



University of St.Gallen

Institute of Computer Science

Architectural Decisions (ADs) as a Management Tool: Quality-Driven, Cost-Oriented – Artificially Intelligent?

OODACH Aachen, Germany

September 17, 2026

Dr. Olaf Zimmermann

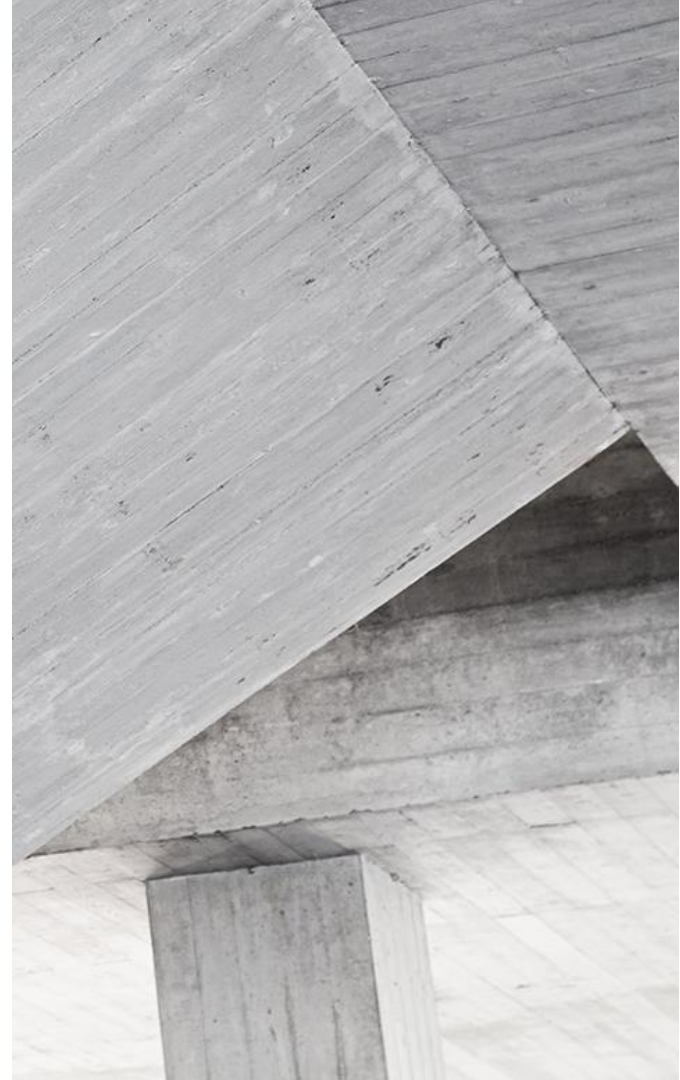
olaf.zimmermann@unisg.ch

From insight to impact.

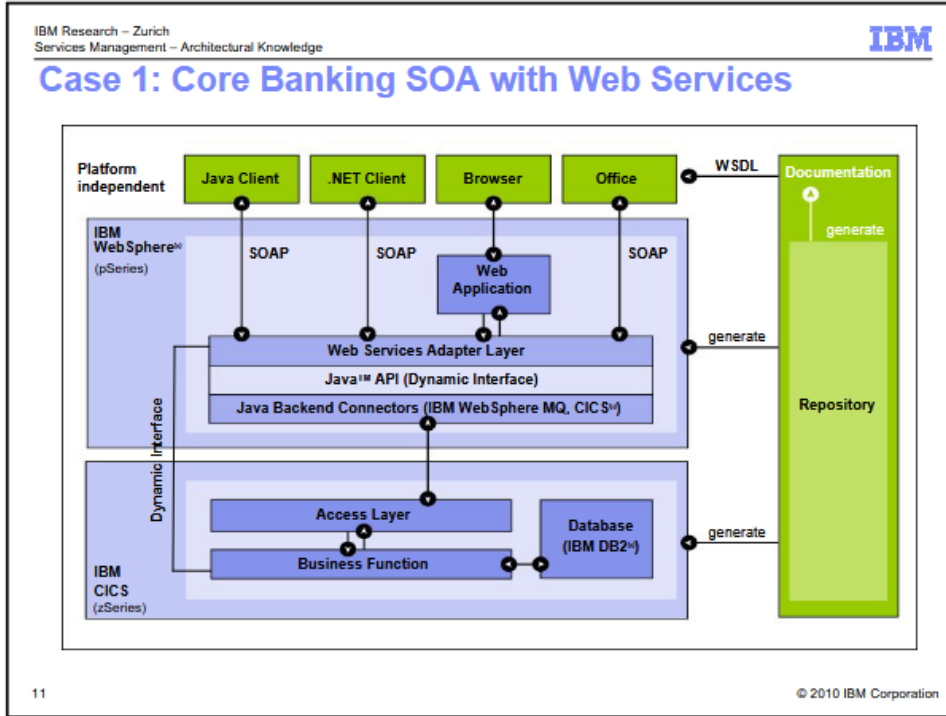


Swiss National
Science Foundation

© 2025–2028 TaSSAreCt Project



A (brief) trip to the past...



Reference: Zimmermann et al, „Second Generation Web Services-Oriented Architecture in Production in the Finance Industry“, OOPSLA 2004 PR, ACM Press

OOPSLA'04
VANCOUVER

Object-Oriented Programming, Systems, Languages, and Applications

Home Program Housing & Transportation Registration Submissions Wiki Maps

> [Technical Program](#) > [Practitioner Reports](#)

: [Thursday](#)

Building Distributed Systems

Program

Technical Program
Invited Speakers
Technical Papers
Onward!
Panels
Practitioner Reports
Tutorials
Workshops
DesignFest
Educators' Symposium

Meeting Rooms 11-12
Thursday, 8:30, 1 hour 30 minutes

7 · 8 · 9 · 10 · 11 · 12 · 13 · 14 · 15 · 16 · 17 · 18 · 19 · 20 · 21

Session Chair: Lougie Anderson, Sabrix, Inc.

Web Services-Oriented Architecture in Production in the Finance Industry
Olaf Zimmermann
Thursday, 8:30, 30 minutes, Meeting Rooms 11-12

One of my first papers was an
OOPSLA practitioner report

[...]

PhD Thesis on decision-centric design
method for service-oriented
computing

[...]

<https://www.ozimmer.ch/blog>,
“Design Practice Reference” eBook,
“[Patterns of API Design](#)”, Addison-
Wesley Professional

Architectural Decision (AD) Recording 101

What is an Architecture Decision (AD)?

Several definitions exist, which agree on the essentials (goal, content, relationships):

- The homepage of the [ADR GitHub organization](https://github.com/adr/adr.github.io) states: "An *Architectural Decision (AD)* is a justified design choice that addresses a functional or non-functional requirement that is architecturally significant".
- Grady Booch emphasizes the cost aspect (source: a recent edition of the "Dear Architects" newsletter): "Architecture represents the significant design decisions that shape a system, where significant is measured by cost of change."
- An InfoQ/IEEE [article from 2011](#) elaborates on these two themes, cost and significance:

<https://github.com/adr/adr.github.io>

*Architectural decisions capture **key design issues** and the **rationale behind chosen solutions** concerning a software-intensive system as a whole or one or more of its core components and connectors.*

*The AD outcome influences the system's nonfunctional characteristics including its **software quality attributes**.*

IBM ARC-100 Template (1998)

More than 10 table entries,
from context information to
decision outcome and impact
(implications)

Filled-out templates shipped
with IBM reference
architectures and inspired
Tyree/Akerman to create a
template for their company,
later published in IEEE Software

IBM Research – Zurich
Services Management – Architectural Knowledge



**The current state of the art and the practice
makes it difficult to reuse issues decisions during architecture design.**

Subject Area		Topic	
Architectural Decision		AD ID	
Decision	"We decided for the Broker pattern [POSA] ..."		
Issue or Problem	... to integrate front end and backend...		
Assumptions			
Motivation			
Alternatives			
Justification	...because we gained positive experience with it on many similar projects."		
Implications			
Derived Requirements			
Related Decisions			

Problem 1:
Template designed to capture decisions made

Unwelcome documentation task, out of design context

Problem 2:
Text documents used

Manual work, limited scalability

Source: UMF Work Product Description for Architectural Decisions, © IBM, 1998

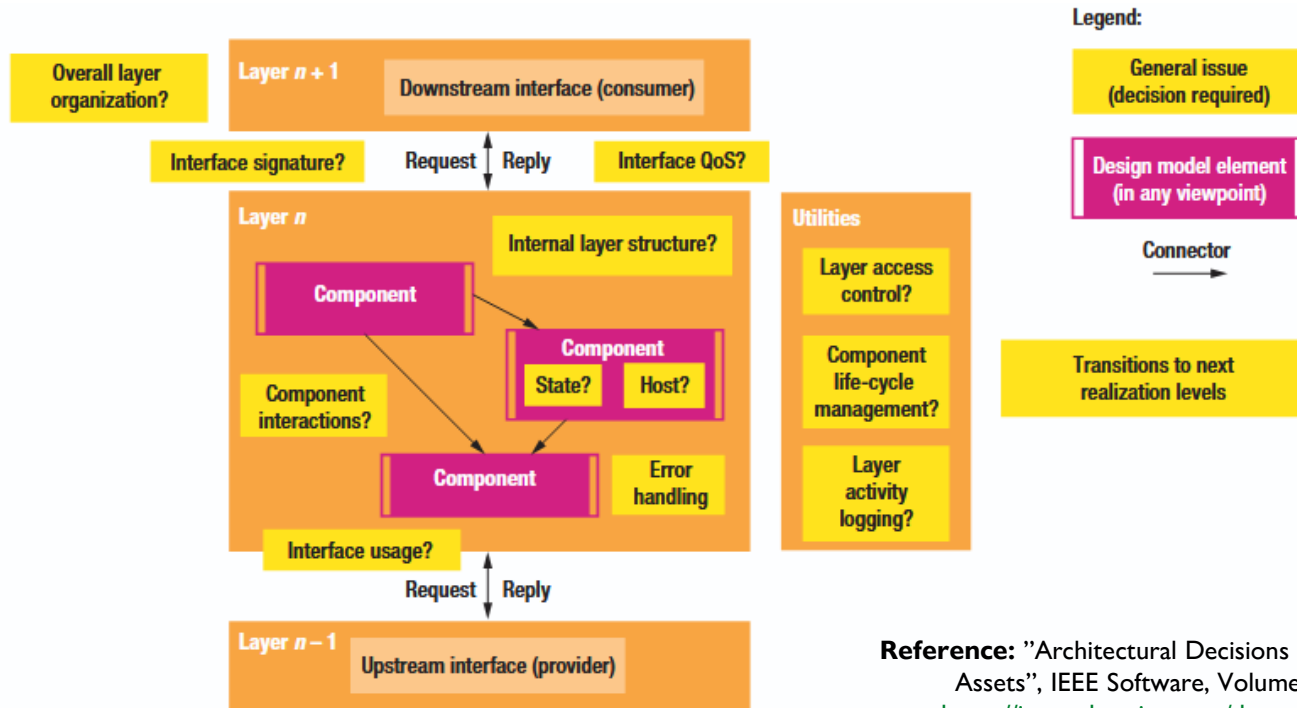
Consequence: Little reuse – no catalog of recurring design issues exists today (SOA or other style)

Design work always conducted from scratch and in isolation – time consuming, error prone

34
© 2010 IBM Corporation

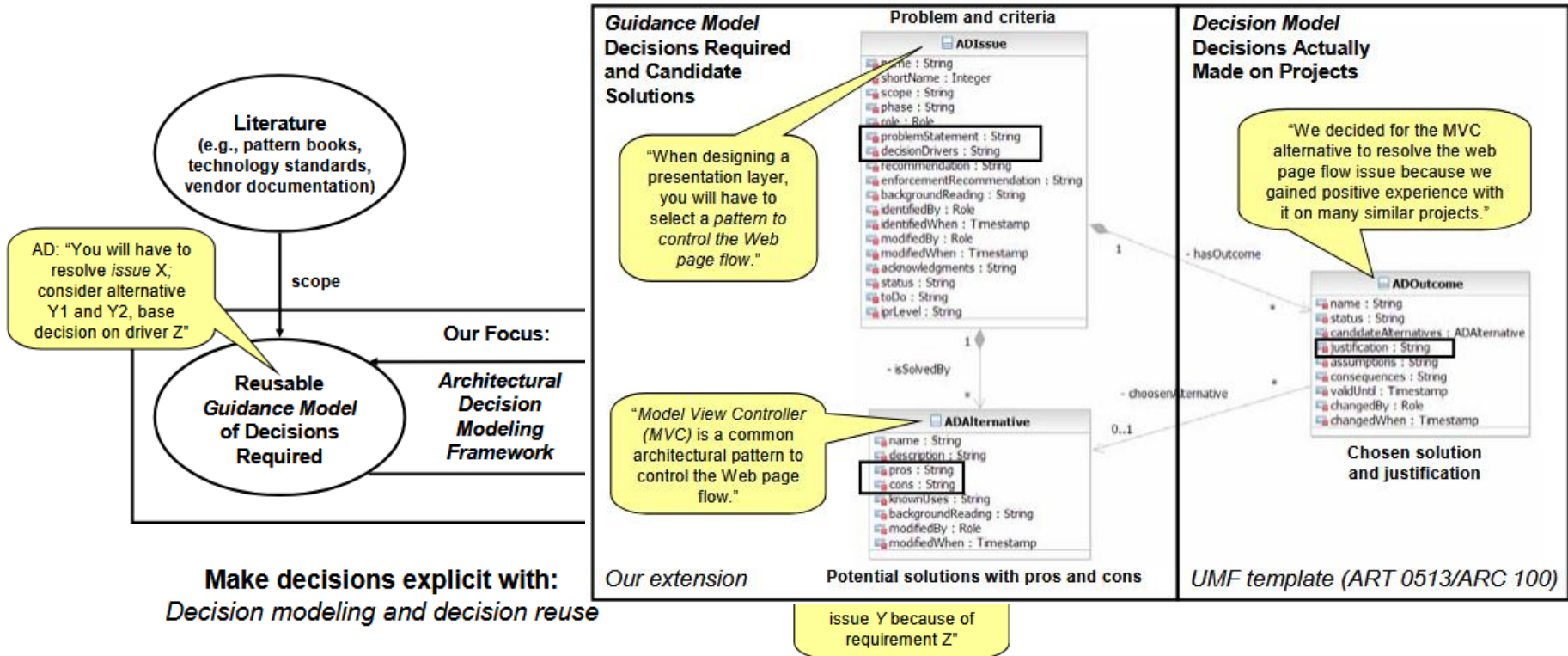
Reference: SEI SATURN 2020, <https://soadecisions.org/download/ZRL-ADModelingTutorialMay2010v101.pdf>

Recurring Decisions: Problem and Alternatives, not Outcome

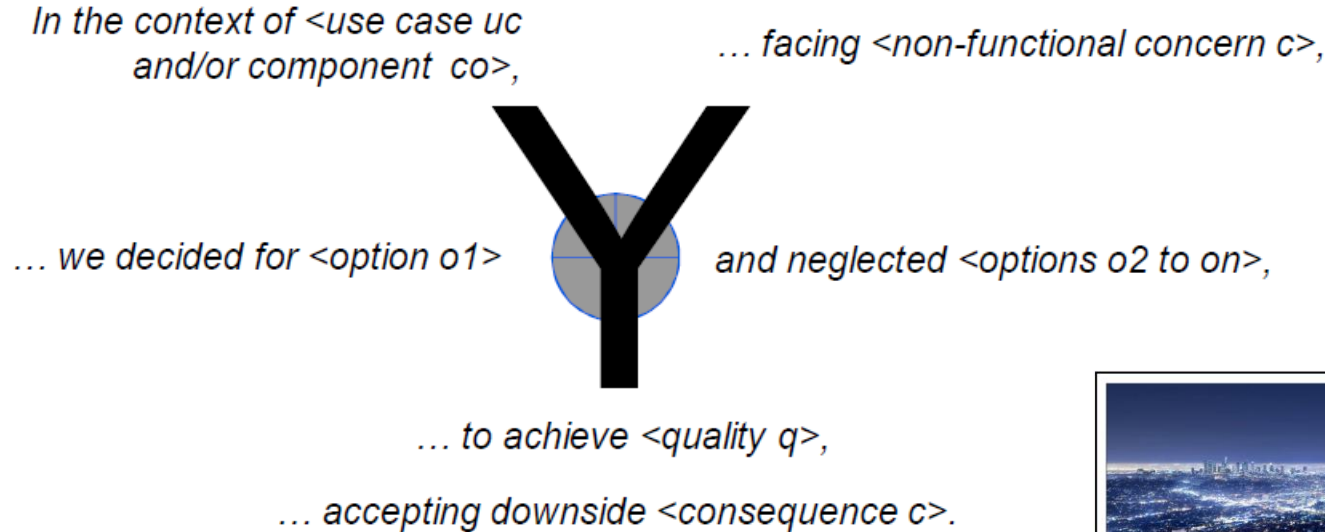


Reference: "Architectural Decisions as Reusable Design Assets", IEEE Software, Volume: 28 Issue: 1
<https://ieeexplore.ieee.org/document/5672520>

SOAD Project (2006 – 2009)



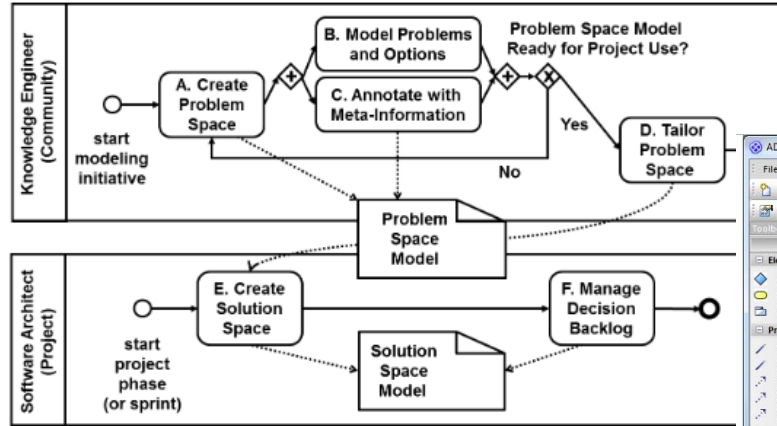
Y-Statement Template (ABB Projects, SEI SATURN 2012)



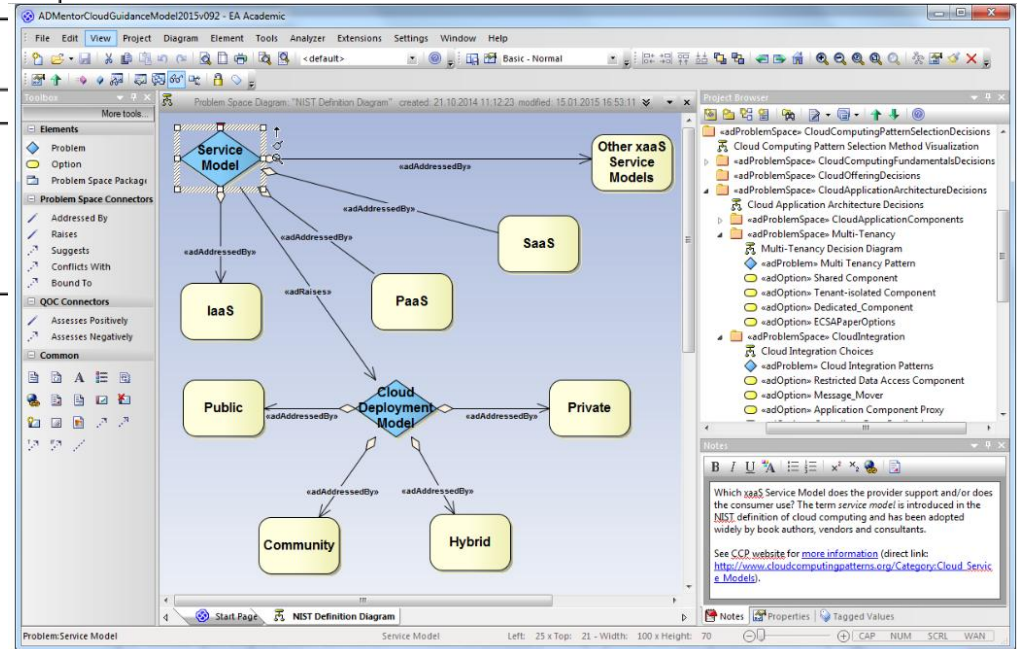
<https://medium.com/olzzio/y-statements-10eb07b5a177>



AD Mentor (ABB Research Grant 2014)

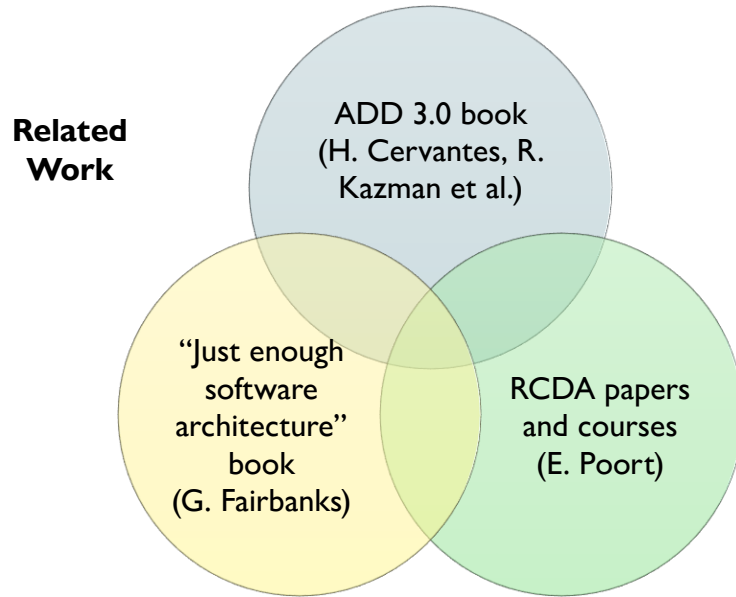


- Plugin/extension for EA, the UML tool from Sparx



- Question, Option, Criteria (QOC) diagrams
- Y-Statements and other AD capturing templates
- Dependencies modelled

Related Work in Architectural Decisions (ADs): Quality-Driven, Cost-Oriented, Prioritized by Risk



Quality attributes as
primary (but not only)
decision drivers

Risk and cost to prioritize
design work

Decision stories

Tradeoffs

“If in doubt, leave it out”

Architectural Decisions (ADs) as a Management Tool: Artificially Intelligent?

AI Assistance for Architectural Decision Making: Domain-Driven Context and Prompt Engineering

Olaf Zimmermann¹ and Ronny Seiger²[0000-0003-1675-2592]

¹ University of St.Gallen, St.Gallen, Switzerland, olaf.zimmermann@unisg.ch

² RWTH Aachen University, Aachen, Germany, ronny.seiger@rwth-aachen.de

Abstract. Generative Artificial Intelligence (AI) is receiving a lot of focus in research and practice at present; positions vary from euphoria to “just another tool” to skepticism. Software architecture analysis, synthesis, and evaluation offer many opportunities for AI, e.g., support for knowledge-intensive tasks such as making and recording architectural decisions. It is not yet fully understood how architects can be assisted by AI effectively and efficiently: AI services have to be supervised via input specifications and output validation, requiring architecting skills and application domain expertise. In this paper, we identify architecturally significant requirements to drive the design of a future ecosystem of AI-Assisted Decision Assistance tools (AID). We apply Domain-Driven Design (DDD) to establish a logical component architecture for AID that leverages model-driven knowledge management concepts to accelerate context and prompt/skills engineering and to guard response processing.

Reference: Zimmermann, O., Seiger, R.,
«AI Assistance for Architectural Decision Making:
Domain-Driven Context and Prompt Engineering»,
Proc. of IDEA-ARCH Workshop at ECSA , 2026

Overarching Research Questions

1. *How can AI support human architects during architectural analysis, synthesis, and evaluation so that productivity increases and risks decrease?*
2. *How should generative and agentic AI be integrated into human activities during the preparation, making, recording and review of architectural decisions?*

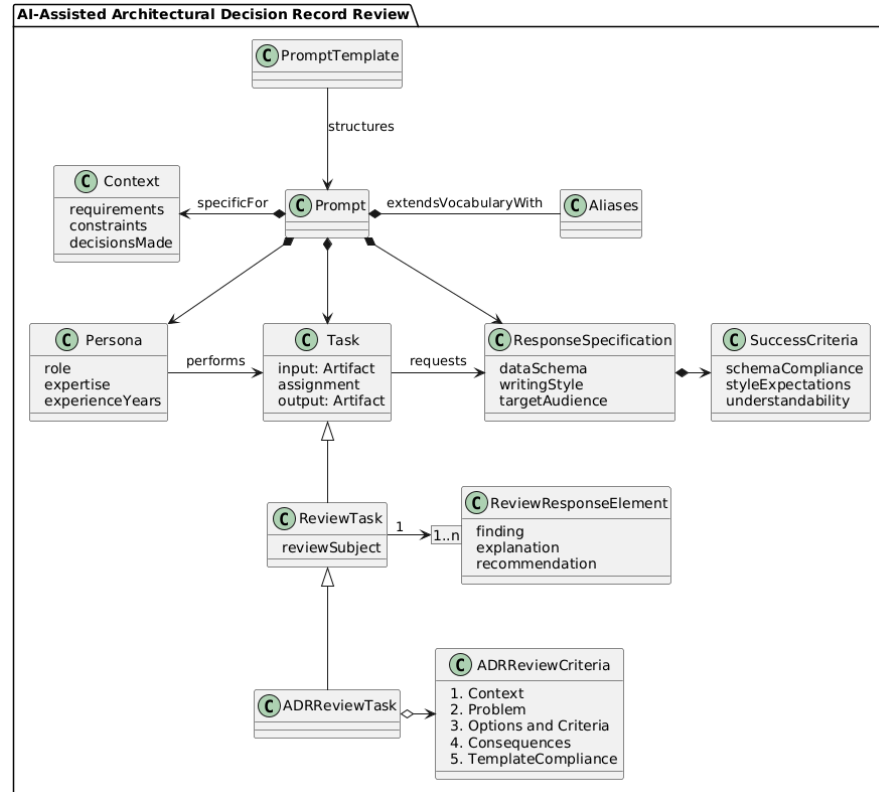
Project Goal

in Design Science Methodology (DSM) Template by R. Wieringa

- *Improve* the requirements engineering and architecture design work in CPSoS domain/application genre
- *by* supporting practicing software and system architects with a reusable, guiding Requirements and Decision Management Model (ReDeMo), tool supported and integrated with AI assistants,
- *such that* it comprises recurring issues with commented/curated options and decision-making criteria,
- *in order to* make better decisions (better as in: more informed, more involved; less risky, less costly, with healthier/more sustainable pos. consequences, partially automatable output)

Conceptual Model of ADR Review Prompt

- Template vs. instance
- Persona and other patterns
- ADR quality criteria checklist
- Finding-explanation-recommendation triples as response format
- Success criteria



ADR Review Prompt Example

<https://zenodo.org/records/20830043/files/ADR-PromptingInstance-ApacheJamesADR37Included.txt?download=1>

Please review an Architectural Decision Record (ADR) with respect to the following 5 criteria:

1. Quality of context section: quality attributes mentioned, scope of decision clear (which system part?), status quo clear
2. Solved problem clearly stated
3. Presence of options with pros and cons, serving as decision criteria; chosen option is able to satisfy stated requirements (resolve design forces)
4. Quality of consequences section: both good and bad consequences stated, multiple stakeholders and their concerns mentioned, forces from context picked up
5. ADR template conformance, here: Nygard template: presence of template sections, content fits the section description

Sample AD: Mailbox Component of Apache James (Java Apache Mail Enterprise Server)

<https://github.com/apache/james-project/blob/master/src/adr/0037-eventbus.md>

How about:

- *Options/alternatives?*
- *Decision drivers (“-ilities”)?*
- *Good and bad consequences?*



Status

Accepted (lazy consensus) & implemented

Context

Many features rely on behaviors triggered by interactions with the mailbox API main interfaces (`RightManager` , `MailboxManager` , `MessageManager` , `MessageIdManager`). We need to provide a convenient extension mechanism for organizing the execution of these behaviours, provide retries and advanced error handling.

Also, protocols enable notifications upon mailbox modifications. This is for instance the case for `RFC-2177 IMAP IDLE` , leveraged for `RFC-3501 IMAP unsolicited notifications` when selecting a Mailbox, as well as maintaining the `IMAP Message Sequence Number` <-> `Unique Identifier` MSN <-> `UID` mapping. Changes happening for a specific entity (mailbox) need to be propagated to the relevant listeners.

Decision

James mailbox component, a core component of James handling the storage of mails and mailboxes, should use an event driven architecture.

It means every meaningful action on mailboxes or messages triggers an event for any component to react to that event.

`MailboxListener` allows executing actions upon mailbox events. They could be used for a wide variety of purposes, like enriching mailbox managers features or enabling user notifications upon mailboxes operations performed by other devices via other protocol sessions.

Interactions happen via the managers (`RightManager` , `MailboxManager` , `MessageManager` , `MessageIdManager`) which emit an event on the `EventBus` , which will ensure the relevant `MailboxListener` s will be executed at least once.

`MailboxListener` can be registered in a work queue fashion on the `EventBus` . Each work queue corresponds to a given `MailboxListener` class with the same configuration, identified by their group. Each event is executed at least once within a James cluster, errors are retried with an exponential back-off delay. If the execution keeps failing, the event is stored in `DeadLetter` for later reprocessing, triggered via WebAdmin.

Guice products enable the registration of additional mailbox listeners. A user can furthermore define its own mailboxListeners via the use of `extension-jars` .

`MailboxListener` can also be registered to be executed only on events concerning a specific entity (eg. a mailbox). The `registrationKey` is identifying entities concerned by the event. Upon event emission, the manager will indicate the `registrationKey` this event should be sent to. A mailboxListener will thus only receive the event for the registration key it is registered to, in an at least once fashion.

Consequences

We need to provide an `In VM` implementation of the `EventBus` for single server deployments.

We also need to provide [a distributed event bus implementation](#).

ADR Review: YAML Responses

See [Claude Sonet 4.6 response in data set](#)

Key element: *Finding-Explanation-Recommendation (FER) triples*

- Designed starting from reflection of own review practices, then demonstrated to practitioners
- Output validation DEMO
 - In online tools

```
overall-quality-assessment:  
  score: good or ok or poor  
  explanation-of-score: TODO  
review-of-context-section:  
  - &item11  
    finding: TODO  
    explanation: TODO  
    recommendation: TODO  
  - &item12  
    finding: TODO  
    explanation: TODO  
    recommendation: TODO  
review-of-problem-question:  
  - &item21  
    finding: TODO  
    explanation: TODO  
    recommendation: TODO  
review-of-options-and-criteria:
```

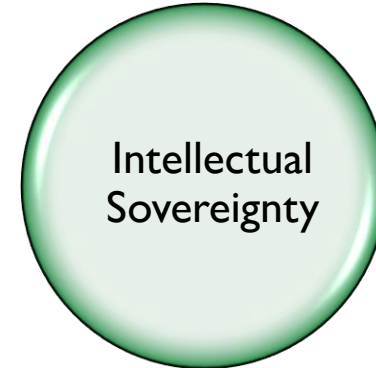
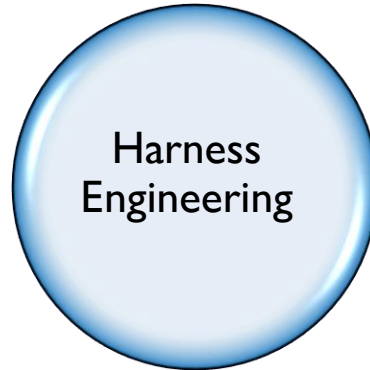
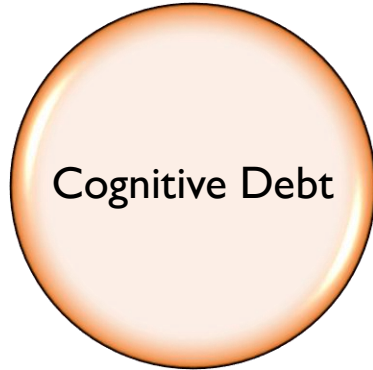
What Will Remain Human in Software Architecture?

Reference: What Will Remain Human in Software Architecture?

A Focus Group Report

U. van Heesch, O. Zimmermann, C. Kohls

Proc. of EuroPLoP 2026 ([preprint at arxiv](#))



Contribution Summary and Outlook

Our vision: *assistive cognitive offloading* of architectural decision work

- Rather than substitutive or disruptive cognitive offloading [[Jose et al. 2026](#)]
- In line with observations and recommendations for AI use by psychologists

Contributions so far (early results):

- A strategic Domain-Driven Design (DDD) for the generative/agentive AI landscape
- Requirements and use-case driven tool architecture (tactic DDD, other patterns)
- ADR review prompt template

Future work (ongoing and planned):

- AIDR tool implementation and evaluation
- LLM as a judge for ADRs and ADR reviews
- Support for other SWA tasks/ADR use cases (design agents? agent orchestration?)
- [FERC](#) process integration, work on argument quality (Wachsmuth taxonomy)

Projects and topics we don't have time for today...

- Markdown ADRs (MADR), YAML ADRs (YADR)
- Design Practice Repository (DPR)
- Interface Refactoring Catalog (IRC)
- Ethical Software Engineering (ESE)
- DSLs
- Agile architecting practices: DoR, DoD, ...
- From business process models to software engineering artifacts and event simulation
- Cloud-native application architecture traits (SUPER-IDEAL)
- ...

Check out my blog... <https://ozimmer.ch>

<https://ozimmer.ch/practices/2026/09/12/ADRMistakes.html>

The Concerned Architect

Why is this blog called "The Concerned Architect"? Software architects¹ (hopefully) address functional and **non-functional** stakeholder concerns.² If the blog name has whetted your appetite, you may want to start exploring it by role:

- (IT architect) *Architectural Decisions — The Making Of* recaps how Architectural Decisions (ADs) have been treated since the 1990s, features a lightweight decision capturing template and gives examples. An AD Definition of Done and [more posts about AD making/capturing](#) are available.
- (API designer) *Getting Started with Microservice API Patterns (MAP)* explains the scope of our pattern language for API design and evolution and points you at starter material. From 2016 to 2022, MAP evolved into an Addison Wesley book: "*Patterns for API Design: Simplifying Integration with Loosely Coupled Message Exchanges*".
- (Technical writer) *How to Write Review-Friendly Articles* discusses what reviewers expect, shares some techniques how to meet these expectations and provides pointers to deeper technical writing advice. The *Authoring* category contains all posts of this kind.

All posts are listed in an [index post](#). The [tags](#) page provides a keyword-based directory. There is an [RSS feed](#) for this blog. Terms and conditions apply.³

New and Noteworthy



Seven Architectural Decision Making Fallacies (and Ways Around Them)

When making Architectural Decisions (ADs), things can go wrong. Biases and misconceptions are common. As a result, answers to "why this design?" questions, captured in...

ZIO
Updated: 27 Oct 2025
Published: 01 Sep 2025



More Resources about AI You Might Want to Review

I came across additional reading resources since I posted "Ten Resources about AI in Software Engineering You Do Not Want to Miss". You might find...

ZIO
Updated: 31 Aug 2025
Published: 28 Aug 2025

Thanks!

Olaf Zimmermann, HSG

firstname.lastname@unisg.ch

with
Ronny Seiger, RWTH



University of St.Gallen

Institute of Computer Science



© 2025–2028 TaSSAreCt Project



University of St.Gallen
Institute of Computer Science
Rosenbergstrasse 30
9000 St.Gallen

ics.unisg.ch