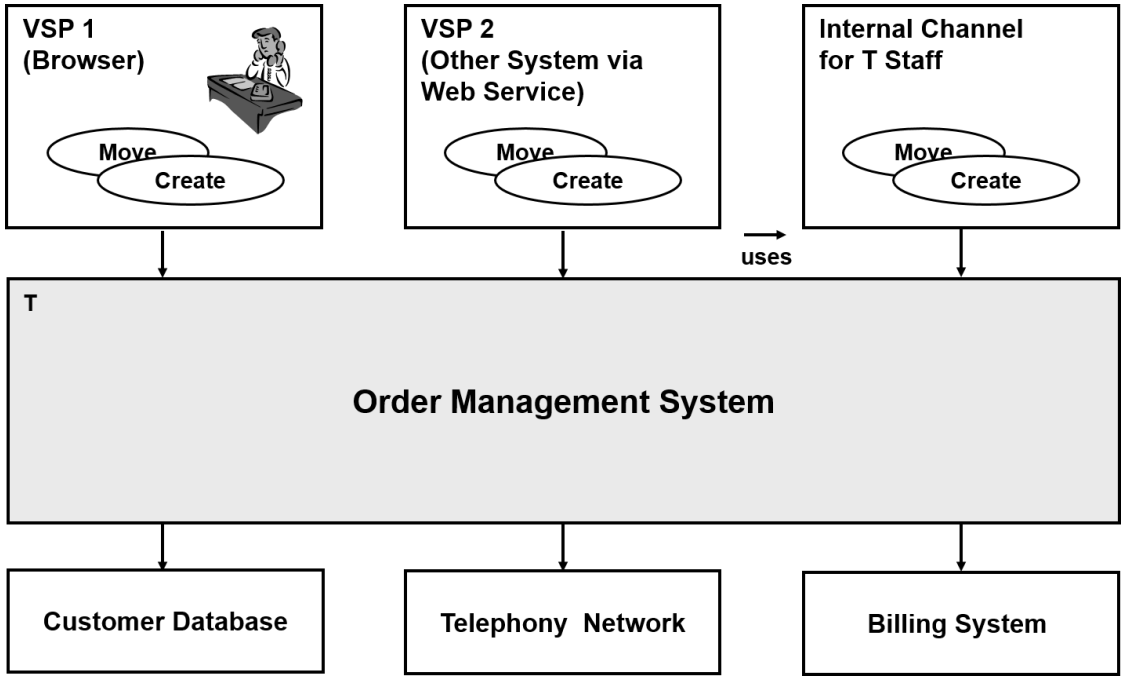
Design Practice Repository (DPR) Version 1.5 (Git Pages)

Summaries of artifacts, templates, practices, and techniques for agile architecting (DPR-mm) and service design (SDPR-nn).

Sample Content (2)

Hints and Pitfalls to Avoid (Common Pitfalls)

- Organize the diagram for readability; for instance, place and order consumers and providers by their type (human user, external system) and/or importance and/or time of activity. Label all arrows, and explain their meaning in a legend (for example, runtime API call or compile-time dependency?).
- Do not confuse black box and white box and views. Here, no internals should be shown.
- Do not stop at the outside view, but zoom in (with a container diagram).
- Do not forget to update this diagram as a design evolves; when a new API is consumed, this has to be shown as an outbound external interface in any context diagrams drawn earlier.
- Consider Strategic DDD to carve out contexts.



https://socadk.github.io/design-practice-repository/artifact-templates/DPR-ContextDiagram.html

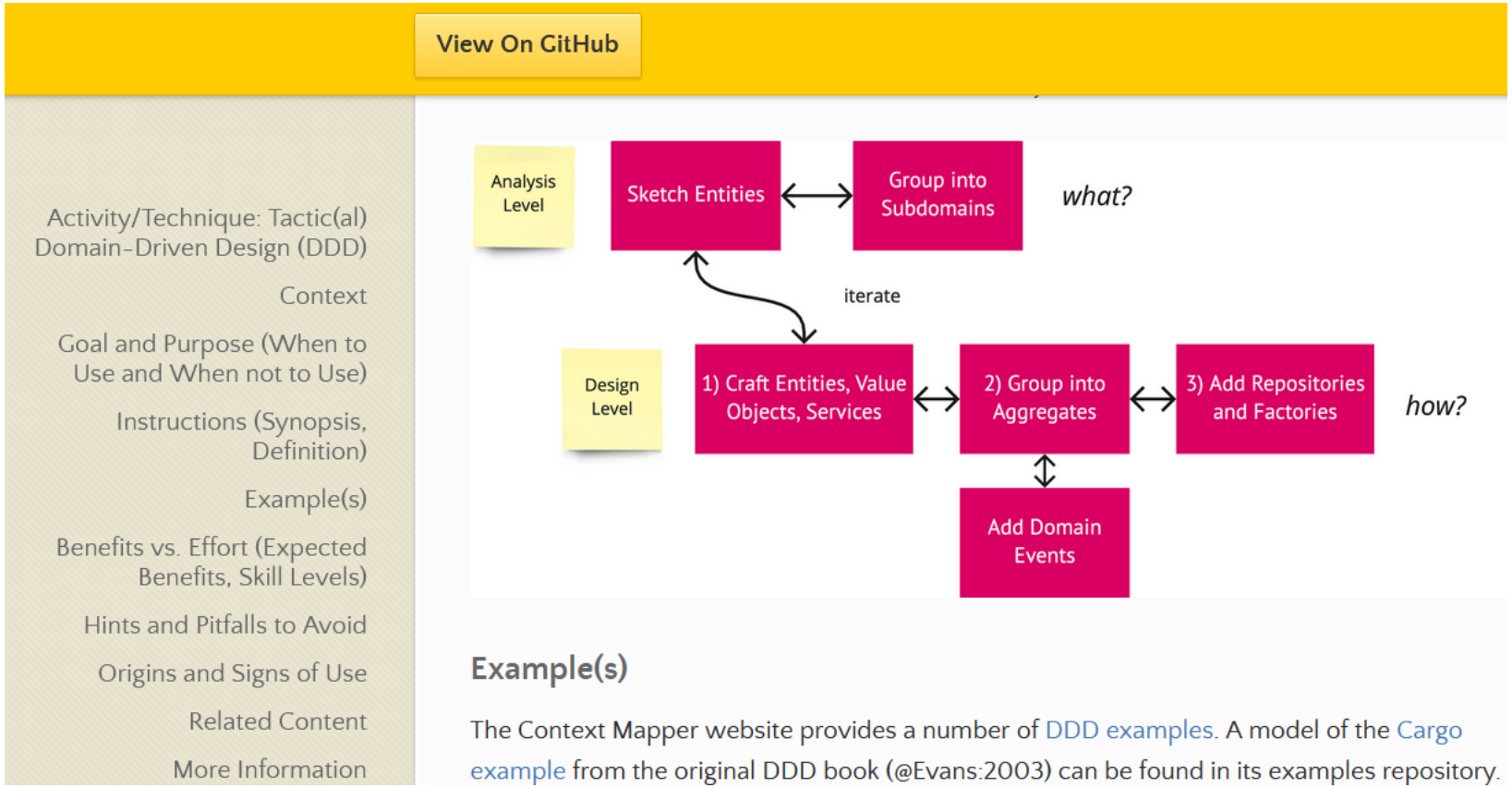
Design Practice Repository (DPR) Version 1.5 (Git Pages)

Summaries of artifacts, templates, practices, and techniques for agile architecting (DPR-mm) and service design (SDPR-nn).

Sample Content (3)

Tactic DDD:

- Aggregate
- Entity
- Value Object
- Domain Events
- Analysis and design steps



https://socadk.github.io/design-practice-repository/activities/DPR-TacticDDD.html

https://leanpub.com/dpr/read_sample

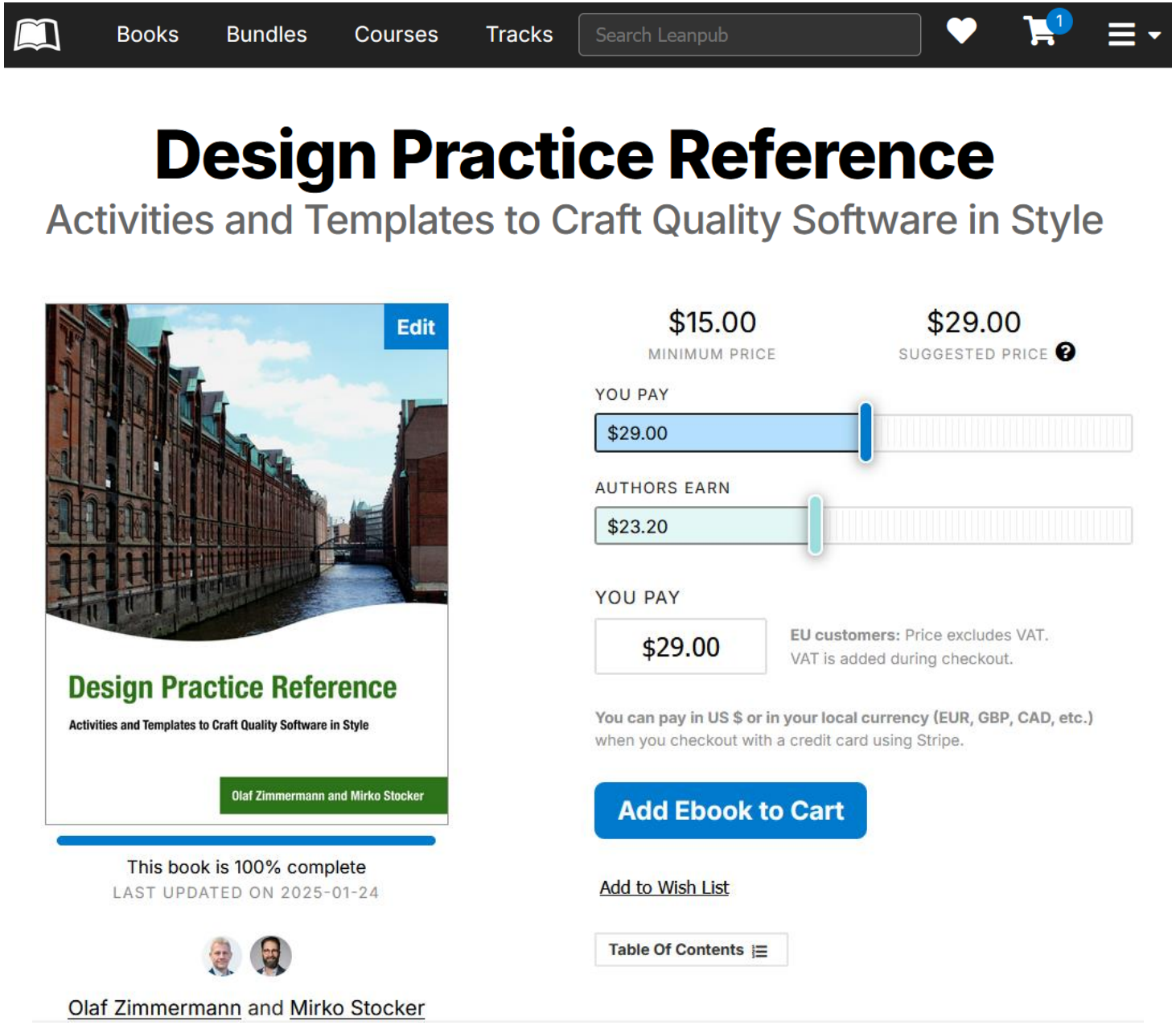
DPR eBook: Method Creation/Usage Principles

- **Context matters**
  - What works well for one role in a particular client, team, and project environment might be a major source of headache and trouble elsewhere.
- **If in doubt, leave it out**
  - Do not create a “big ball of method mud”.
  - Create a template and other method elements with a target audience in mind.
- **Value purpose/usefulness over template compliance**
  - Do not follow templates and process advice by the book but adopt them to your needs. No blind obedience or cargo cults!
- **Do not reinvent the wheel but look for existing templates**
  - Tweak them as needed. Acknowledge and reference your input properly.

DPR eBook

- Motivation
- Method engineering and adoption principles
- Examples when and how to apply
- Content of GitHub repository

https://leanpub.com/dpr



OST
Eastern Switzerland
University of Applied Sciences

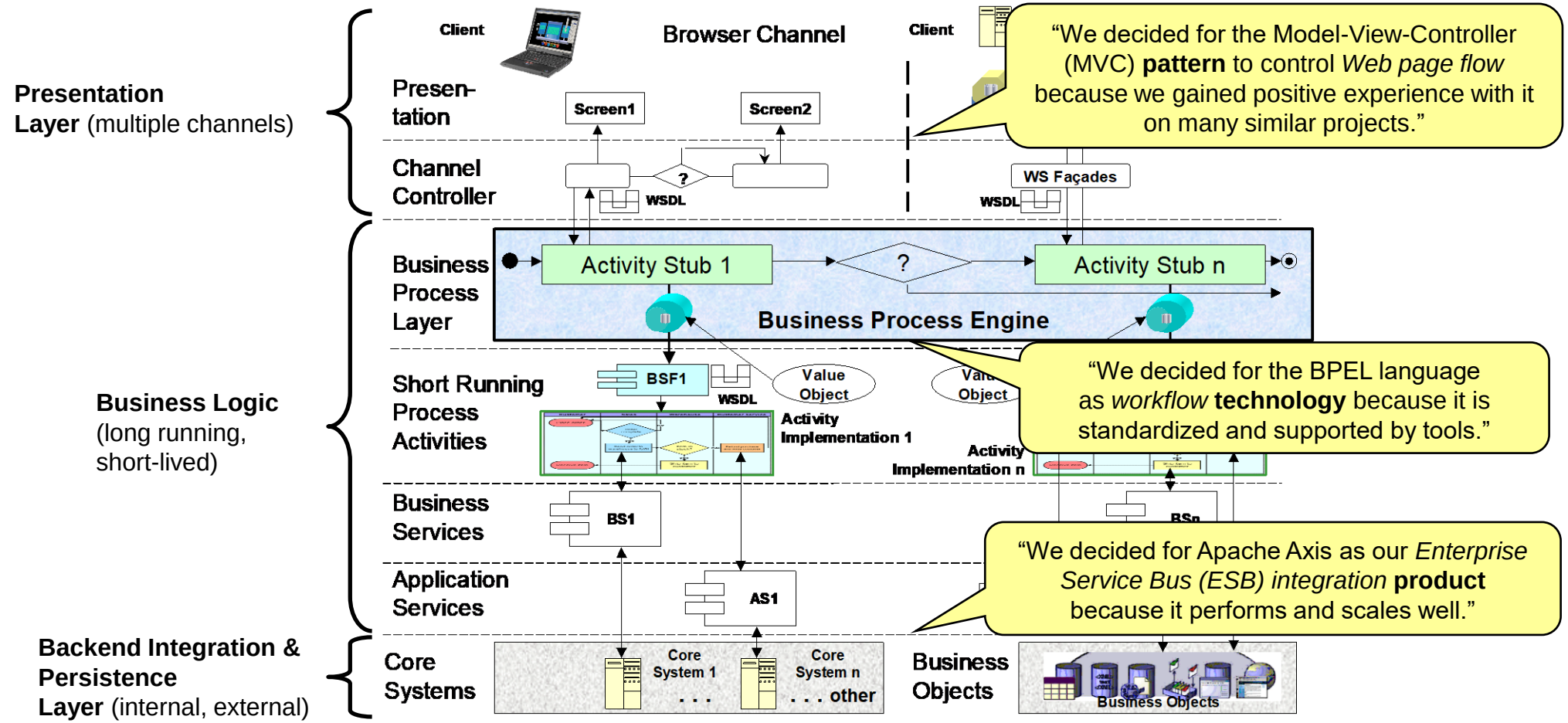
Introduction to DPR, MADR, LASR
Part D: Markdown ADRs (MADR)

Dr. Olaf Zimmermann

olaf.zimmermann@ost.ch (until 8/25)

Reference: Zimmermann et al, „Service-Oriented Architecture (SOA) and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned“, OOPSLA 2005 conference companion, ACM Press, 2005

ADs in a Typical Enterprise Application (2004)



Two ADR Templates: Nygard and WH(Y)

- Cognitect Blog 2011: “ADR”
  - Context
  - Outcome
  - Status
  - Consequences
- ABB 2012: “WH(Y)”
  - Two-part context
  - Chosen and neglected options
  - Good and bad consequences
  - Metadata left out in template

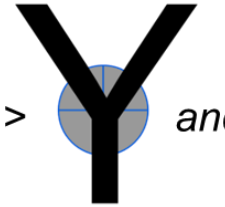
DOCUMENTING ARCHITECTURE DECISIONS

Michael Nygard - November 15, 2011 AGILITY ARCHITECTURE

In the context of <use case uc and/or component co>,

… facing <non-functional concern nfc>,

… we decided for <option o1>



and neglected <options o2 to on>,

… to achieve <positive consequence/quality q>,

… accepting that <negative consequence c>.

Reference: Sustainable Architectural Design Decisions, IEEE Software, Vol. 30, Issue 6, 2013 and SEI SATURN 2012

Markdown ADRs (MADR)

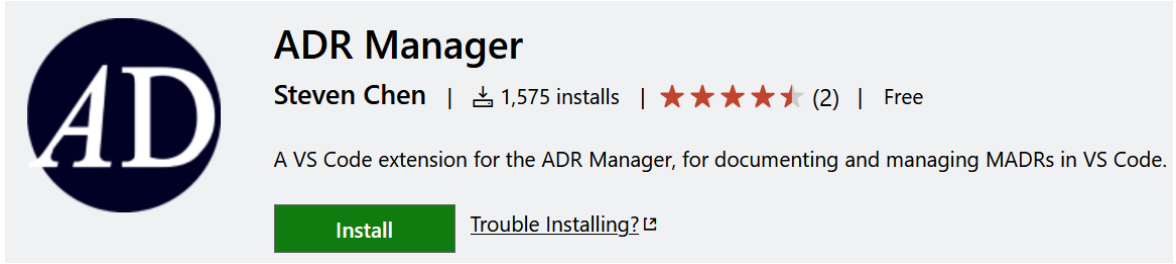
https://adr.github.io/madr/
https://github.com/adr/madr

```
# {short title, representative of solved problem and found solution}

## Context and Problem Statement

{Describe the context and problem statement, e.g., in free form using
illustrative story. You may want to articulate the problem in form of a
boards or issue management systems.}

## Considered Options

* {title of option 1}
* {title of option 2}
* {title of option 3}
* … <!-- numbers of options can vary -->

## Decision Outcome

Chosen option: "{title of option 1}", because {justification. e.g., only op
driver | which resolves force {force} | … | comes out best (see below)}

<!-- This is an optional element. Feel free to remove. -->
### Consequences

* Good, because {positive consequence, e.g., improvement of one or r
* Bad, because {negative consequence, e.g., compromising one or mo
* … <!-- numbers of consequences can vary -->
```

```
# Use Markdown Architectural Decision Records

## Context and Problem Statement

We want to record architectural decisions made in this project.
Which format and structure should these records follow?

## Considered Options

* [MADR](https://adr.github.io/madr/) 2.1.0 - The Markdown Architectural Decision Records
* [Michael Nygard's template](http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions)
* [Sustainable Architectural Decisions](https://www.infoq.com/articles/sustainable-architectural-design-deci
* Other templates listed at <https://github.com/joelparkerhenderson/architecture_decision_record>
* Formless - No conventions for file format and structure

## Decision Outcome

Chosen option: "MADR 2.1.0", because

* Implicit assumptions should be made explicit.
  Design documentation is important to enable people understanding the decisions later on.
  See also [A rational design process: How and why to fake it](https://doi.org/10.1109/TSE.1986.6312940).
* The MADR format is lean and fits our development style.
* The MADR structure is comprehensible and facilitates usage & maintenance.
* The MADR project is vivid.
* Version 2.1.0 is the latest one available when starting to document ADRs.
```

Tools: General Purpose, Visual Studio Code

- Common options:
  - Wikis
  - Issue trackers
  - Backlog managers
- Documentation as code:
  - MADR: VSC extensions, text editors
  - Git, often GitHub or Gitlab
  - Pandoc for document conversion (optional)
- adr-tools (N. Pryce)
  - CLI for ADR file I/O
  - Nygardian template only



https://marketplace.visualstudio.com/items?itemName=StevenChen.vscode-adr-manager

https://adr.github.io/madr/tooling.html

https://adr.github.io/#decision-capturing-tools

AD Guidance (ADG) Tool

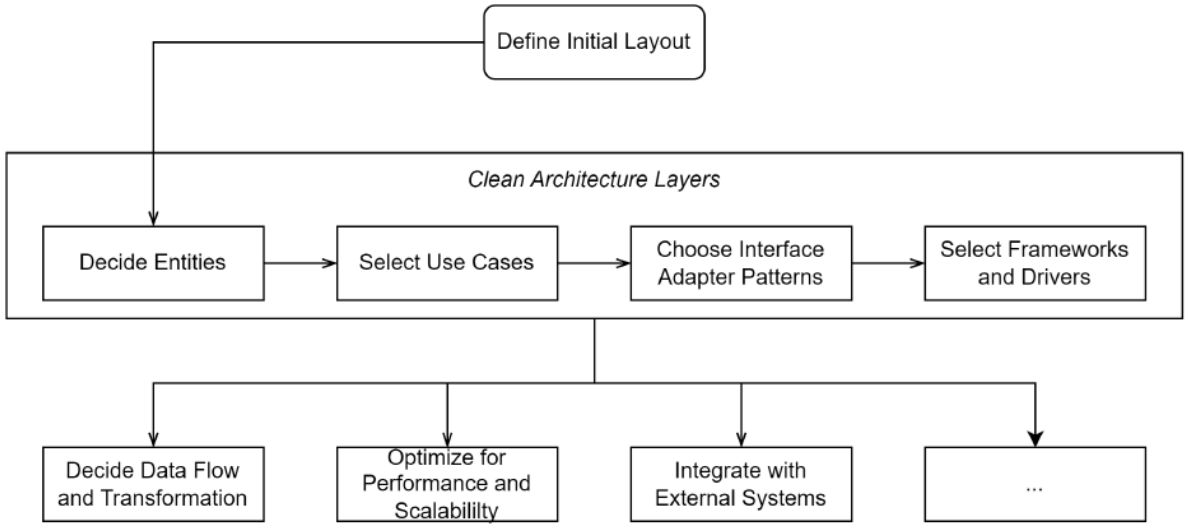
Concept Alternatives for the Management of Architectural Decisions in Clean Architectures

Raphael Schellander
Supervised by Olaf Zimmermann
Eastern Switzerland University of Applied Sciences

September 6, 2024

- R+D at OST
  - MSE student projects: ADG tool development
  - Clean Decision Handbook
  - First thesis report available
- From past tense to present and future tense:
  - 14 commands for decision guidance/reuse and decision making
  - Command-line interface

```
> adg init clean-architecture project/doc/adr
```



More AD Management Process and Practices

- Architectural significance test: 5+2 criteria
- Definition of ready for single AD: START
- “Big” AD criteria (most responsible moment)
- Definition of done for single AD: ecADR
- Advice regarding good and bad ADR practices (justifications, …)
- (emerging) AD adoption model, catalogs for domains/genres

Slides: https://ozimmer.ch/

Concepts proposed in blog posts
(no scientific publications yet)

OST
Eastern Switzerland
University of Applied Sciences

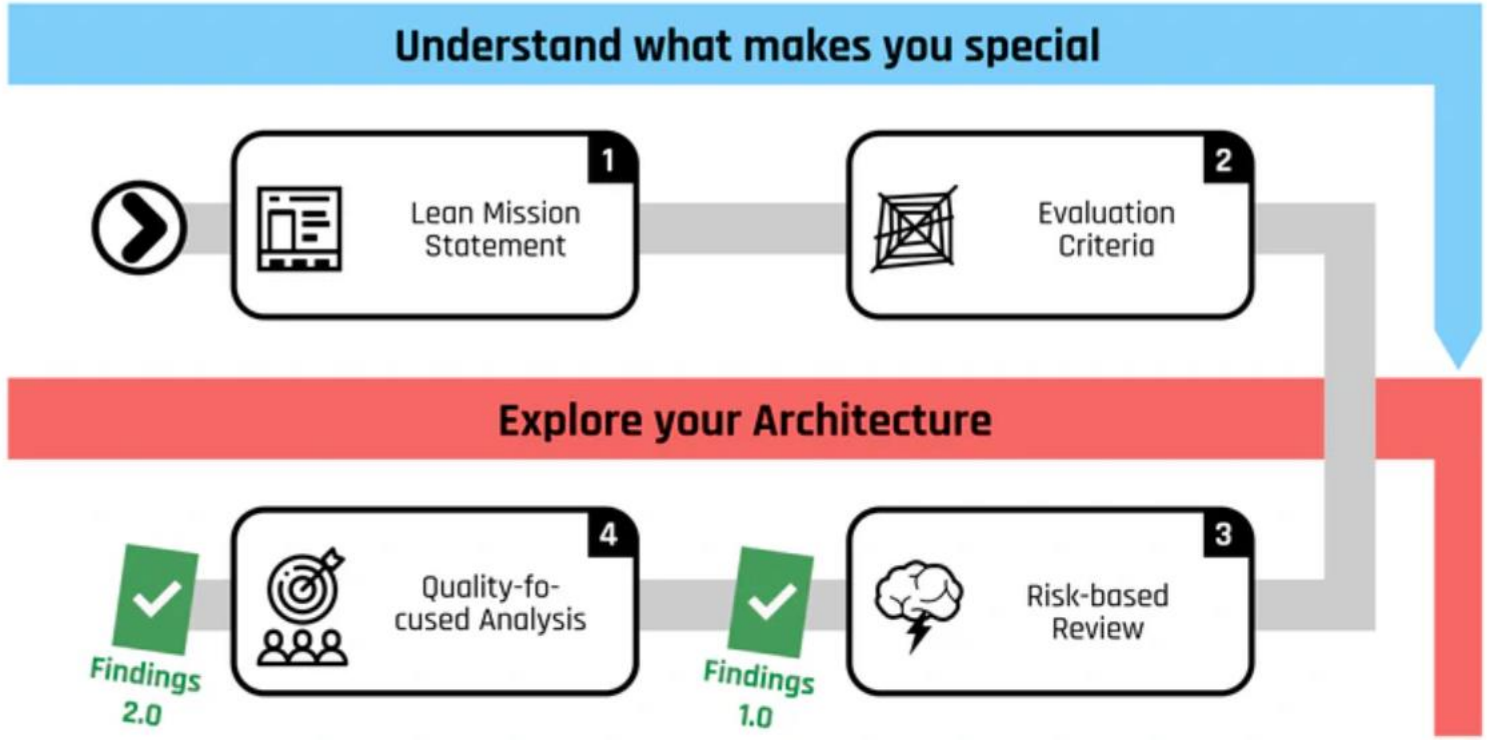
Introduction to DPR, MADR, LASR
Part R: Lightweight Approach for Software Reviews (LASR)

Dr. Olaf Zimmermann

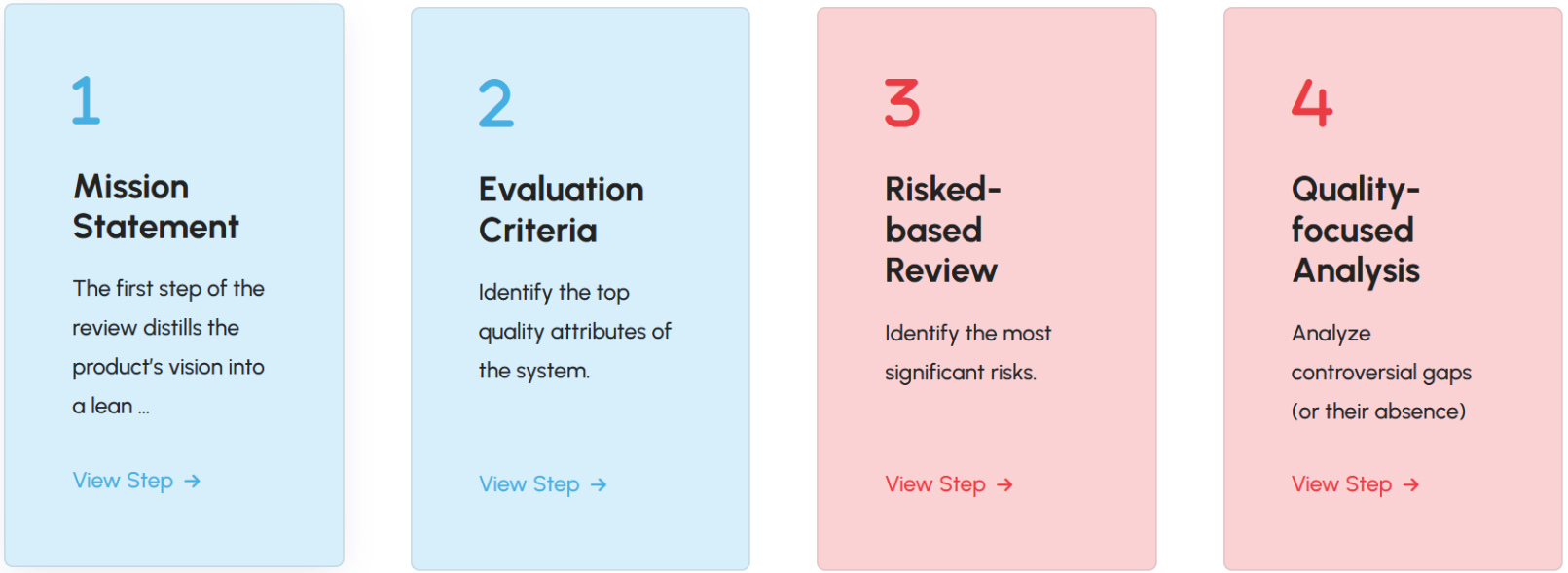
olaf.zimmermann@ost.ch (until 8/25)

LASR (by Stefan Toth and Stefan Zörner): A Lightweight Approach for Software Reviews

- https://www.lasr-reviews.org/



LASR Steps



1. Lean Mission Statement
2. Evaluation Criteria
3. Risked-based Review
4. Quality-Focused Analysis

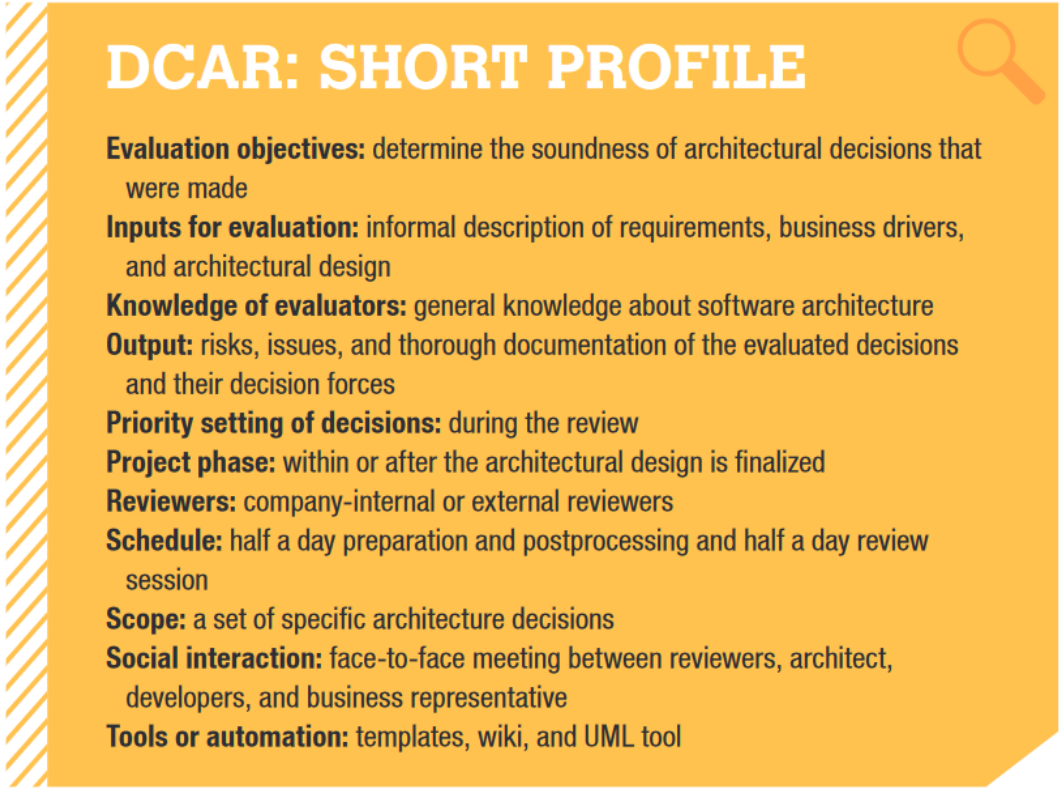
https://www.lasr-reviews.org/

LASR Tools/Resources

- 4-page *cheat sheet*:
  - Page 1: LASR Introduction and Overview
  - Page 2: Understand what makes you special
  - Page 3: Explore your architecture
  - Page 4: Optional deep dive (LASR+):
    - Method maturity
    - Tool-based evaluation
    - Constraints
    - Organizational fitness

Check out the book, the process diagrams, the card set/other tools

Alternative: DCAR



DCAR: SHORT PROFILE

Evaluation objectives: determine the soundness of architectural decisions that were made
Inputs for evaluation: informal description of requirements, business drivers, and architectural design
Knowledge of evaluators: general knowledge about software architecture
Output: risks, issues, and thorough documentation of the evaluated decisions and their decision forces
Priority setting of decisions: during the review
Project phase: within or after the architectural design is finalized
Reviewers: company-internal or external reviewers
Schedule: half a day preparation and postprocessing and half a day review session
Scope: a set of specific architecture decisions
Social interaction: face-to-face meeting between reviewers, architect, developers, and business representative
Tools or automation: templates, wiki, and UML tool

Decision-Centric Architecture Reviews (DCAR)

Uwe van Heesch, Capgemini Germany

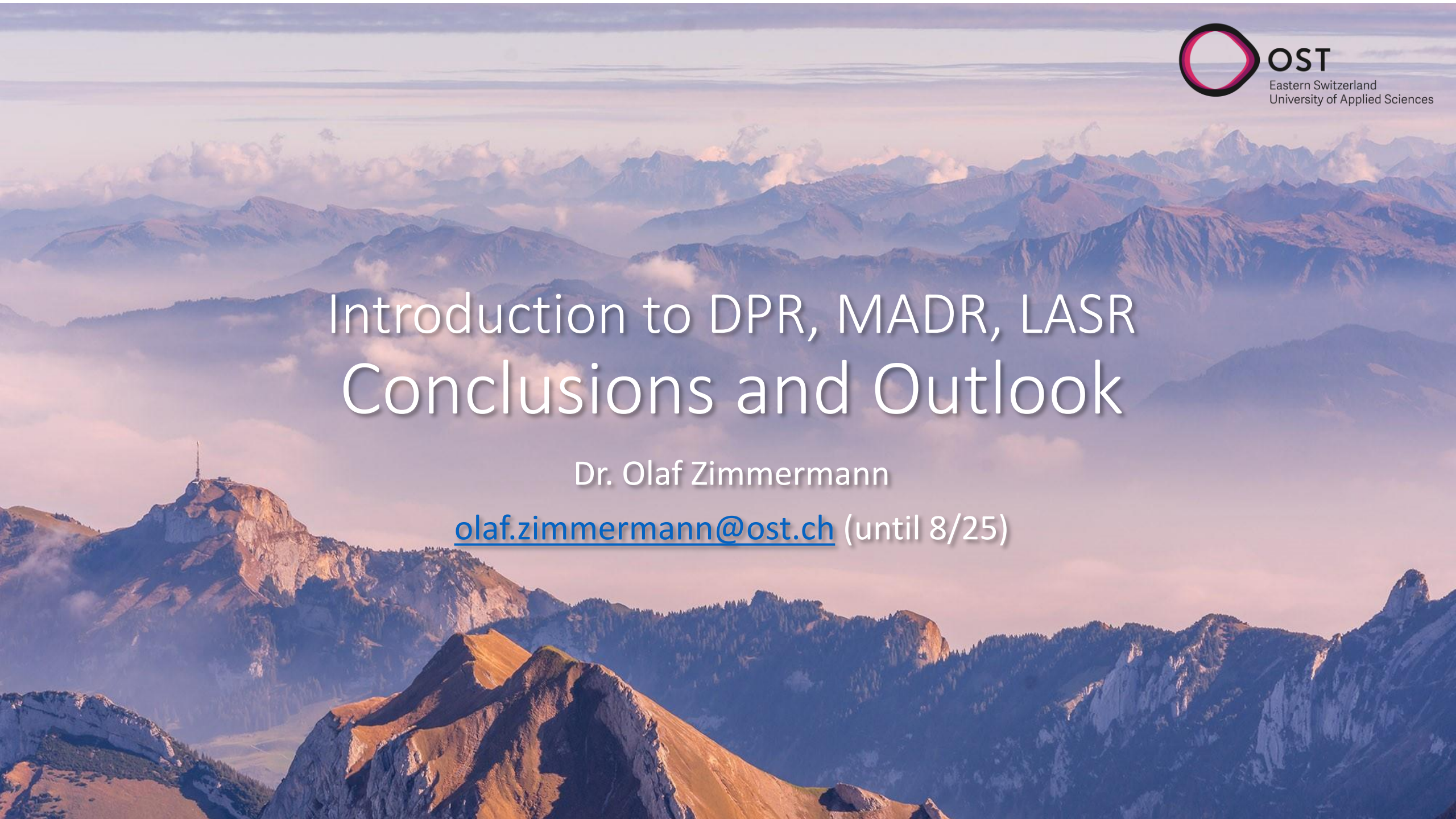
Veli-Pekka Eloranta, Tampere University of Technology

Paris Avgeriou, University of Groningen

Kai Koskimies, Tampere University of Technology

Neil Harrison, Utah Valley University

- Other architecture-centric options include:
  - ATAM (SEI)
  - Tara (Eoin Woods)
  - Academic works (e.g., ARID)



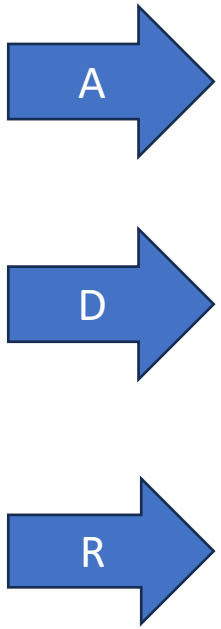
OST
Eastern Switzerland
University of Applied Sciences

Introduction to DPR, MADR, LASR
Conclusions and Outlook

Dr. Olaf Zimmermann

olaf.zimmermann@ost.ch (until 8/25)

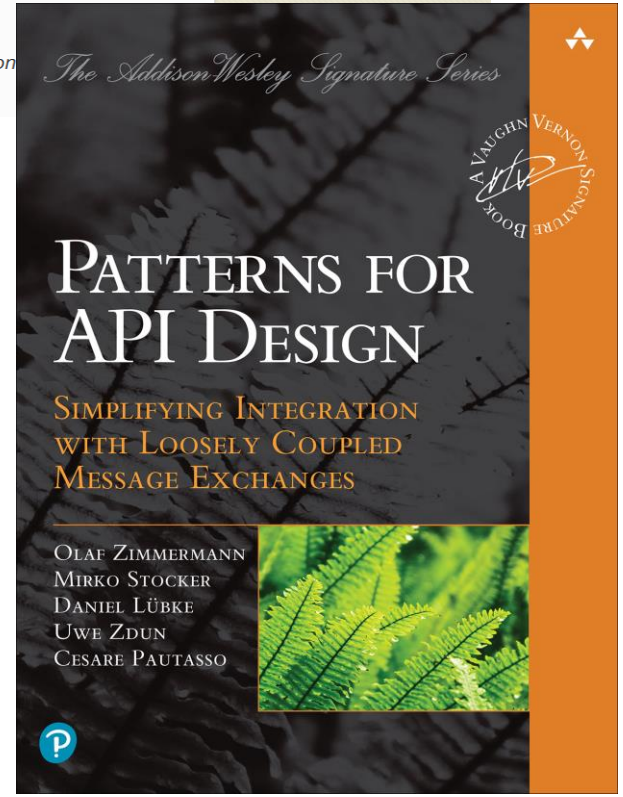
Σ: A, D, R Steps and Community Toolbox



- **Step A: Analyze and architect**
  - User stories or use cases and SMART NFRs
  - Tactic and strategic DDD
  - Architecture modeling, C4
- **Step D: Decide and document**
  - Decide consciously and adequately
  - Document efficiently and reproducibly (e.g., arc 42)
- **Step R: Review and refine**
  - Execute pragmatically and diligently
  - Enforce and review constructively (not authoritatively)

Books, Open Repos

- "Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges" (website)
  - Arch. Significant Requirements (ASRs) in API design
  - Many decisions (about APIs), Y-statements
  - 44 patterns, focus on message content
- Design Practice Reference: e-book on LeanPub
  - Y-Statements, SMART NFRs, Architecture Modeling (C4 plus), DDD, … (content also available on GitHub/GitPages)



Thank You & Keep in Touch

Discussion time!

- I hope you find a few ideas to take away from this presentation
- I will be happy to answer questions and discuss arch. decisions of all kinds, as well as other topics – after the talk? Later on?

- https://medium.com/olzzio
- https://www.linkedin.com/in/ozimmer/
- mailto:olaf.zimmermann@ost.ch
- mailto:itolz@bluewin.ch