OST
Eastern Switzerland
University of Applied Sciences

Slides:
https://ozimmer.ch

Timing Architectural Decisions (ADs)

ITARC 2024

Dr. Olaf Zimmermann
olaf.zimmermann@ost.ch

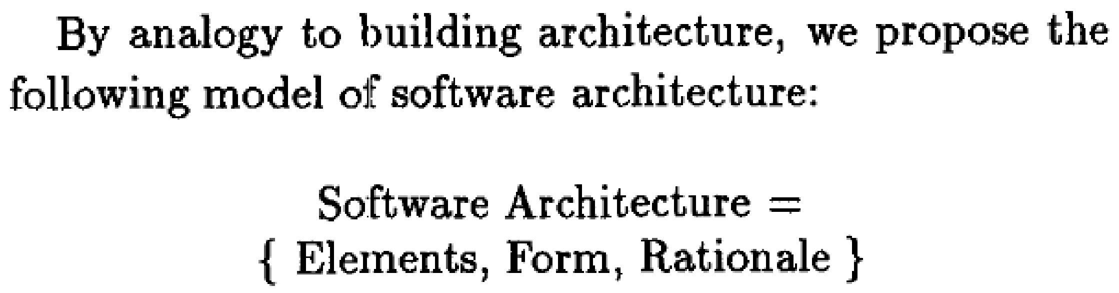
The Open Group Open Certified
Distinguished Architect

Take-Away Messages of This Talk

1. Making Architectural Decisions (ADs) is a key IT architect responsibility
   - Fundamental concept, as old as the field, but revived more recently
2. AD mileage varies – one size (AD management practices) does not fit all
   - Notion of Most Responsible Moment (MRM), many “big” ADs have an early MRM
3. ADs should be justified in ADRs
   - Record answers to «why?» questions
4. Agile architecting embraces ADs
   - Light templates, definitions of ready and done
5. You can start small and grow responsibly
   - Five AD adoption levels; ethics as architecturally significant requirements

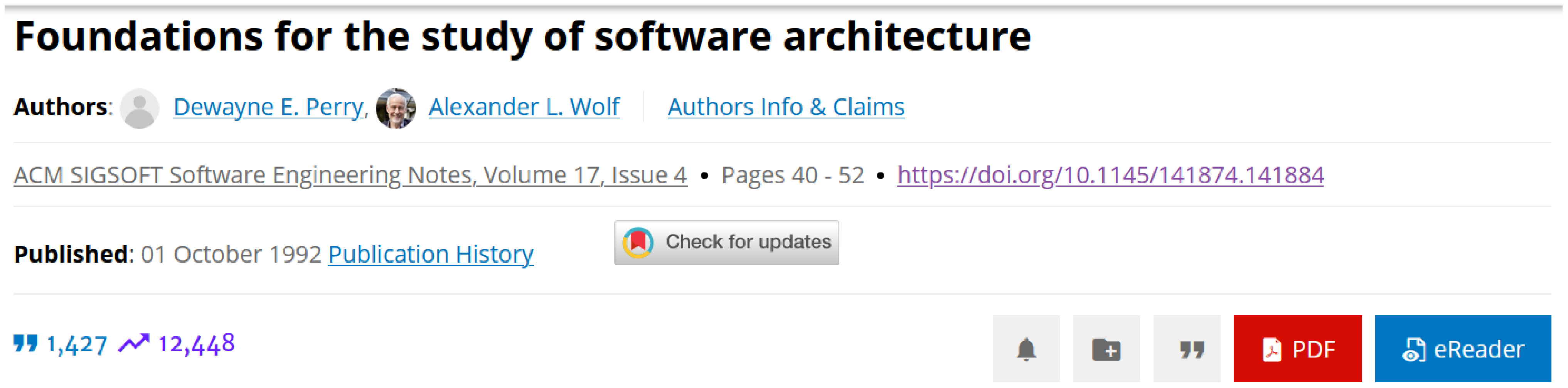
Blog posts about these topics:
https://ozimmer.ch/tags/#architectural-decisions

The Making of ADs: Inception (1992-1998)



By analogy to building architecture, we propose the following model of software architecture:

$$\text{Software Architecture} = \{ \text{Elements, Form, Rationale} \}$$



Foundations for the study of software architecture

Authors: Dewayne E. Perry, Alexander L. Wolf | Authors Info & Claims

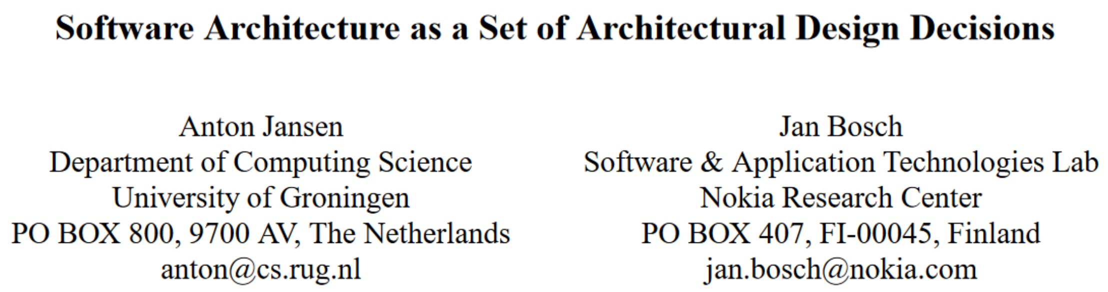
ACM SIGSOFT Software Engineering Notes, Volume 17, Issue 4 • Pages 40 - 52 • https://doi.org/10.1145/141874.141884

Published: 01 October 1992 Publication History

Reference: https://dl.acm.org/doi/10.1145/141874.141884

ADs complement structural view (since 2004)

- Explicit vs. tacit knowledge
  - Risk of knowledge vaporization

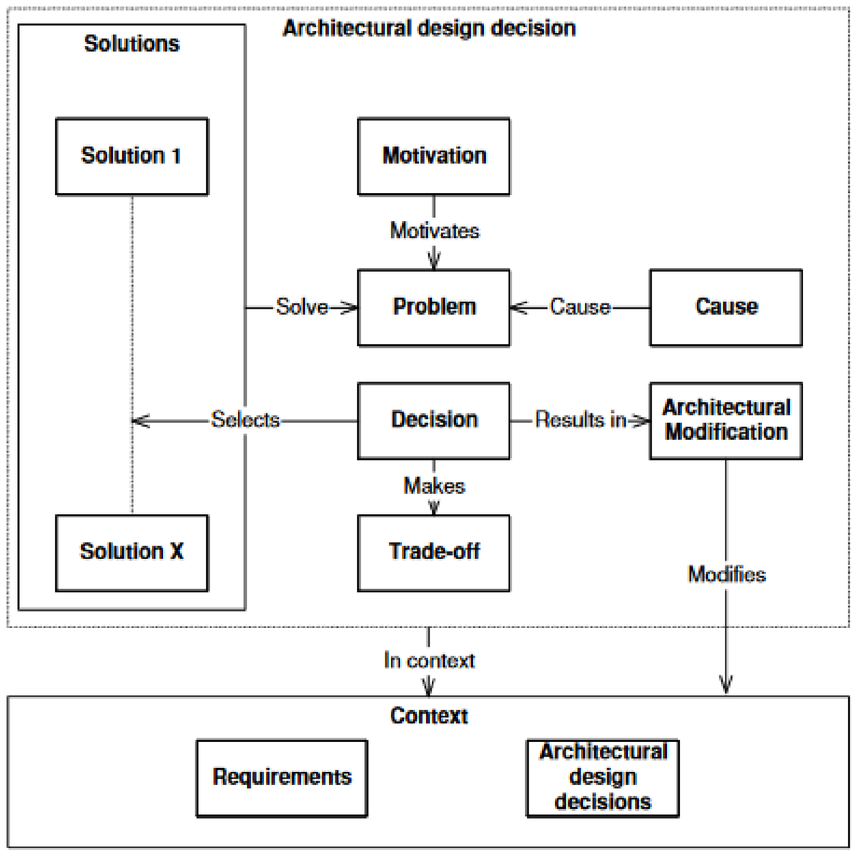


Software Architecture as a Set of Architectural Design Decisions

Anton Jansen
Department of Computing Science
University of Groningen
PO BOX 800, 9700 AV, The Netherlands
anton@cs.rug.nl

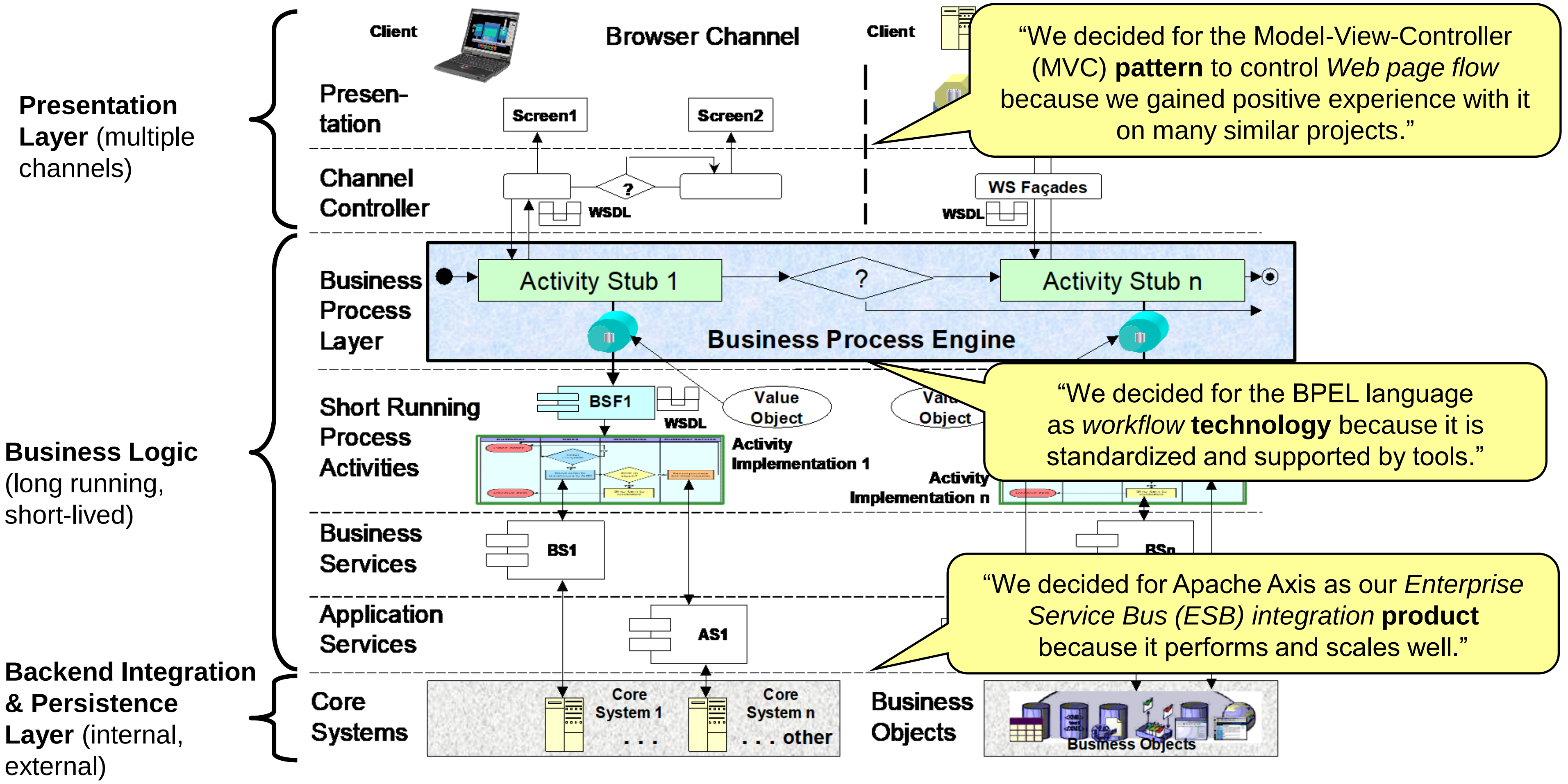
Jan Bosch
Software & Application Technologies Lab
Nokia Research Center
PO BOX 407, FI-00045, Finland
jan.bosch@nokia.com

- ISO/IEC/IEEE [42010]() Definition:
  - Structure *and* decision rationale
  - Rationale first included in now superseded IEEE [1471]()



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 Canonical Enterprise Application



"We decided for the Model-View-Controller (MVC) **pattern** to control *Web page flow* because we gained positive experience with it on many similar projects."

"We decided for the BPEL language as *workflow* **technology** because it is standardized and supported by tools."

"We decided for Apache Axis as our *Enterprise Service Bus (ESB) integration* **product** because it performs and scales well."

Timing Architectural Decisions (ADs): Presentation Flow

I: AD Identification and Prioritization

- Architectural Significance Test
- Most Responsible Moment (MRM)
- Big ADs

II: AD Making and Capturing

- AD Definition of Ready
- Architectural Decision Record (ADR) Templates
- Patterns and anti-patterns

III: AD Execution and Review

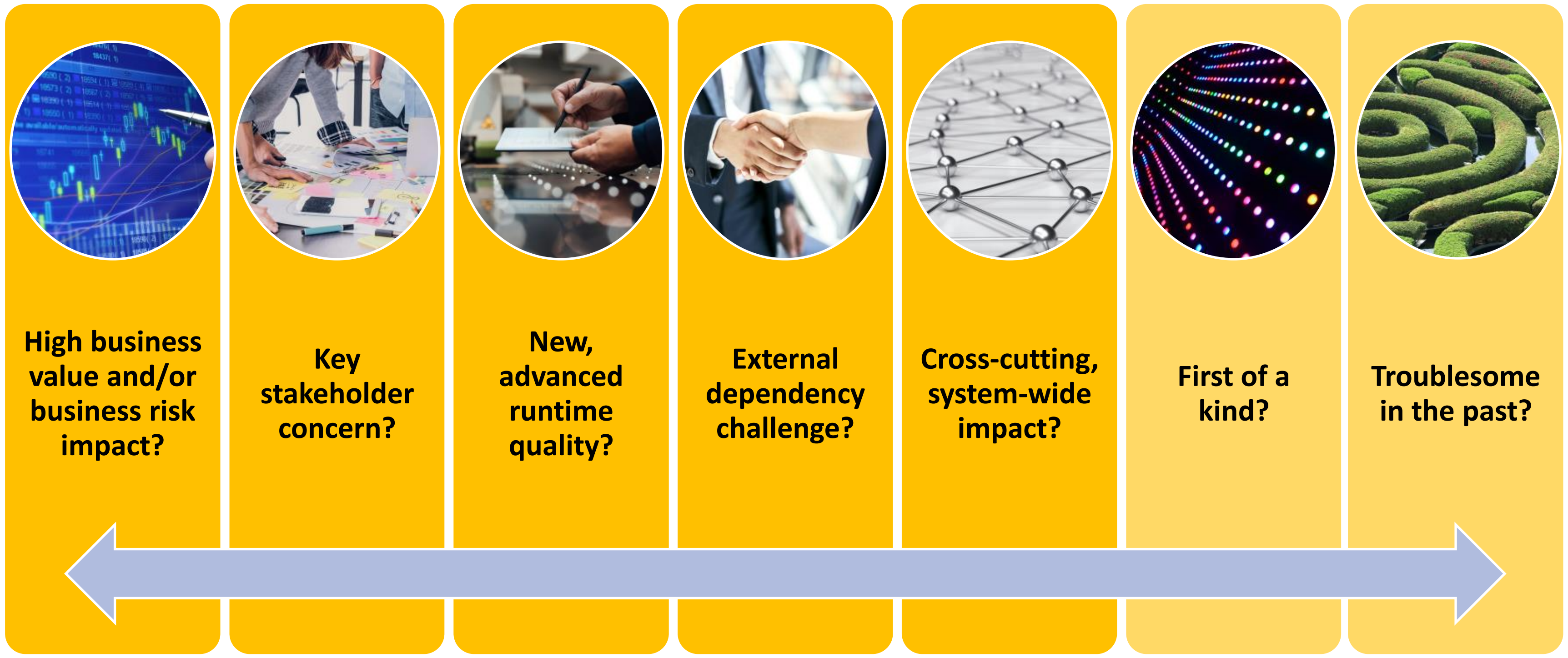
- AD Definition of Done
- Review practices: DCAR, ATAM
- Patterns and anti patterns (for reviews)

I: Decision Identification and Prioritization

Architectural Significance, Big ADs (Early MRM)

Architectural Significant Requirements (ASRs): 5+2 Criteria in ASR Test

- To prioritize requirements, decisions, change requests, design elements


High business value and/or business risk impact?
Key stakeholder concern?
New, advanced runtime quality?
External dependency challenge?
Cross-cutting, system-wide impact?
First of a kind?
Troublesome in the past?

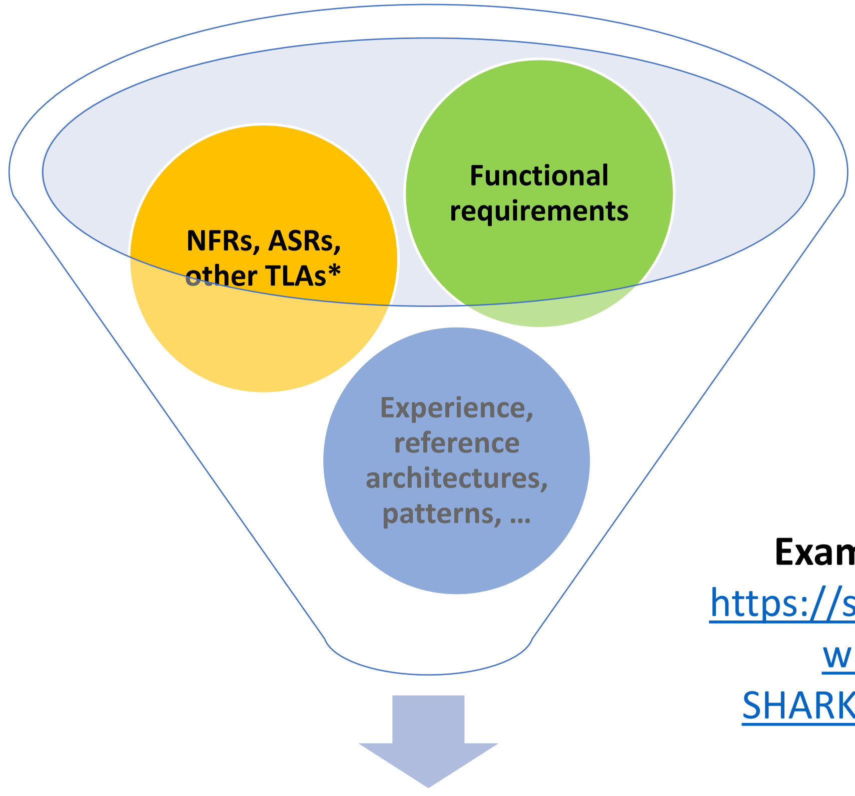
https://medium.com/olzzio/architectural-significance-test-9ff17a9b4490

***TLA:** Three-Letter Acronym

Start from ASRs and prior knowledge

- **“The hard stuff”** (M. Fowler) that is **“costly to change”** (G. Booch):
  - In/out decisions, external interfaces
  - Patterns and concepts
    - Per layer, all viewpoints
  - Technologies and standards choices
  - Products and open-source project selection

Project goals, product vision

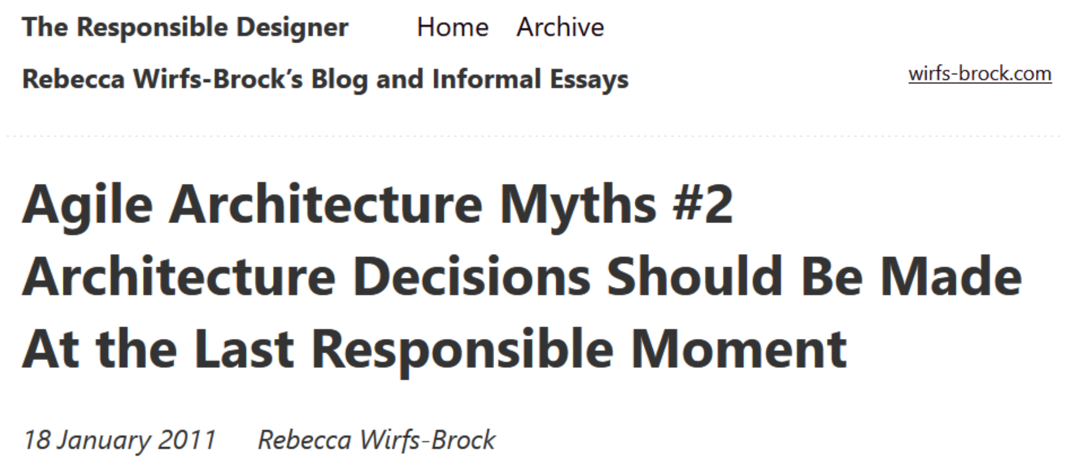


ADs (possibly) required

Examples: Table 2 in https://soadecisions.org/download/SOAD-SHARK2012v13Final.pdf

Last/Most Responsible Moment (LRM/MRM)

- LRM is a principle from "Lean Software Development" book
  - Good: avoids analysis paralysis
  - Bad: risk of procrastination in the name of flexibility


The Responsible Designer   Home   Archive

Rebecca Wirfs-Brock's Blog and Informal Essays   wirfs-brock.com

Agile Architecture Myths #2 Architecture Decisions Should Be Made At the Last Responsible Moment

18 January 2011   Rebecca Wirfs-Brock

Most responsible might still be early – when has "time is right" come?

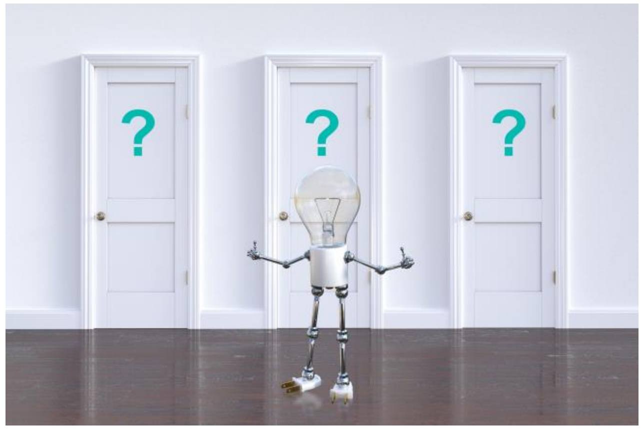
"[…] make decisions when the time is right. Which can be hard to figure out sometimes. That's what makes development challenging. Decisions shouldn't be forced or delayed, but taken up when the time is right. And to help me find the right times, I prefer the mindset of '**the most responsible moment**' not the 'last responsible one.'"

https://wirfs-brock.com/blog/2011/01/18/agile-architecture-myths-2-architecture-decisions-should-be-made-at-the-last-responsible-moment/

Thought Experiment: Criteria for Early MRM

Tunnel or dead end, effort to revise

(too) early



(too) late

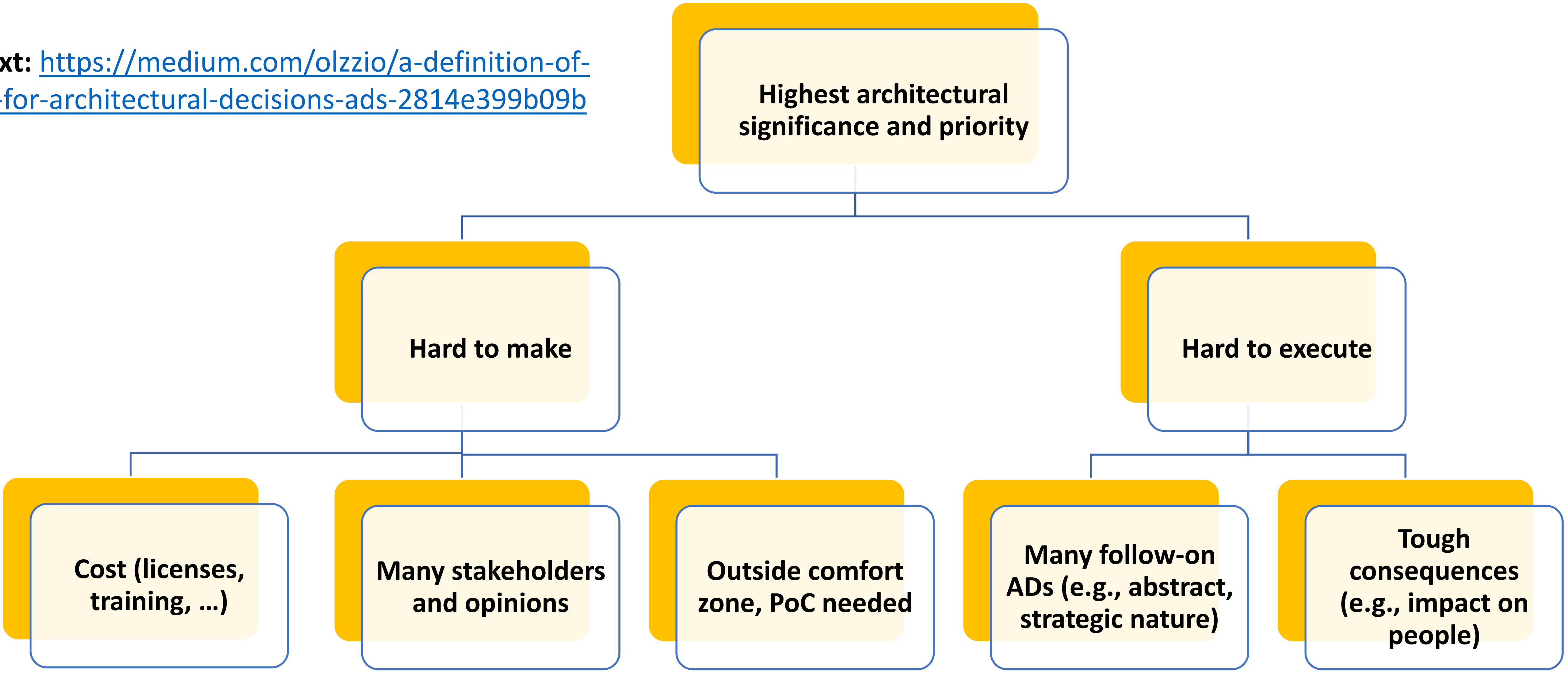
Conceptual integrity at risk, technical debt

- Which decisions cannot wait very long? And why?
  - For example, which ADs are required in Inception/Elaboration (in UP terms)?
  - Which ADs cannot wait to made in late sprints?

Think about your position and experience please (1-2 mins)

My Thoughts on "Big" ADs (Early MRM)

Full text: https://medium.com/olzzio/a-definition-of-ready-for-architectural-decisions-ads-2814e399b09b


Highest architectural significance and priority
Hard to make
Hard to execute
Cost (licenses, training, …)
Many stakeholders and opinions
Outside comfort zone, PoC needed
Many follow-on ADs (e.g., abstract, strategic nature)
Tough consequences (e.g., impact on people)

Some More Prioritization Heuristics

- Worst first
- Simple-complex-simple
- Urgent-important, followed by mix of not-urgent-but-important and/or urgent-but-not-important

https://arnon.me/2010/05/utility-trees-hatching-quality-attributes/

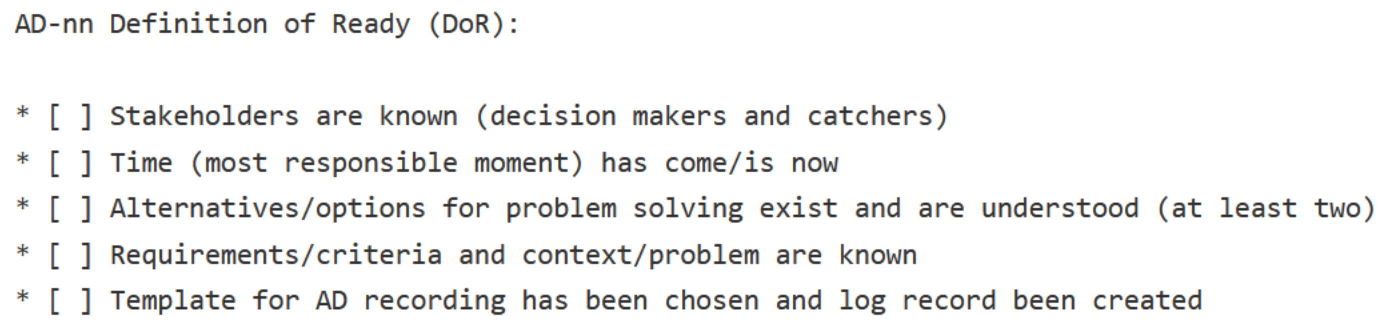
- SEI quality tree: importance/value x technical risk/difficulty (H, H) etc.
- MoSCoW from requirements analysis/project management
- Value x effort scores
  - Start with (H, L) followed by (H, M) and then (H, H), (M, L)

OST
Eastern Switzerland
University of Applied Sciences

II: Decision Making and Capturing

Definition of Ready, Y-Statements, (Anti) Patterns

AD Definition of Ready (DoR)



```
AD-nn Definition of Ready (DoR):

* [ ] Stakeholders are known (decision makers and catchers)
* [ ] Time (most responsible moment) has come/is now
* [ ] Alternatives/options for problem solving exist and are understood (at least two)
* [ ] Requirements/criteria and context/problem are known
* [ ] Template for AD recording has been chosen and log record been created
```

- START criteria, inspired by agile DoR for features/stories/issues

- **Stakeholders identified (and contacted)**
- **Timing is right (MRM, Step I on AD Identification and Prioritization!)**
- **Alternatives identified (decision options)**
- **Requirements/decision drivers defined (ASR Test from Step I!)**
- **Template chosen (ADR format)**

Definition of START criteria, example and template: https://medium.com/olzzio/a-definition-of-ready-for-architectural-decisions-ads-2814e399b09b

How do other disciplines decide/work?

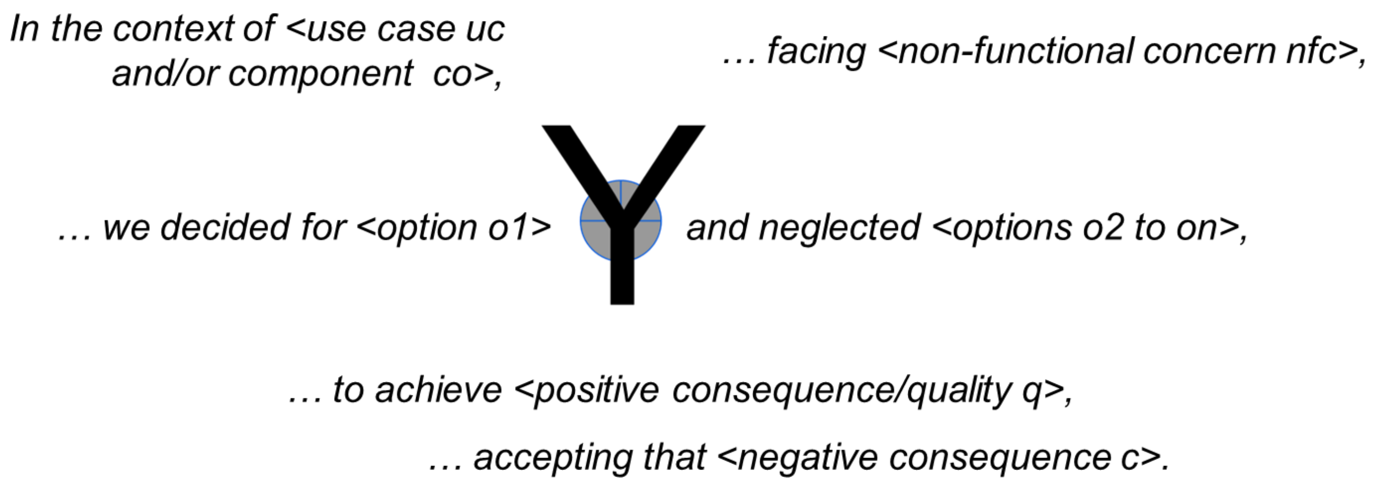
- Civil architecture:
    - Section 2.3.1 in Perry/Woolf paper has a comparison, draws analogies
    - Patterns and pattern languages to norm solution space (Christopher Alexander)
- Mechanical and electrical engineering («genius» is in the word!):
    - Systems engineering processes, handbooks
    - Interdisciplinary method collection: td-net toolbox
- Public health care:
    - Evidence-based medicine
    - “Factfulness” book by Hans Rosling, Anna Rosling Rönnlund, Ola Rosling
- Psychology/economics:
    - Cognitive biases, thinking fast and slow (Daniel Kahneman et al)
    - Red, yellow, blue, green communication styles (SbI book)

Two ADR Templates: Nygard and WH(Y)

- Cognitect Blog 2011: “ADR”
    1. Context
    2. Outcome
    3. Status
    4. Consequences
- SEI SATURN 2012: “WH(Y)”
    1. Two-part context
    2. Chosen and neglected options
    3. Good and bad consequences

DOCUMENTING ARCHITECTURE DECISIONS

Michael Nygard - November 15, 2011



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>.

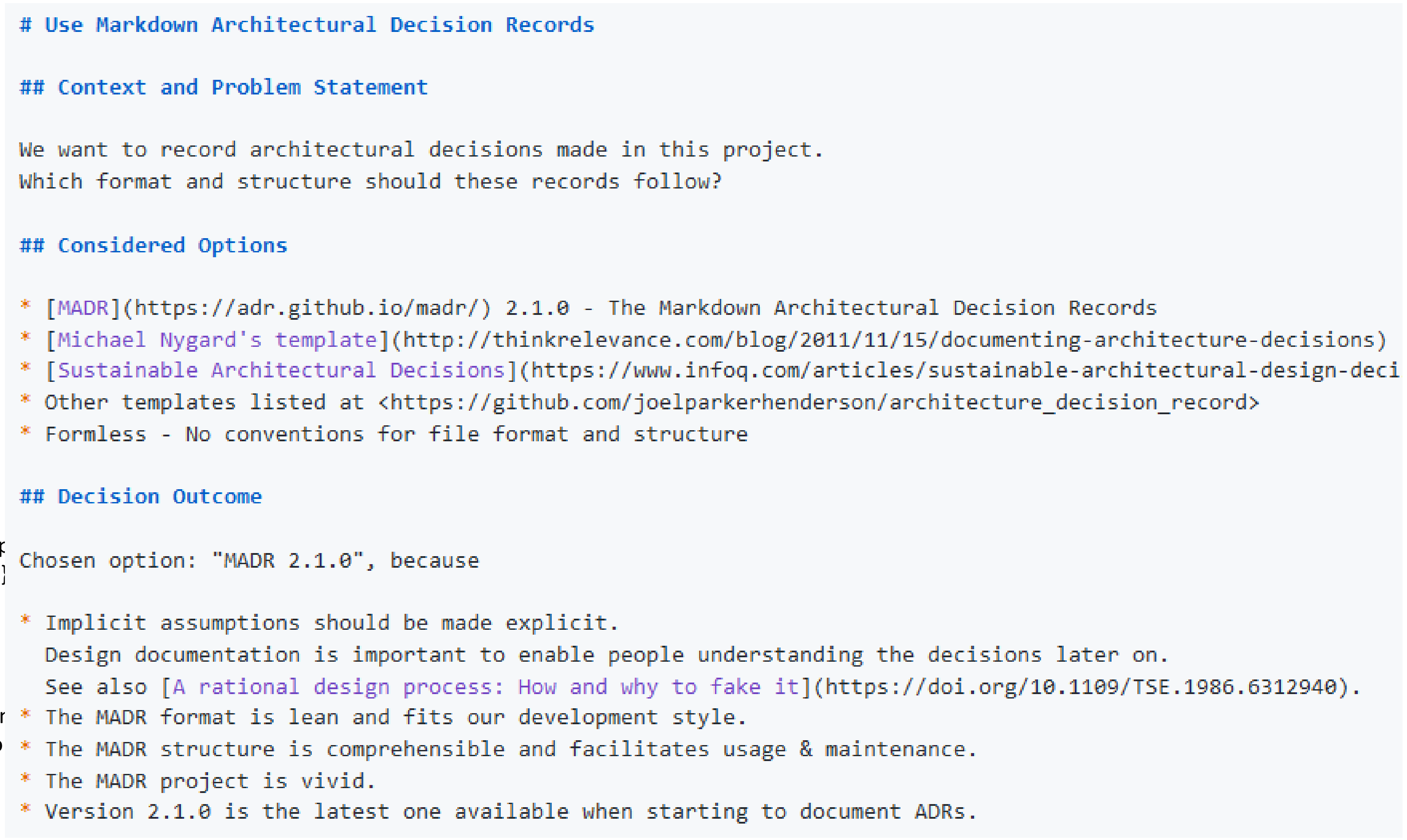
Sample WH(Y): Logical Decomposition

- *In the context of* the entire order management system,
- *facing* the need to organize the overall system and manage complexity,
- *we decided* for the Layer-based decomposition pattern
- *and neglected* other decomposition pattern such as pipes-and-filters or process-based decomposition (workflow)
- *to achieve* a) high flexibility regarding technology selections within the layers (changeability) and b) that teams can work on system parts in parallel
- *accepting that* there might be a performance penalty for each level of indirection and some undesired replication of implementation artifacts.

Markdown ADRs (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.
```

Blog Post: "ADR Patterns and Anti Patterns"

Metaphors

Journal *Action Plan* *Contract*

Executive Summary *Verdict/Scale* *Letter of Intent*

Anti Patterns (Don't)

Fairy Tale, Sales Pitch, Free Lunch, Dummy Alternative

Sprint, Tunnel Vision, Maze

Blueprint or Policy in Disguise, Mega-ADR, Novel/Epic

Magic Tricks:
False Urgency, Problem-Solution Mismatch, Pseudo-Accuracy

Good Practices (Do)

Prioritization and timing by significance

Meta Qualities, objectivity and bias awareness

Tradeoff analysis, editorial quality of ADR, rightsizing

Slicing in stages, disclosure of confidence level

Author Pledge

Architectural Significance Check, Template Use, Explicit Question, Option, Criteria
Presentation Quality: Thorough, Focused, Factual
Candor/Honesty

Related Advice (Other Posts)

Architectural Significance Test, MADR or Y-Statements, Definition of Done
Good ADR review practices

OST
Eastern Switzerland
University of Applied Sciences

Steps I and II Applied

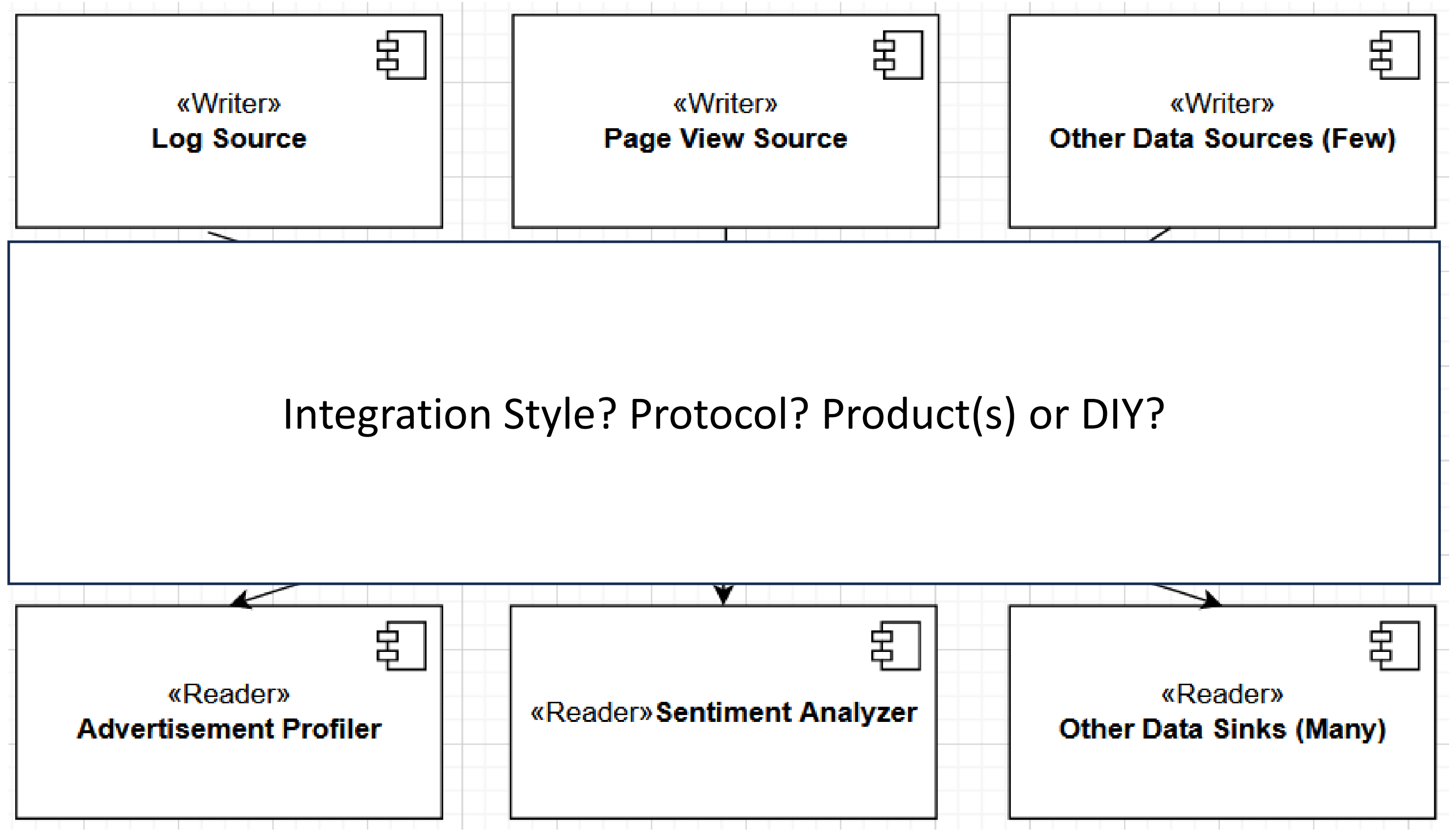
Social Network Case Study

Sample Case: EAI/EDA at a Social Network

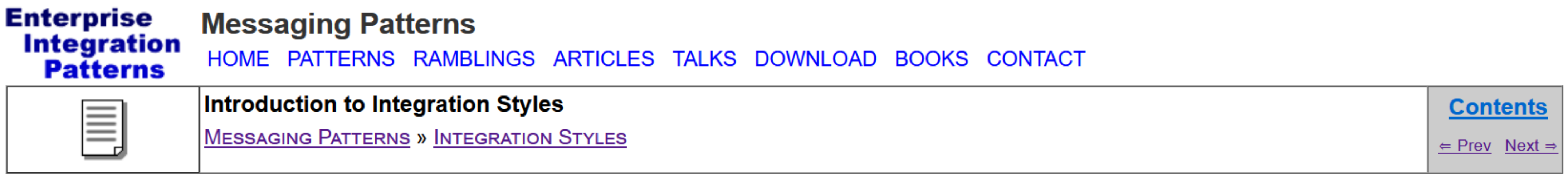
- Few applications for content creation (by users), writing events
- Many applications analyzing activities, reading events; even more such applications will be developed in the future
  - Most readers are interested in events from multiple writers (n:m relation)
  - Multiple programming languages in use, diverging data definitions and formats
  - Application availability and application lifetimes differ
  - Several data centers hosting the applications (on premises, in the cloud)
- Data variety: from logs to UI impressions/clicks/page views (apps, PC)
- Thousands (1000s) of posts per second, millions (1000000s) of reads
  - >1 billion registered users, not all of which are active at the same time
- High throughput required (due to data volumes)

EAI at Social Network: AD Required

Which middleware patterns and technologies should be used to inform consumer applications (bottom) about new posts and other user activities (top)?



Interlude: Sample Case


Enterprise Integration Patterns

Messaging Patterns

HOME PATTERNS RAMBLINGS ARTICLES TALKS DOWNLOAD BOOKS CONTACT

Introduction to Integration Styles

Messaging Patterns » Integration Styles

Contents

⇐ Prev Next ⇒

AD Options

- Point-to-Point RPC?
- Pub-Sub Messaging?
- ~~File Transfer?~~
- ~~Shared Database?~~
- Web-based Data Transfer Resource?
- Distributed Tx Log and Streaming?

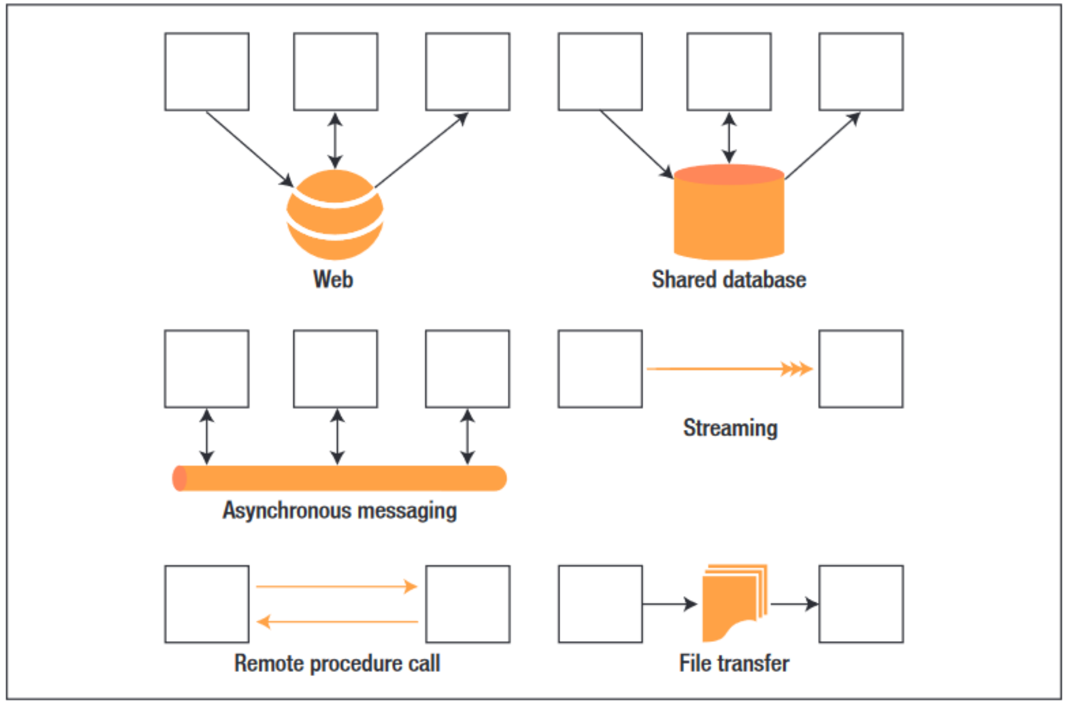


Figure taken from "The Web as Software Connector":
https://ieeexplore.ieee.org/document/8239944

Decision Rationale in Sample Case

- Version A: Informal, no template used (one sentence):
    - "We decided for Apache Kafka as our event sourcing infrastructure for future big data analytics projects because it offers publish-subscribe semantics, supports time-based reasoning capabilities and scales well."
- Version B: Pattern-oriented ADR template used (1-2 pages max.):
    - *Title:* "ADR-01: Kafka as global event messaging infrastructure."
    - *Context:* "Kafka was presented at recent conference, apparently the speaker was very convincing. Our ActiveMQ administrators are bored."
    - *Decision:* "We will replace all existing messaging software with Kafka topics, including request-reply channels (point-to-point connections)."
    - *Status:* "decided"
    - *Consequences:*
        - Good: "I can add Kafka to the skills section of my CV."
        - Neutral: "We are in line with what everybody does these days."
        - Bad: "We will have to re-implement all existing messaging endpoints and migrate all data in transit. Test cases and audit procedures will have to be revisited too."

OST
Eastern Switzerland
University of Applied Sciences

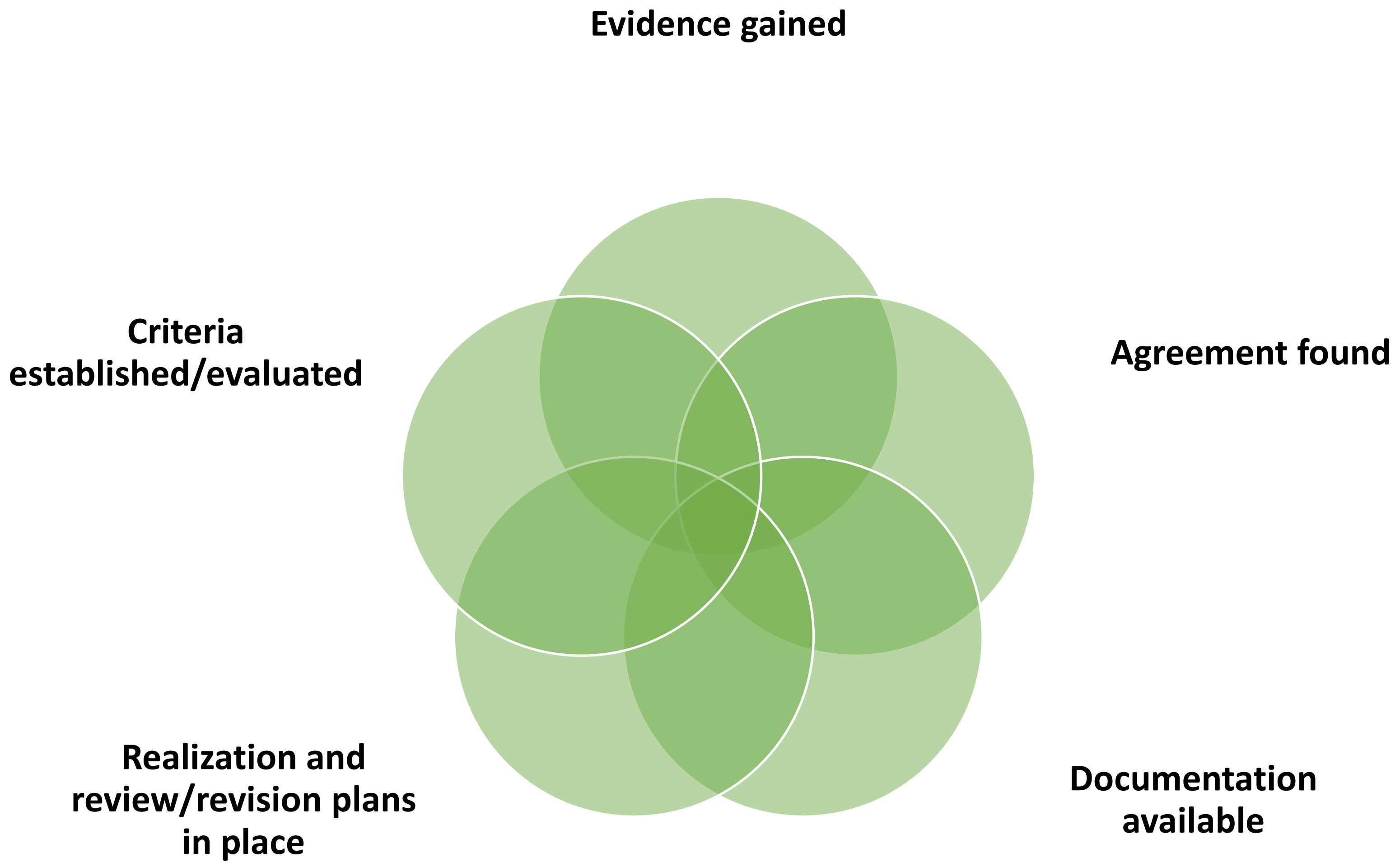
III: Decision Execution and Review

Definition of Done (DoD), DCAR, (Anti) Patterns

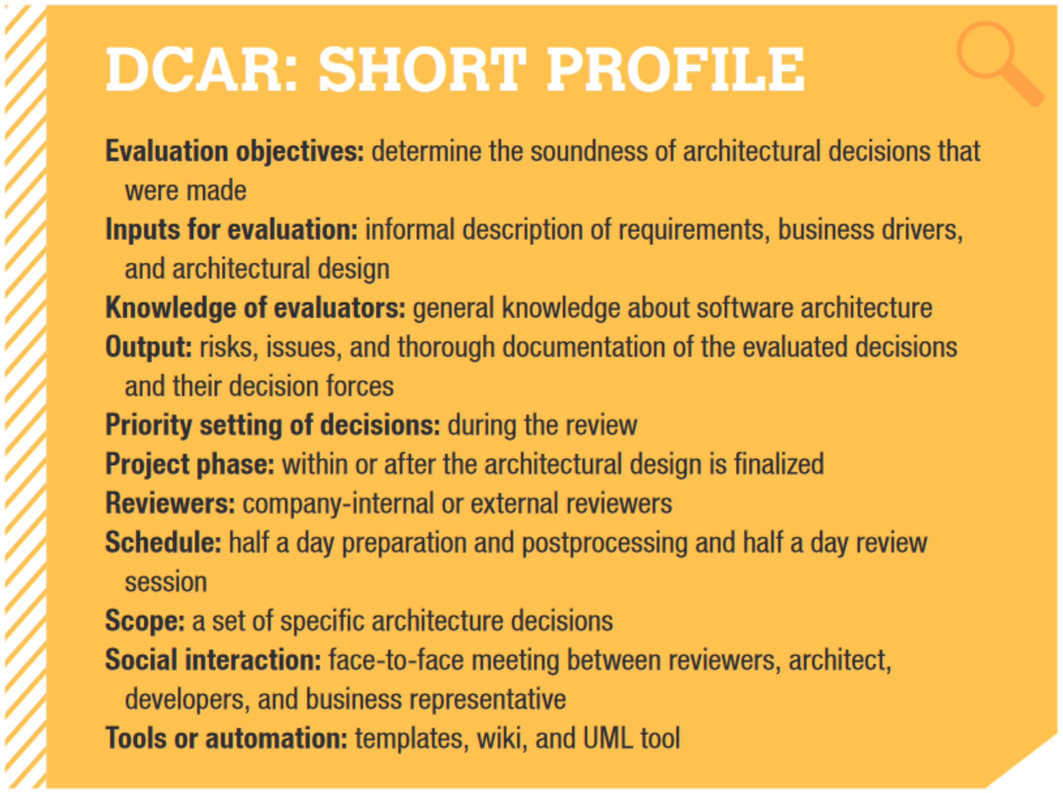
AD Definition of Done

- Inspired by Agile DoD for features

Definitions and example:
"A Definition of Done for Architectural Decisions" on [Medium]()


Evidence gained
Agreement found
Documentation available
Realization and review/revision plans in place
Criteria established/evaluated

AD rationale unveils a lot!



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

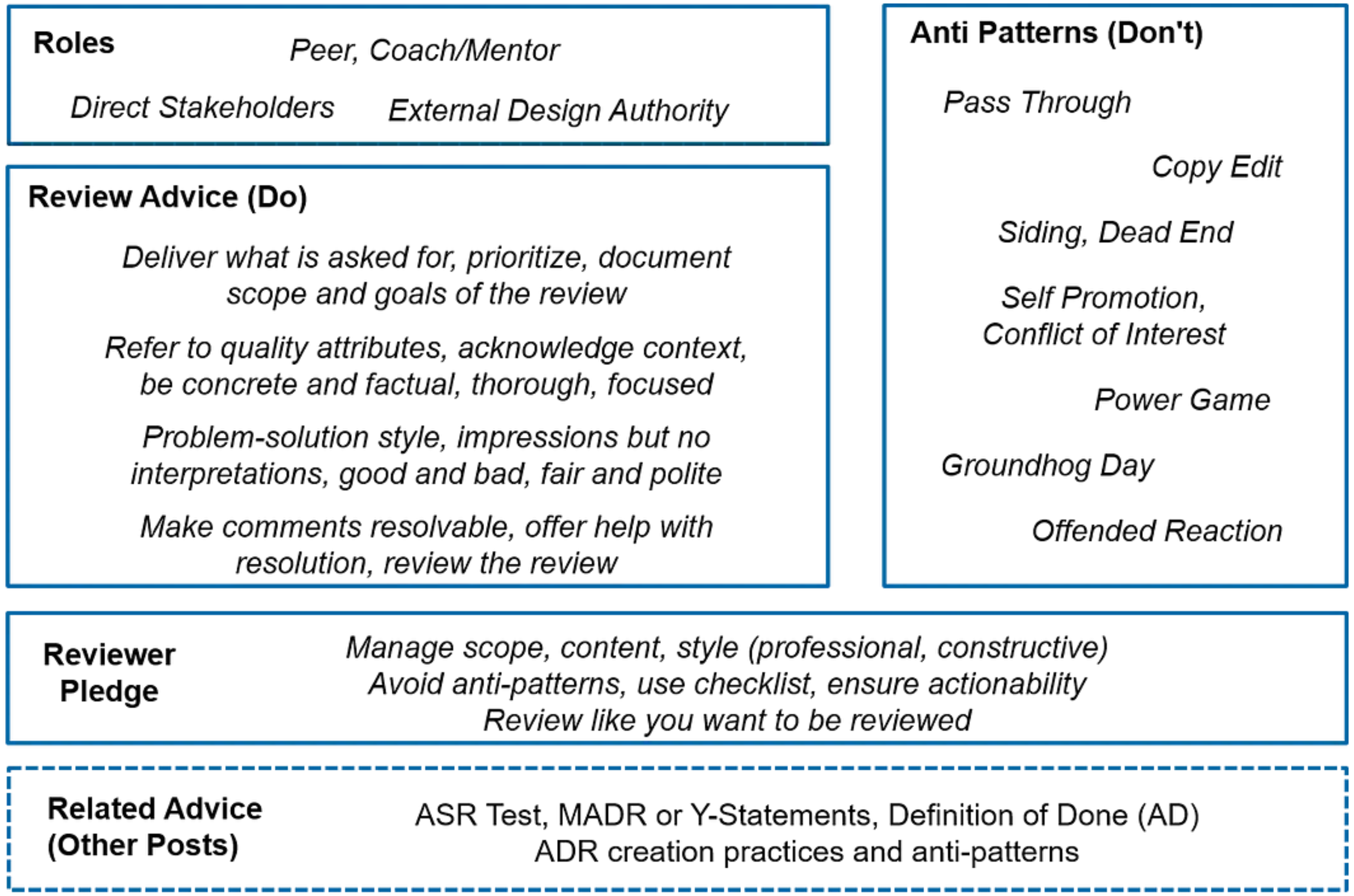
Paris Avgeriou, University of Groningen

Kai Koskimies, Tampere University of Technology

Neil Harrison, Utah Valley University

- Other options include:
  - ATAM (SEI), Tara (E. Woods)
  - Lightweight Approach for Software Reviews (LASR)
    - by S. Toth and S. Zoerner
    - in German (for the time being)

Blog post: “How to review Architectural Decision Records (ADRs) — and how not to”



Roles

Peer, Coach/Mentor

Direct Stakeholders *External Design Authority*

Review Advice (Do)

Deliver what is asked for, prioritize, document scope and goals of the review

Refer to quality attributes, acknowledge context, be concrete and factual, thorough, focused

Problem-solution style, impressions but no interpretations, good and bad, fair and polite

Make comments resolvable, offer help with resolution, review the review

Anti Patterns (Don't)

Pass Through

Copy Edit

Siding, Dead End

Self Promotion, Conflict of Interest

Power Game

Groundhog Day

Offended Reaction

Reviewer Pledge

Manage scope, content, style (professional, constructive)
Avoid anti-patterns, use checklist, ensure actionability
Review like you want to be reviewed

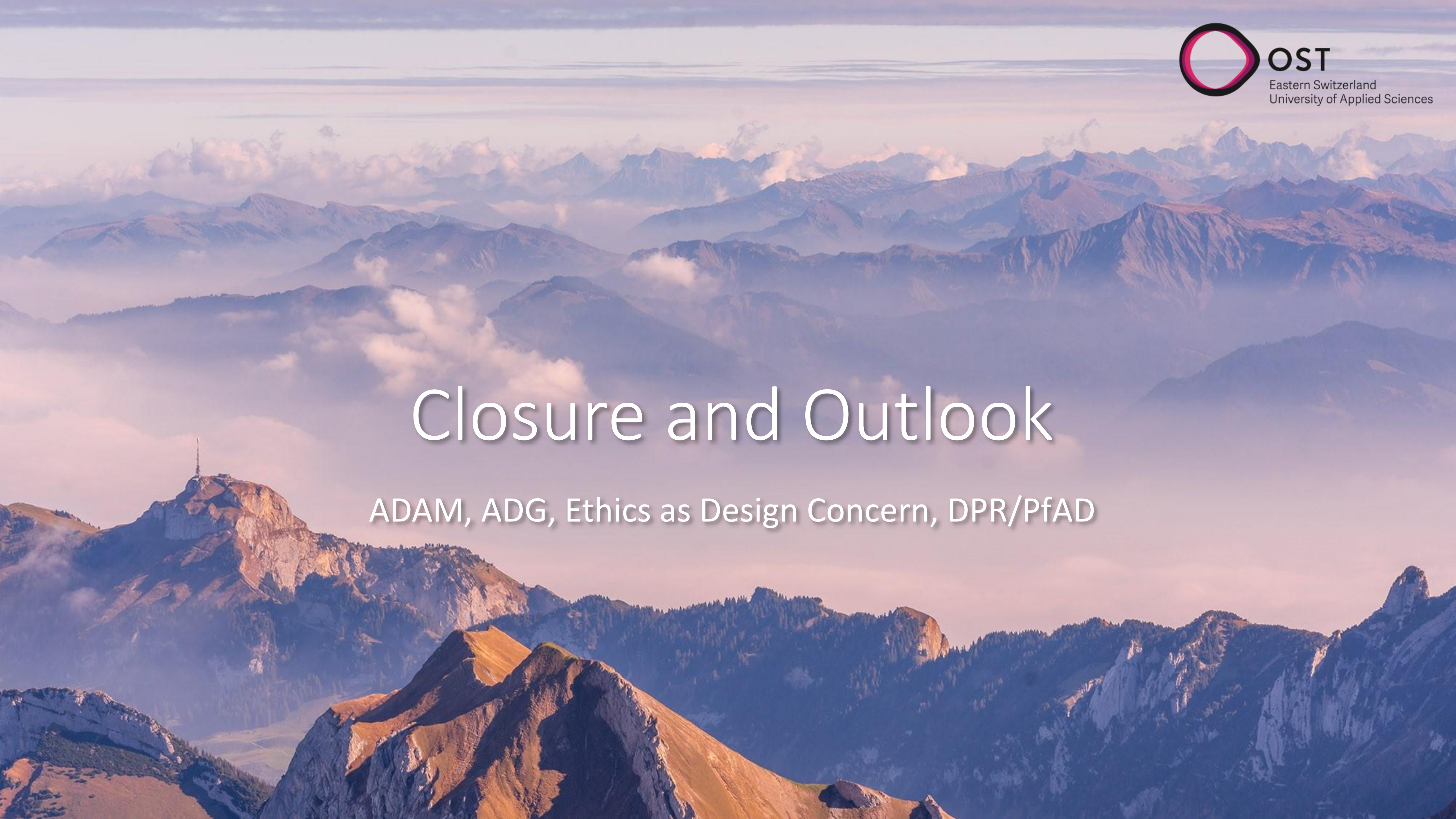
Related Advice (Other Posts)

ASR Test, MADR or Y-Statements, Definition of Done (AD)
ADR creation practices and anti-patterns

Following Up on ADs Made (the R in ecADR)

Goal: avoid (or at least notice and fix) architectural drift

- Project management practice/technique/templates such as RACI
  - Responsible, Accountable, Consulted, Informed (AD metadata!)
- Issue tracking, technical backlog (aka managed issue list)
  - A and T in SMART: agreed upon, time bound
- Software development business-as-usual ("Construction" in UP)
  - With architects as team members or coaches/consultants

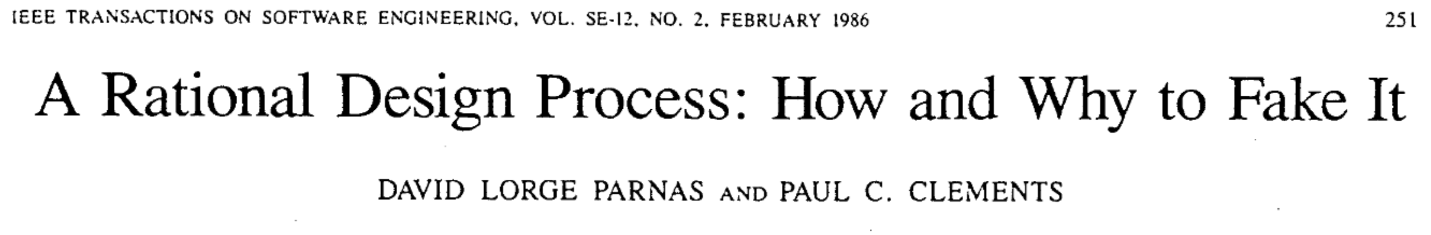


Closure and Outlook

ADAM, ADG, Ethics as Design Concern, DPR/PfAD

Sequential AD making is a – valuable – illusion

https://www.researchgate.net/publication/260649064_A_Rational_Design_Process_How_and_Why_to_Fake_it


IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-12, NO. 2, FEBRUARY 1986 251

A Rational Design Process: How and Why to Fake It

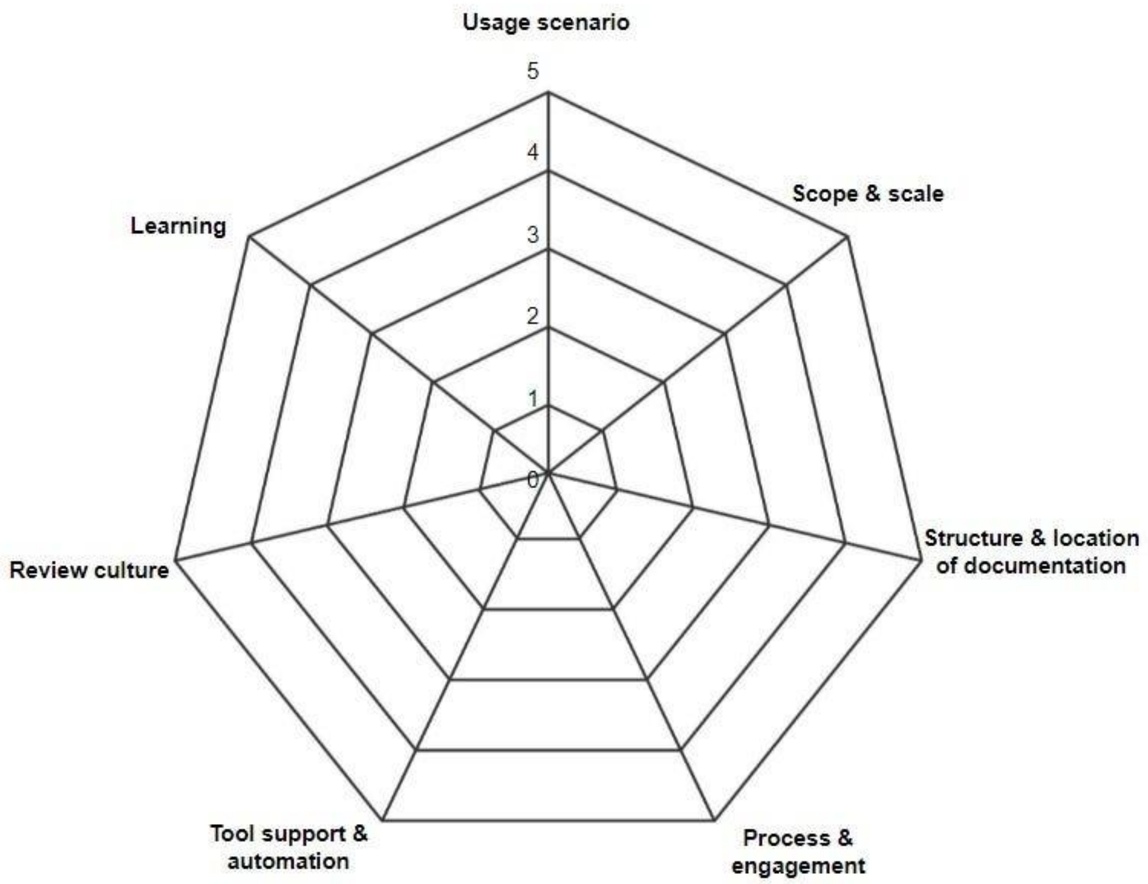
DAVID LORGE PARNAS AND PAUL C. CLEMENTS

- Seven reasons why "we will never see a software project that proceeds in the 'rational' way":
  - Uncertainty, learn as you go, complexity, change, human errors, bias, context
- Five reasons why "we can still follow it as closely as possible and we can write the documentation that we would have produced if we had followed the ideal process":
  - Guidance, learn while you try, standardization, progress tracking, reviewability

*compare with shuhari

Towards an adoption model for AD management (joint work with Mohsen Anvari)

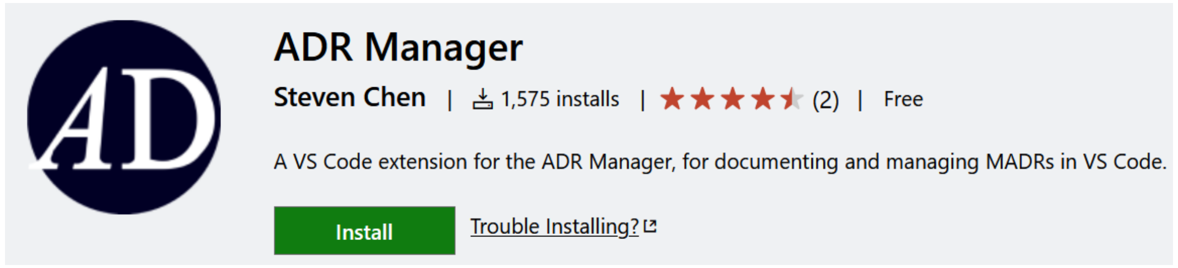
- Five adoption levels:
  1. Undefined and unconscious
  2. Ad-hoc and unstructured
  3. Encouraged and supported
  4. Systematic, selective and diligent
  5. Optimized and rigorous
- No need to go up to 5 for everybody*
  - Context matters (again)



https://medium.com/olzzio/an-adoption-model-for-architectural-decision-making-and-capturing-1399ab81d802

Only light or general-purpose tools needed

- 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


ADR Manager

Steven Chen | 1,575 installs | ★★★★☆ (2) | Free

A VS Code extension for the ADR Manager, for documenting and managing MADRs in VS Code.

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

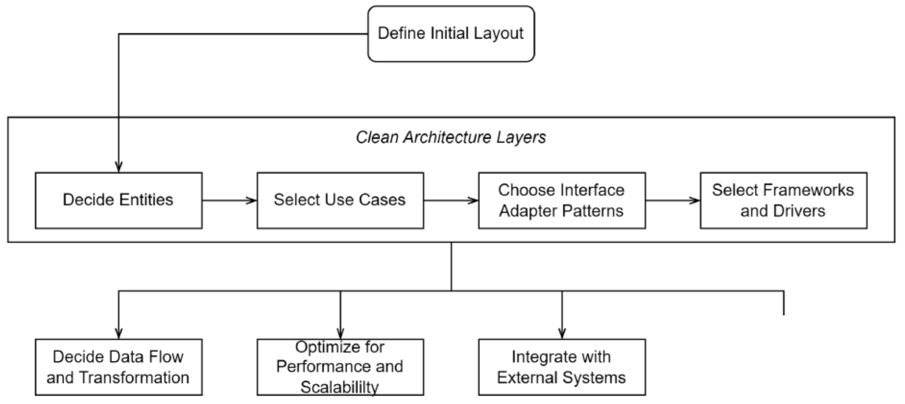
https://adr.github.io/madr/tooling.html

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

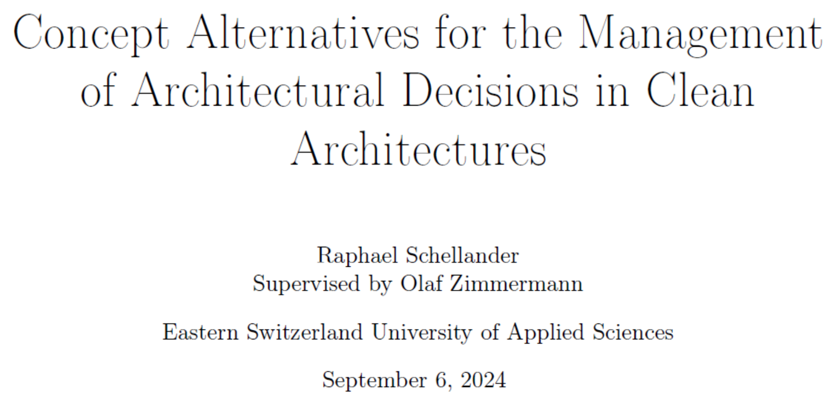
Vision: AD Guidance (ADG)

- From past tense to future tense
- R+D at OST:
  - ADG Tool
  - Clean Decision Handbook
- Enterprise architecture opportunity!
  - TOGAF, SAFe?

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



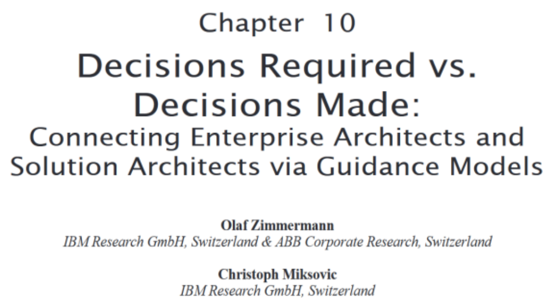
Define Initial Layout
Clean Architecture Layers
Decide Entities
Select Use Cases
Choose Interface Adapter Patterns
Select Frameworks and Drivers
Decide Data Flow and Transformation
Optimize for Performance and Scalability
Integrate with External Systems


Concept Alternatives for the Management of Architectural Decisions in Clean Architectures

Raphael Schellander
Supervised by Olaf Zimmermann

Eastern Switzerland University of Applied Sciences

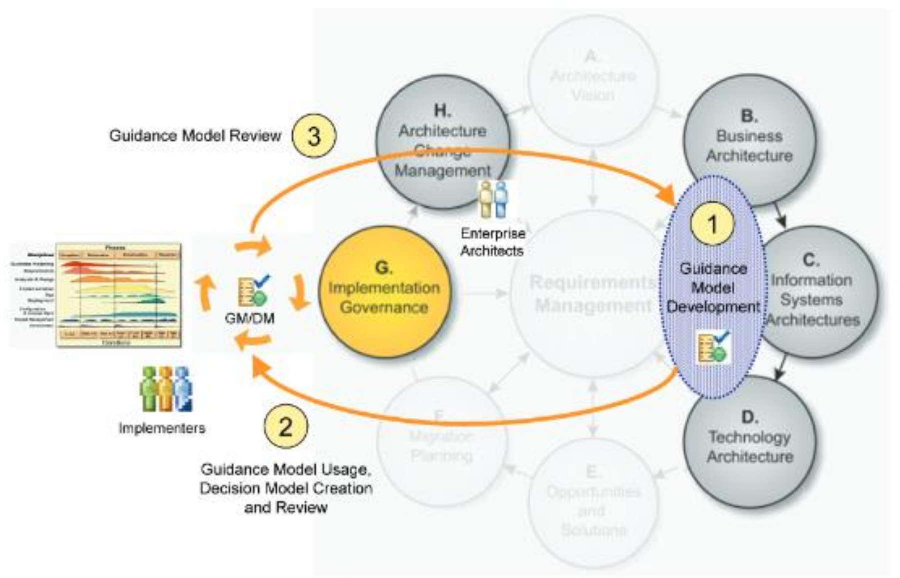
September 6, 2024


Chapter 10

Decisions Required vs. Decisions Made:
Connecting Enterprise Architects and Solution Architects via Guidance Models

Olaf Zimmermann
IBM Research GmbH, Switzerland & ABB Corporate Research, Switzerland

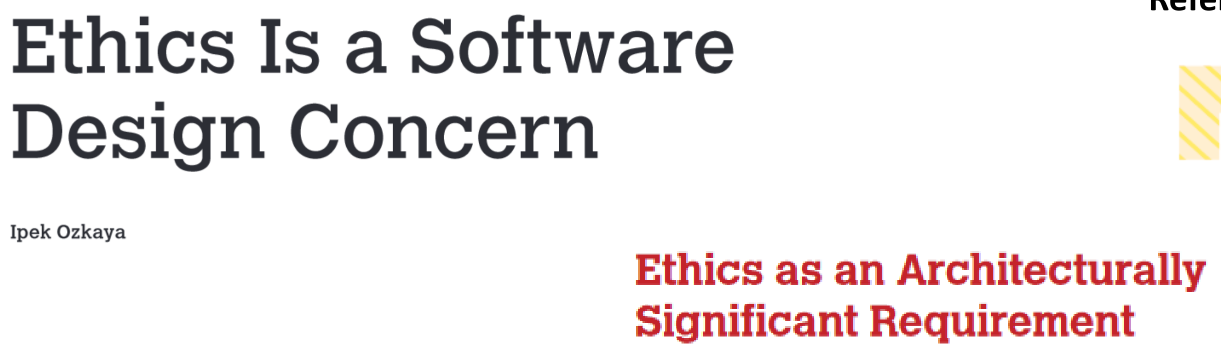
Christoph Miksovic
IBM Research GmbH, Switzerland



Guidance Model Review
3
H. Architecture Change Management
A. Architecture Vision
B. Business Architecture
Enterprise Architects
1
Guidance Model Development
C. Information Systems Architectures
G. Implementation Governance
Requirements Management
GM/DM
Implementers
2
Guidance Model Usage, Decision Model Creation and Review
F. Migration Planning
E. Opportunities and Solutions
D. Technology Architecture

Values… there are way more than $ and UX

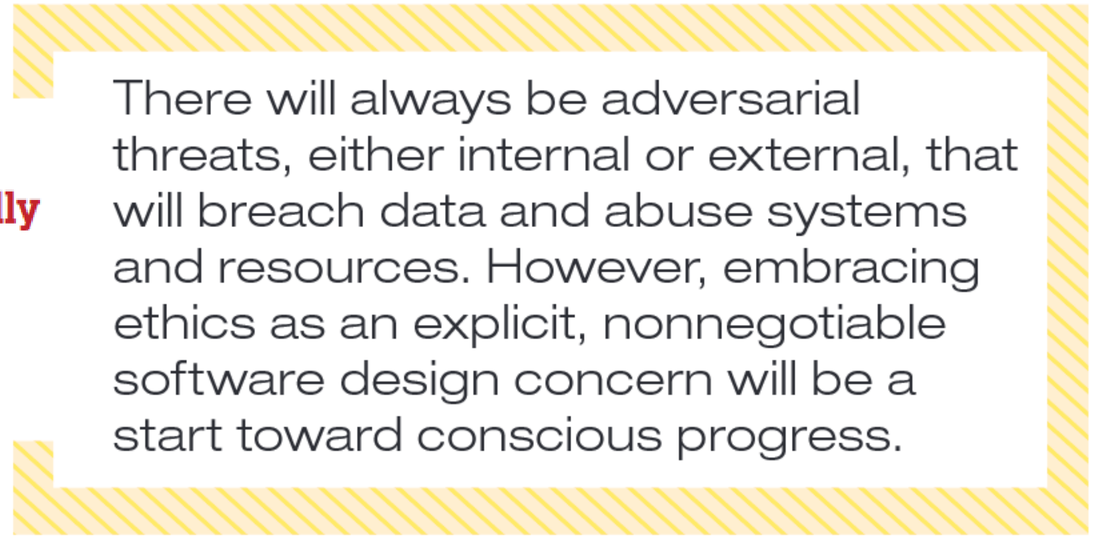
Reference: https://ieeexplore.ieee.org/ielx7/52/8693065/08693077.pdf



Ethics Is a Software Design Concern

Ipek Ozkaya

Ethics as an Architecturally Significant Requirement

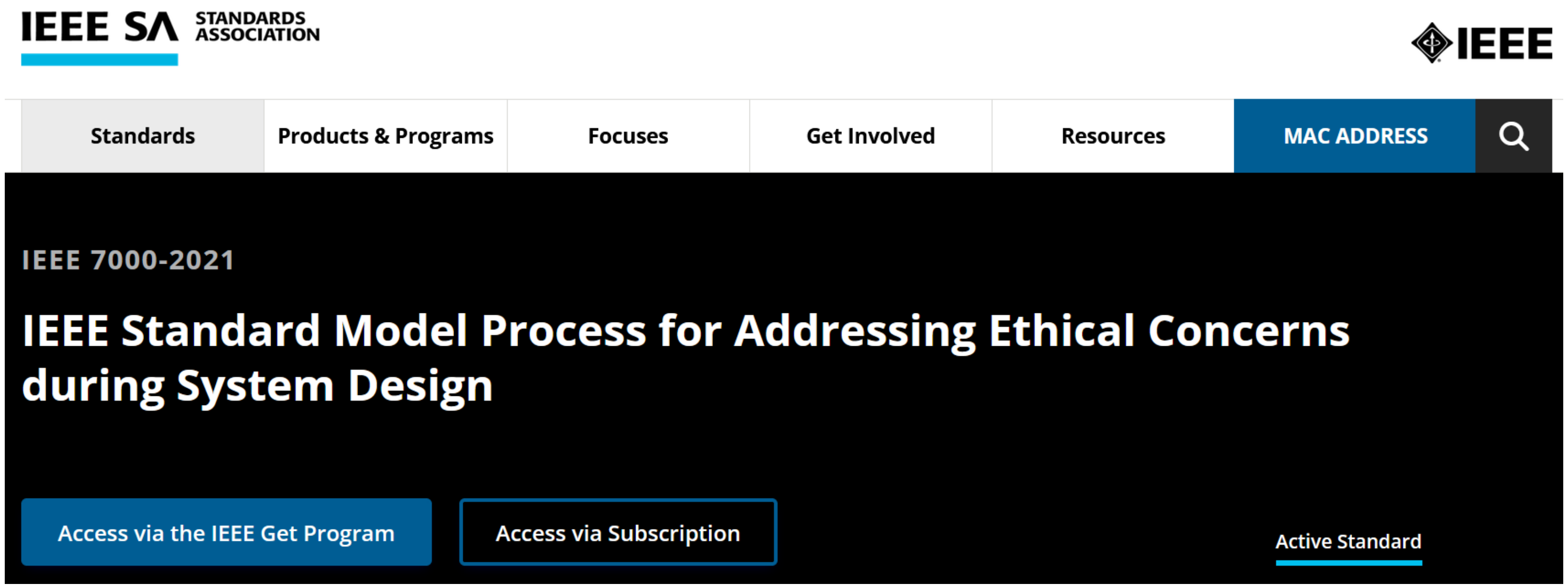


There will always be adversarial threats, either internal or external, that will breach data and abuse systems and resources. However, embracing ethics as an explicit, nonnegotiable software design concern will be a start toward conscious progress.

- Codes of Conduct (ACM, IEEE, …)
  - Related: ProactiveCARE, SoDIS
- Method engineering at OST (open source):
  - VDAD process, ESE practices (feedback welcome!)

Ethical Values yield non-functional requirements

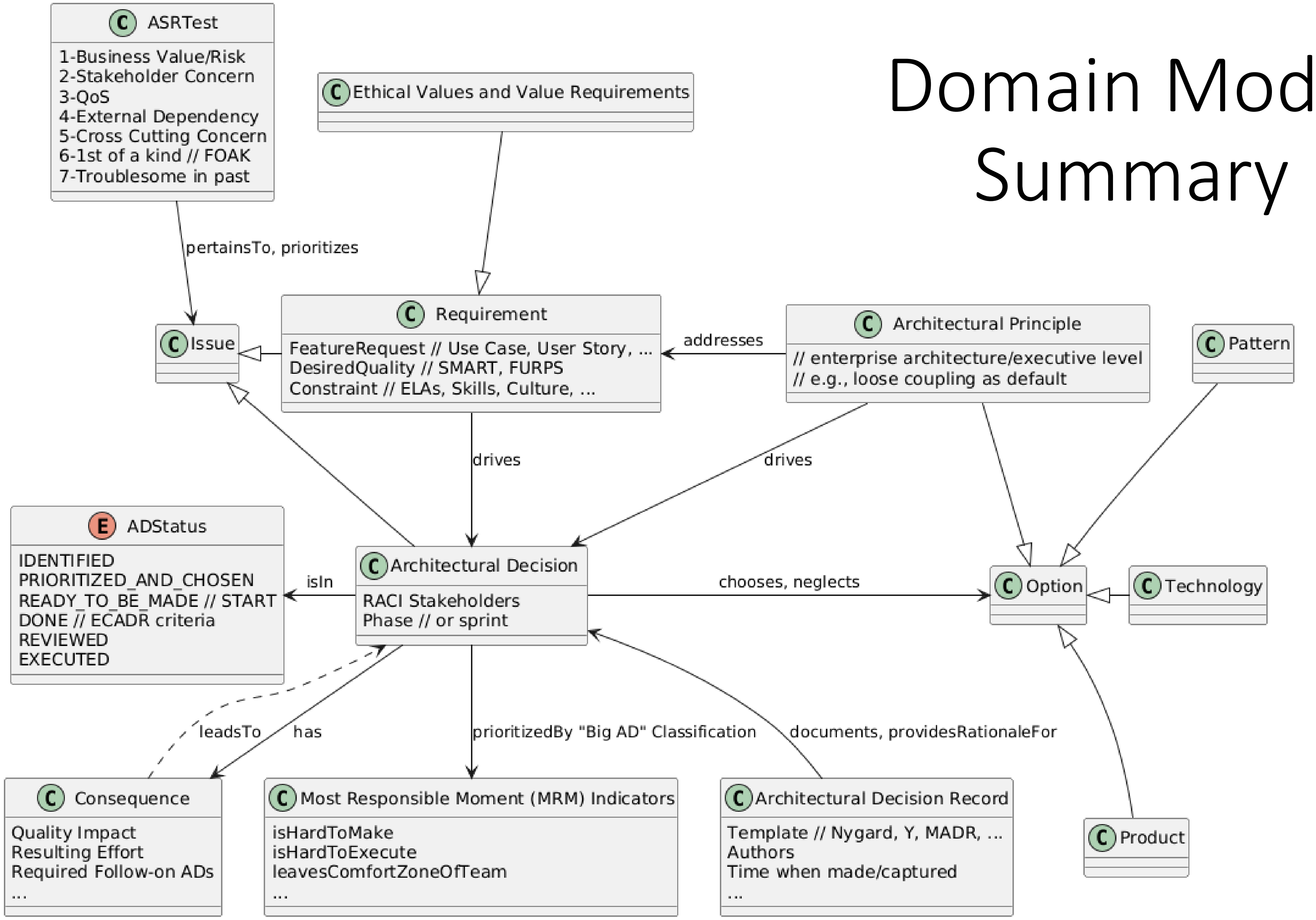
- Ethical values complement business and end user value (and sometimes cause conflicts)
- Impossible to norm a single set of ethics
- Risk-based approach via set of processes



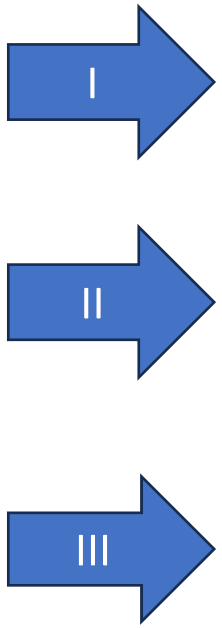
IEEE 7000-2021

IEEE Standard Model Process for Addressing Ethical Concerns during System Design

Domain Model Summary

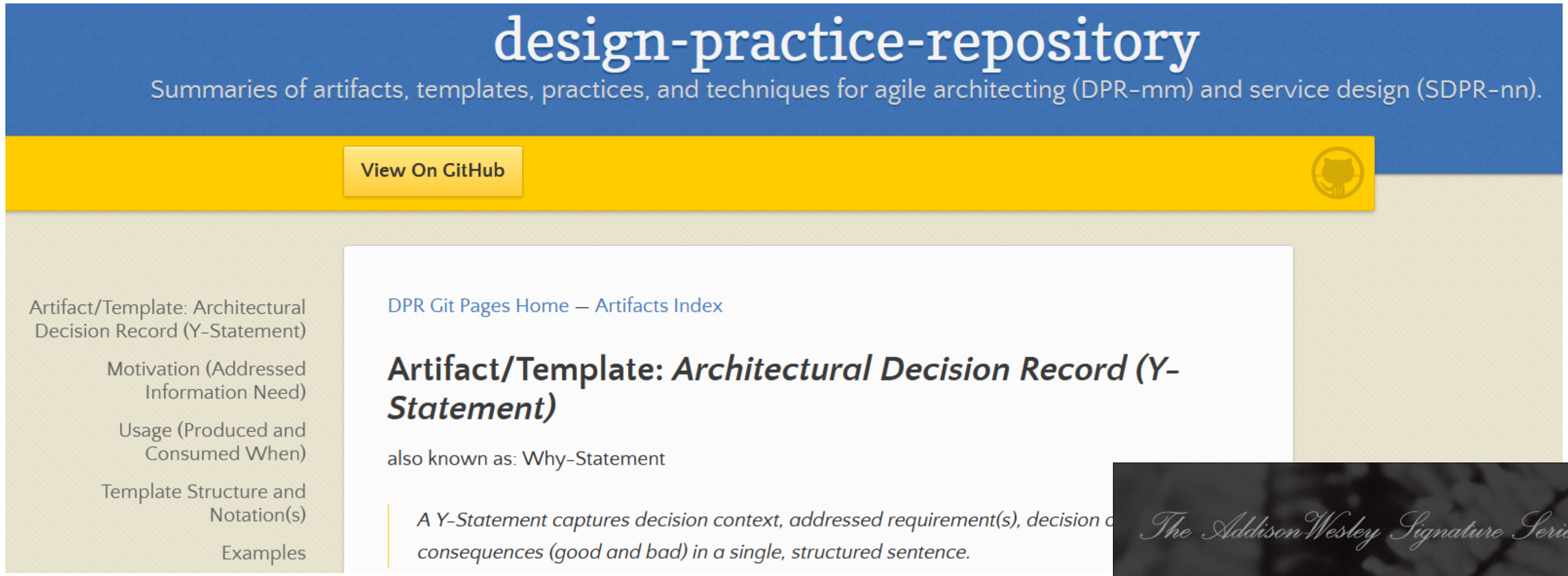


Summary: Three Steps and Practices for Them



- Step I: Identification and Prioritization
  - Identify candidate ADs comprehensively (not perfectly/tediously)
  - Prioritize and select ADs ready to be made carefully (not opportunistically)
- Step II: Making and Capturing
  - Decide consciously and adequately
  - Document efficiently and reproducibly
- Step III: Execution and Review
  - Execute pragmatically and diligently (“agile with discipline”)
  - Enforce and review constructively (not authoritatively)

Books, Open Repos


design-practice-repository

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

View On GitHub

Artifact/Template: Architectural Decision Record (Y-Statement)

Motivation (Addressed Information Need)

Usage (Produced and Consumed When)

Template Structure and Notation(s)

Examples

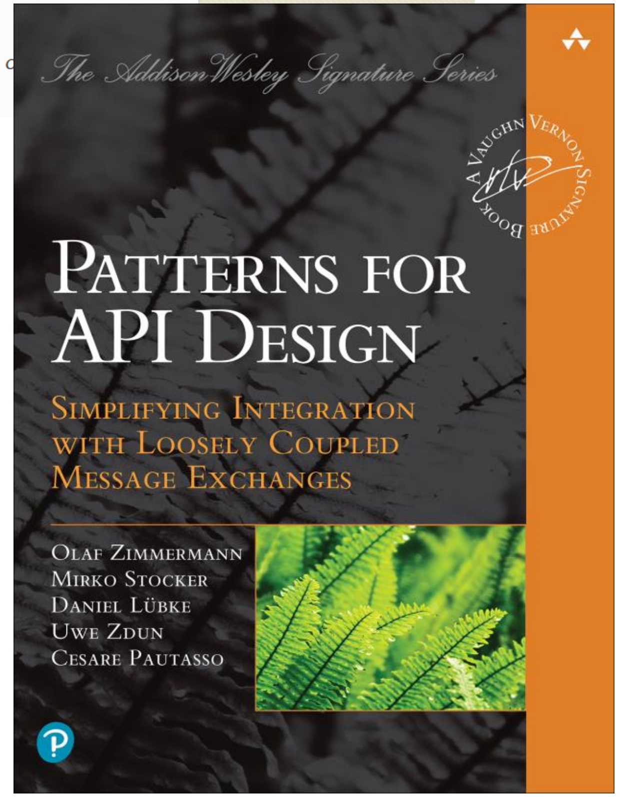
DPR Git Pages Home — Artifacts Index

Artifact/Template: *Architectural Decision Record (Y-Statement)*

also known as: Why-Statement

A Y-Statement captures decision context, addressed requirement(s), decision c

consequences (good and bad) in a single, structured sentence.


The Addison-Wesley Signature Series
A Vaughn Vernon Signature Book
Patterns for API Design
Simplifying Integration with Loosely Coupled Message Exchanges
Olaf Zimmermann
Mirko Stocker
Daniel Lübke
Uwe Zdun
Cesare Pautasso

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

Thank You & Keep in Touch

- I hope you find a few ideas to take away from this presentation – ASR Test, MRM, DoR, Y-Statements, DoD would be my picks
- I will be happy to answer questions and discuss ADs – now, after the talk, after the conference!

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