Software Architecture

CS4505, TU Delft, 2024/2025

Arie van Deursen & Diomidis Spinellis

Image credit: Corné Bastiaansen www.rijksvastgoedbedrijf.nl

Delft Fintech Lab

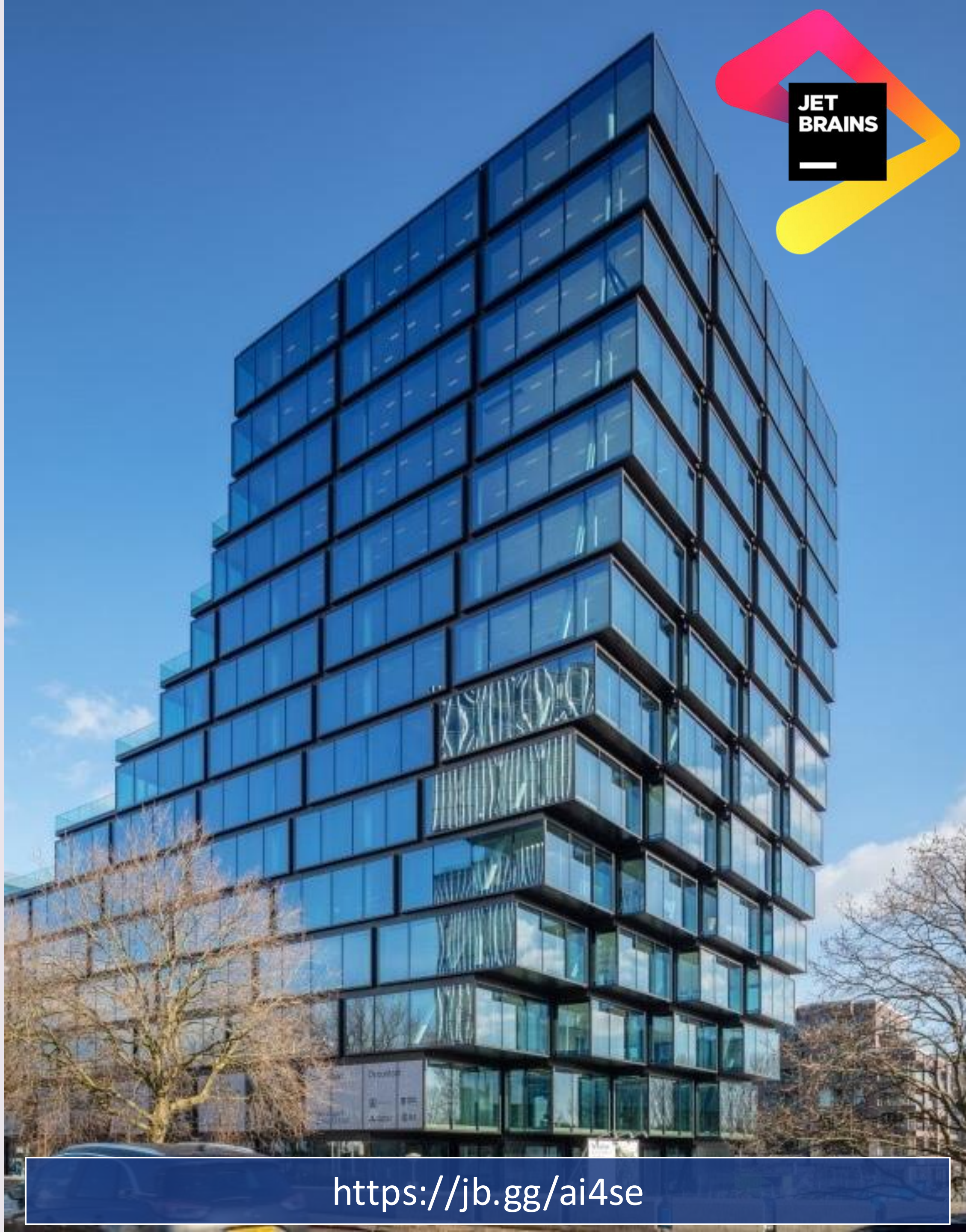
- Research, education, and innovation in Fintech across TU Delft
  - 50 researchers, 25 partner organizations
- ING AI For Fintech Research:
  - 2020-2024, 8 research tracks
  - Explainable AI (counterfactuals)
  - Incident management and AIOps
  - Release planning
  - Search-based testing and repair
- My role: Scientific director

Advisory Council
IT Assessments

- Offer advice to Dutch parliament and cabinet
- Scope: Risky (> €5M) IT projects
- ~20 reports per year
- Office of ~25 people
- Assessment framework
- My role: Council member (0.2 fte)

AI for Software Engineering Lab with JetBrains

- Five year program: 2023-2028
- Five research tracks:
  - Validating (AI) generated code
  - Optimizing code language models
  - IDE-AI alignment
  - Run time information in the IDE
  - Programming education
- 10 PhD candidates
- My role: Scientific director



https://jb.gg/ai4se

Learning Objectives?

What do you hope / expect to learn
in the TU Delft Software Architecture course?

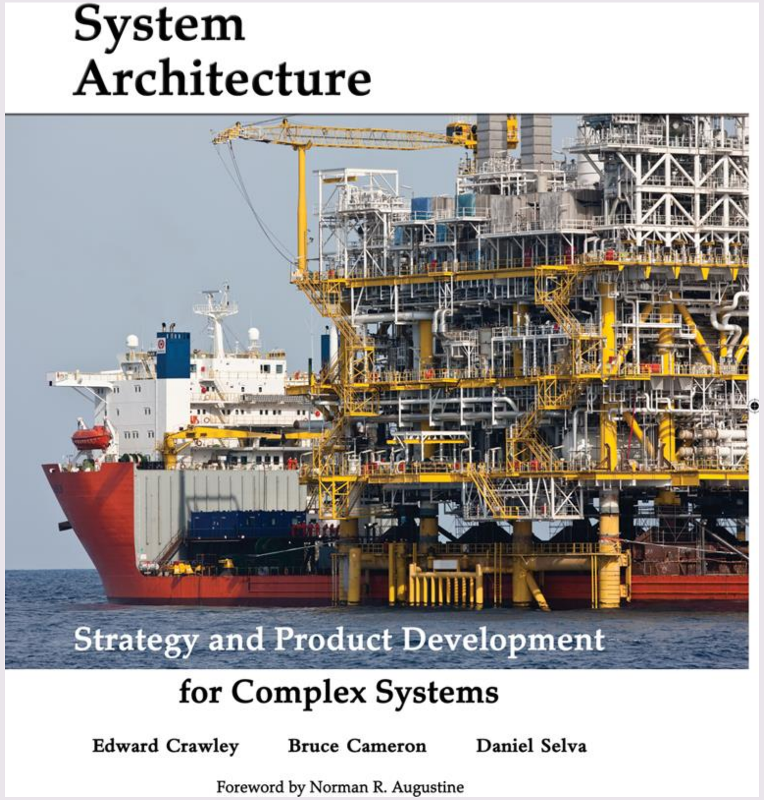
What's your starting point / current knowledge?

What new knowledge would you like to gain?

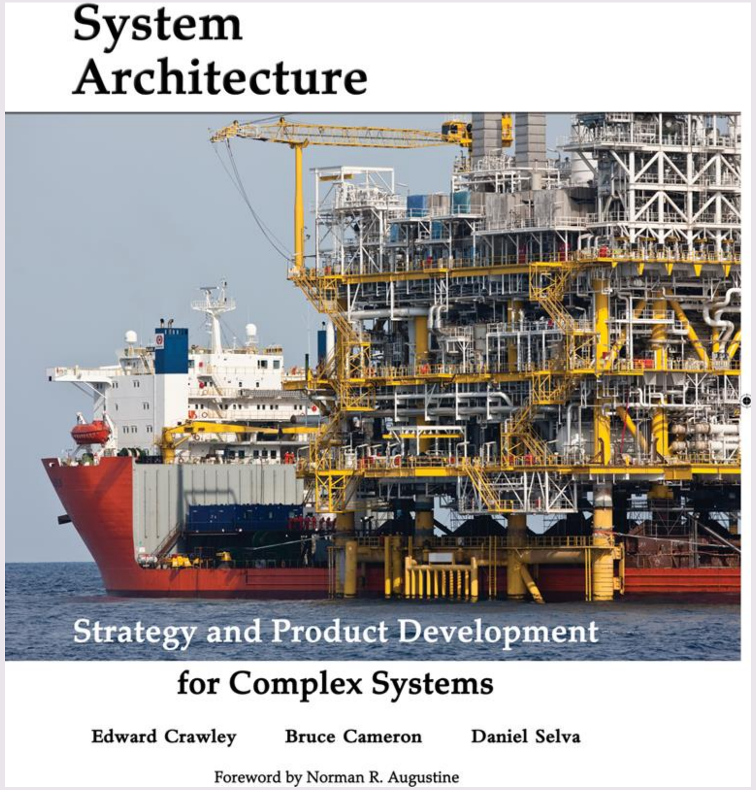
Software Architecture as *Systems* Architecture?

“Enable system architects
to *structure and lead*
the early, conceptual phases
of the system development process,

and to *support* the process
throughout its development, deployment,
operation, and evolution.”



Principles of System Architecture



System Architecture

Strategy and Product Development for Complex Systems

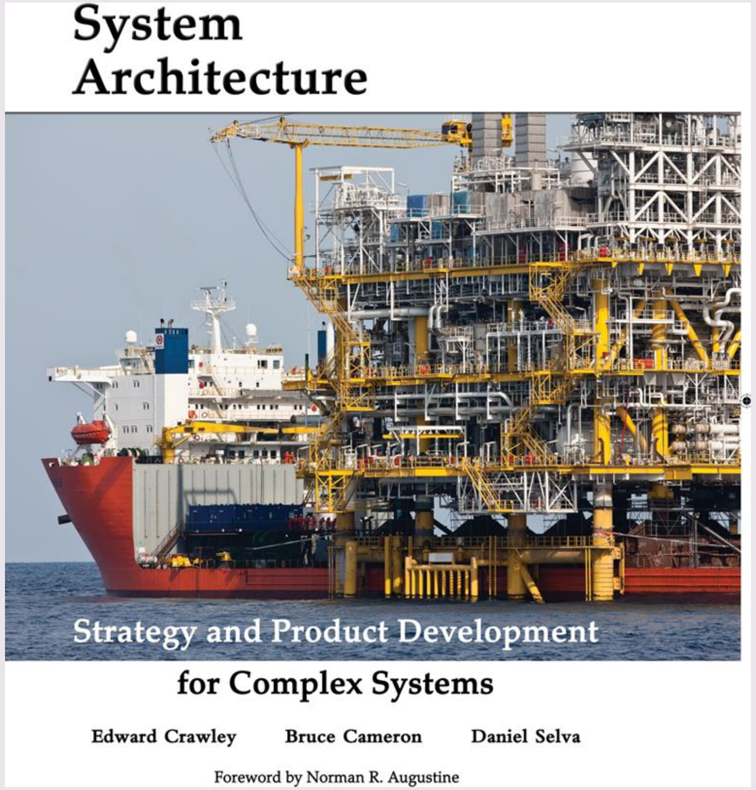
Edward Crawley Bruce Cameron Daniel Selva

Foreword by Norman R. Augustine

26 Principles of System Architecture

- *Principles:* underlying and long-enduring fundamentals that are always (or nearly always) valid.
- *Methods:* ways of organizing approaches and tasks to achieve a concrete end (grounded on principles)
- As architect, you'll develop your own set of principles and methods

Principle of Benefit Delivery



26 Principles of System Architecture

Good architectures deliver benefit,
first and foremost,

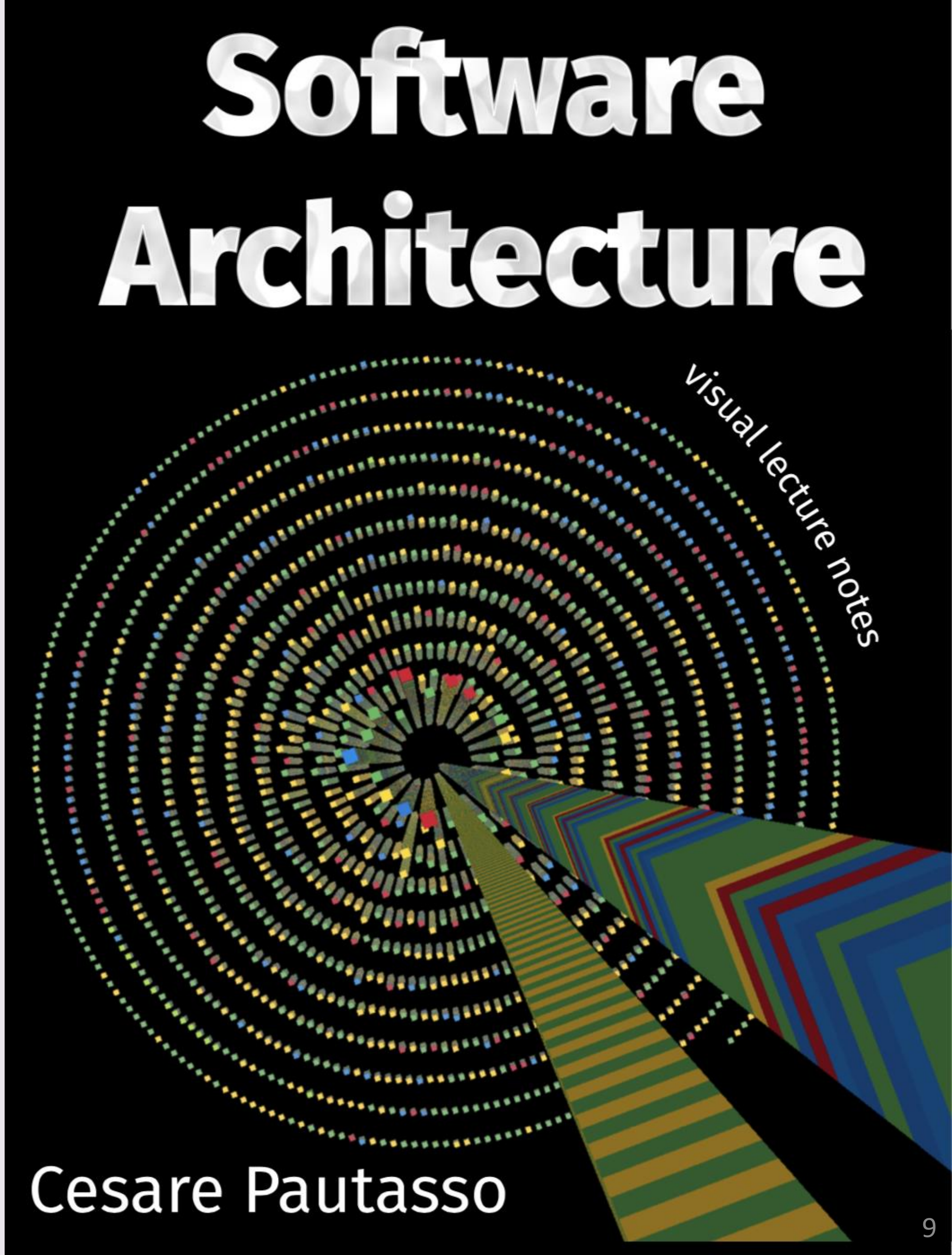
built on the primary externally delivered
function of the systems

by focusing on
the emergence of functions,
and their delivery

across the system boundary
at an interface.

1. Introduction
2. Quality Attributes
3. Definitions
4. Modeling Software Architecture
5. Modularity and Components
6. Reusability and Interfaces
7. Composability and Connectors
8. Compatibility and Coupling
9. Deployability, Portability and Containers
10. Scalability
11. Availability and Services
12. Flexibility and Microservices

https://leanpub.com/software-architecture



Course Objectives: You'll Learn How To:

1. Structure a system's problem space
2. Architect a software solution that meets stakeholder needs
3. Manage evolving needs, keeping architecture aligned
4. Understand and assess architectures of existing software systems
5. Use proof-of-concepts for architectural decision making
6. Communicate architectural decisions
7. Fulfill the role of an architect

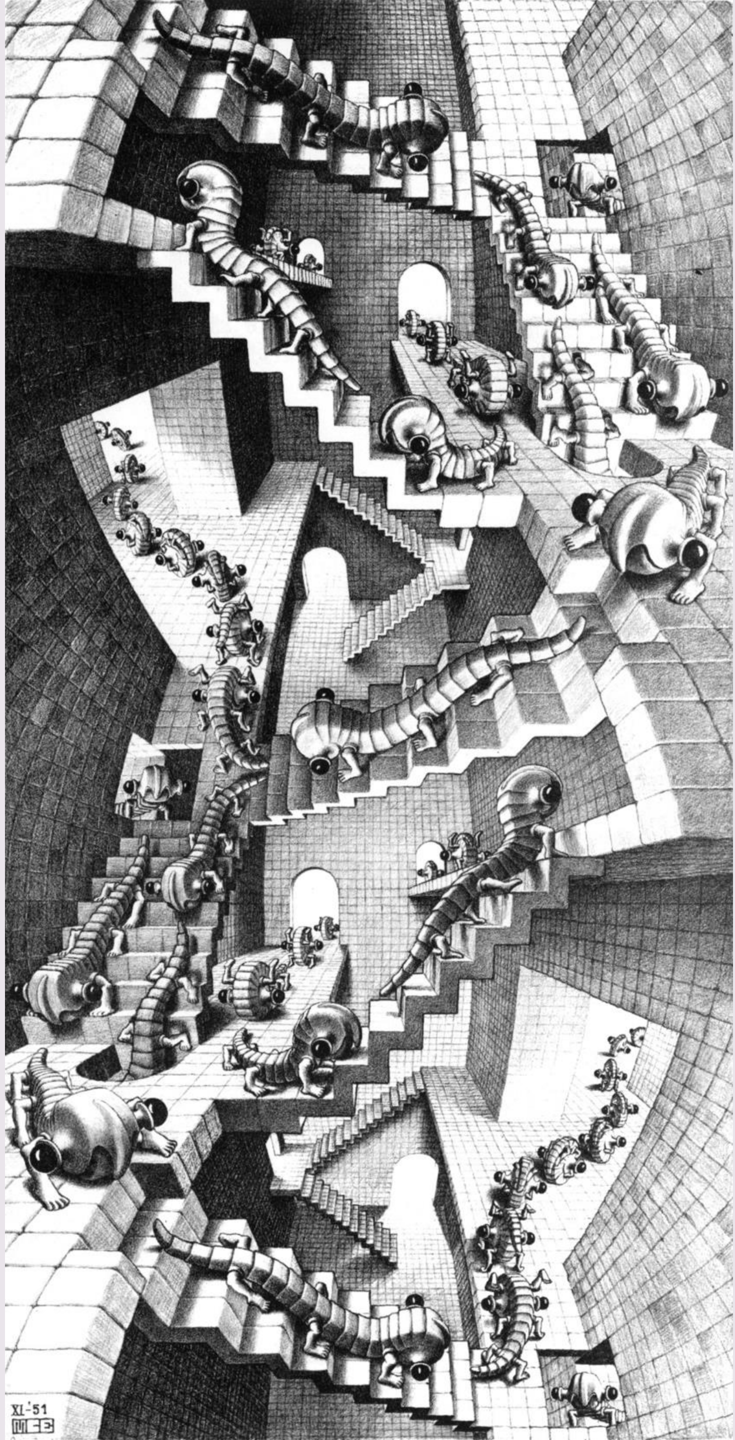
Image credit: Denis Andernach

Labwork Key Elements

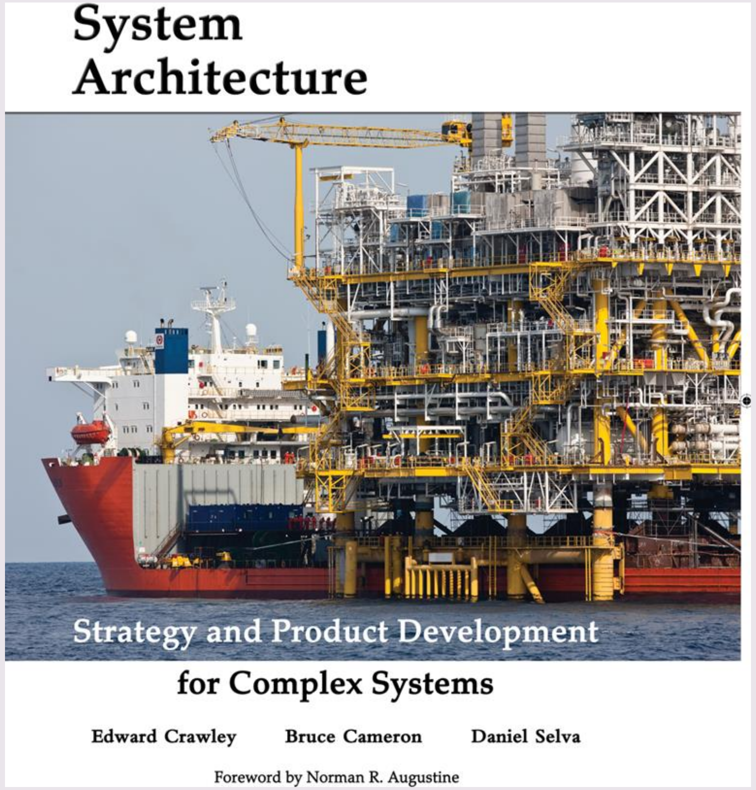
- Work in teams of four
- Develop architecture for *system of choice*
  - Carefully written text and system models
  - Analyze problem and solution domain, and their interplay
- Build proof-of-concept (PoC)
- Grow reports + PoC in three iterations (W3, W6, W9)
- Evaluate architectures from peer teams
- Embrace freedom and open learning

LO1:
Structure The Problem Space

- Identify the primary externally delivered value-related function
- Identify architecturally relevant properties and scenarios
- Resolve upstream *ambiguity*
- Reconcile conflicting stakeholder needs
- Highlight technical limits *and opportunities*



Principle of Ambiguity



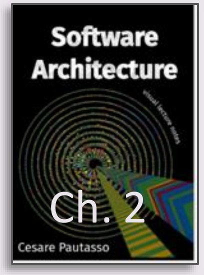
26 Principles of System Architecture

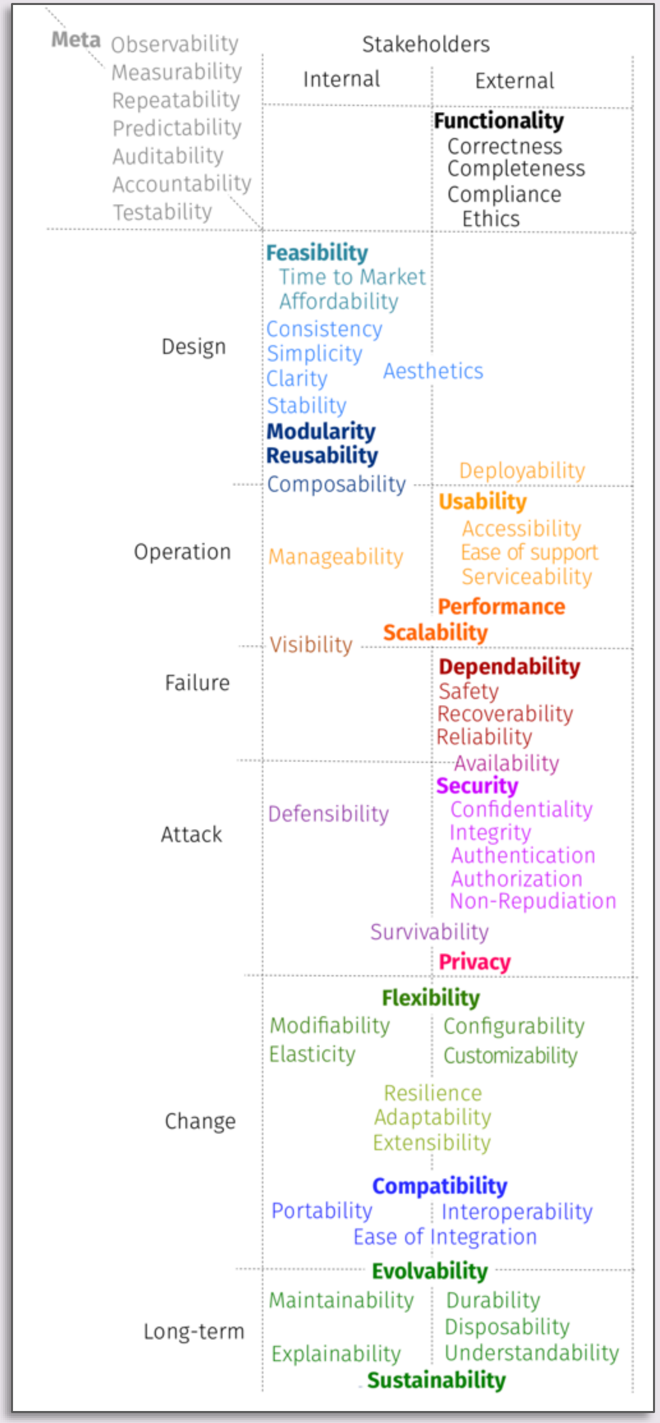
The early phase of a system design is characterized by great ambiguity.

The architect must resolve this ambiguity to produce (and continuously update) goals for the architect’s team.

LO2:
Develop An Architecture

- Develop form that realizes system function
- Represent the system in different views using relevant modeling techniques
- Identify tradeoff points and alternative solutions
- Build on known patterns and styles to realize required quality attributes
- Manage (reduce) complexity
- Reuse (open source) libraries and frameworks





Meta

Observability
Measurability
Repeatability
Predictability
Auditability
Accountability
Testability

Stakeholders

Internal

External

Functionality
Correctness
Completeness
Compliance
Ethics

Design

Feasibility
Time to Market
Affordability
Consistency
Simplicity
Clarity
Aesthetics
Stability
Modularity
Reusability
Composability
Deployability

Operation

Usability
Accessibility
Ease of support
Serviceability
Manageability
Performance
Scalability

Visibility

Failure

Dependability
Safety
Recoverability
Reliability
Availability

Attack

Security
Confidentiality
Integrity
Authentication
Authorization
Non-Repudiation
Defensibility
Survivability
Privacy

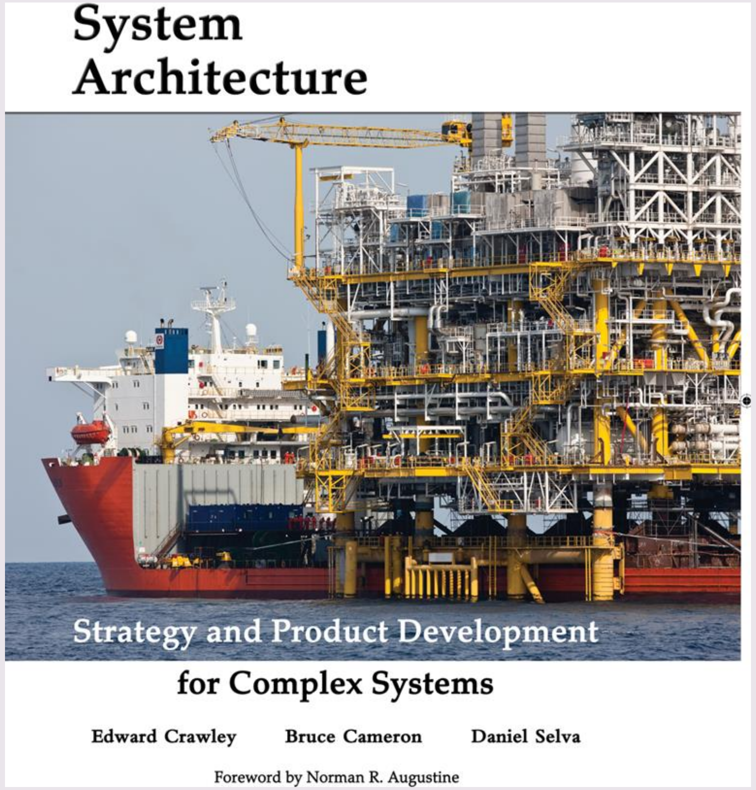
Change

Flexibility
Modifiability
Configurability
Elasticity
Customizability
Resilience
Adaptability
Extensibility
Compatibility
Portability
Interoperability
Ease of Integration

Long-term

Evolvability
Maintainability
Durability
Disposability
Explainability
Understandability
Sustainability

Principle of Value and Architecture



26 Principles of System Architecture

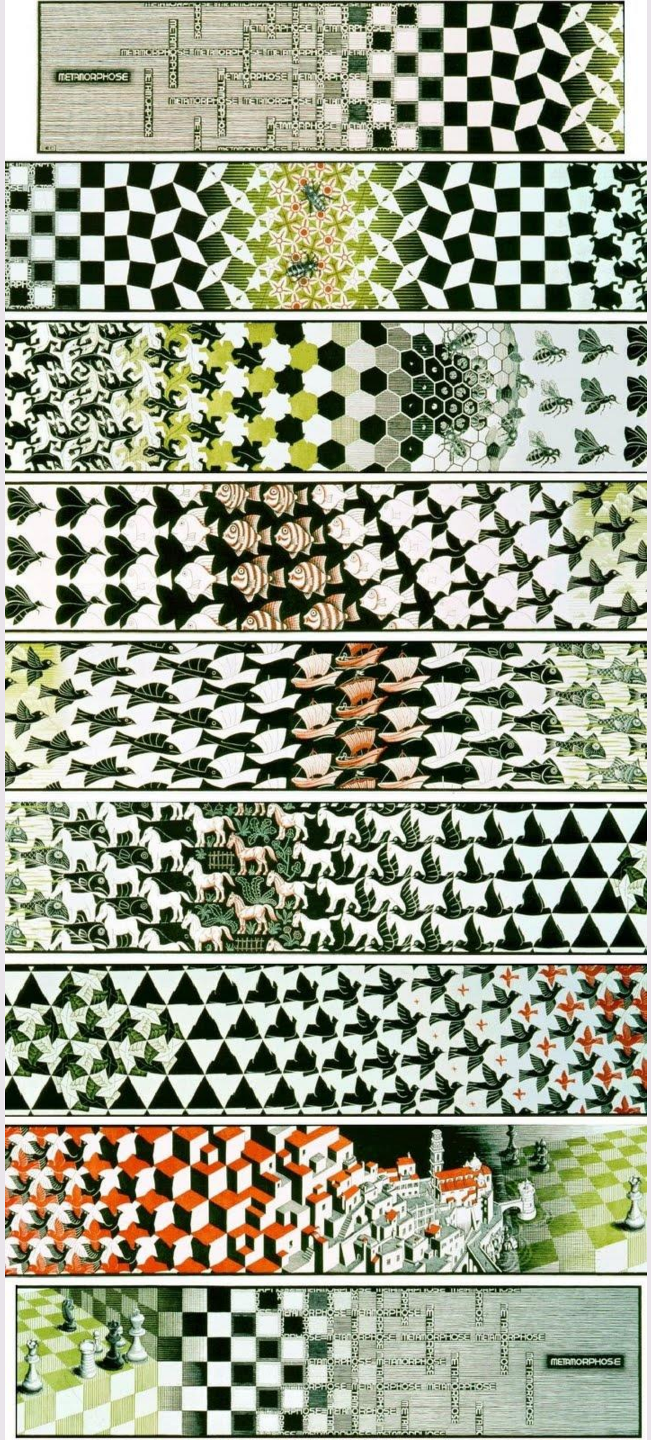
Value is benefit at cost.

*Architecture is function
enabled by form.*

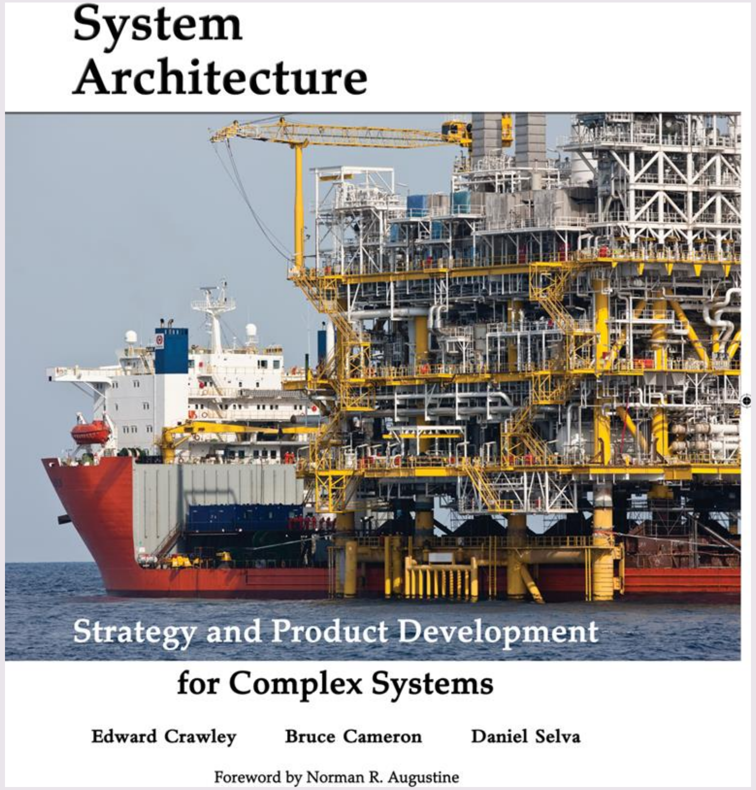
*There is a very close relationship between
these two statements,
because
benefit is delivered by function,
and form is associated with cost.*

LO3:
Manage Evolving Needs

- Setup an observability and experimentation infrastructure to monitor operations,
  with feedback to business and development
- Design and realize test infrastructure supporting safe evolution
- Identify technical debt and architectural erosion and resolve it cost-effectively
- Handle legacy systems effectively
- Support variability with software product lines.



Principle of Evolution



26 Principles of System Architecture

Systems will evolve
or lose competitive advantage.

When architecting,
define the interfaces
as the more stable parts of the system
so that the elements can evolve.

https://www.adviescollegeicttoetsing.nl/onze-werkwijze/documenten/publicaties/2021/12/01/toetskader-acict

Toetskader

Adviescollege ICT-toetsing

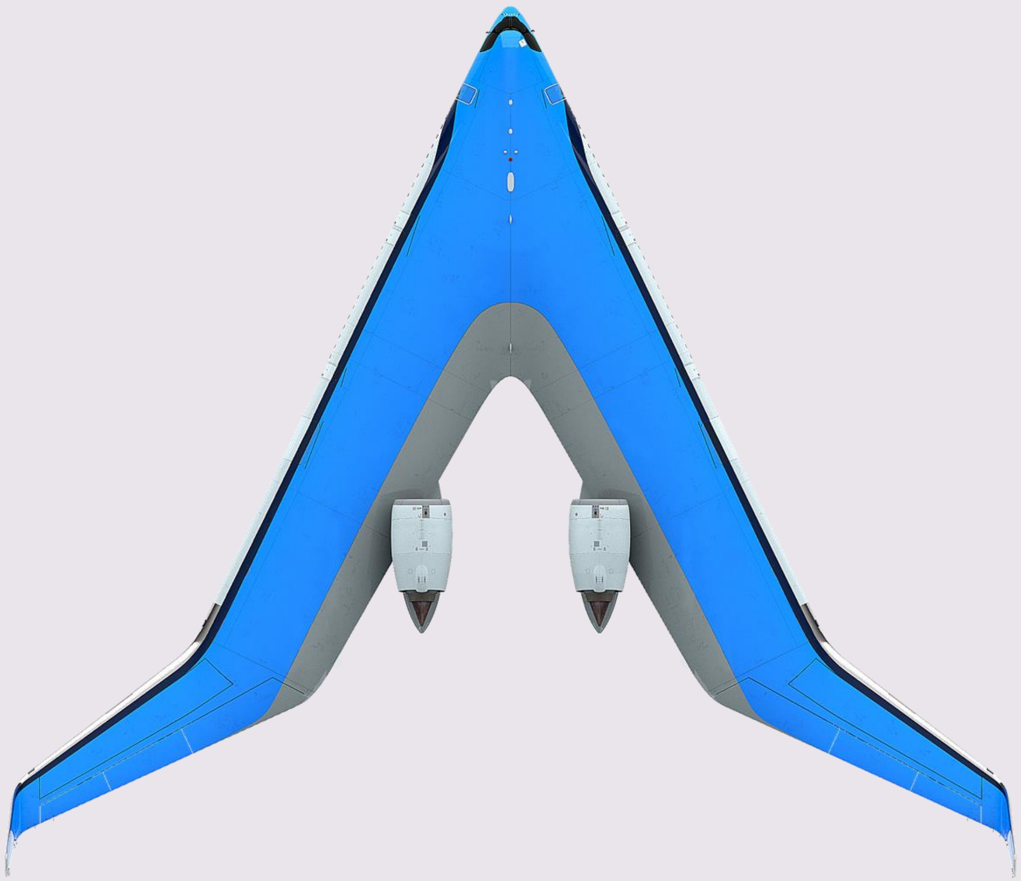
versie december 2021

LO4: Evaluate Architectures

- Learn from best practices in exemplary systems
- Greenfield engineering is the exception;
  norm is ”brown field”, adjusting existing systems
- Distinguish what’s good / bad
  about an existing architecture
- Contrast “as implemented” with “as designed”
- Use automated tools to analyze system artefacts and repositories
- Use assessment frameworks

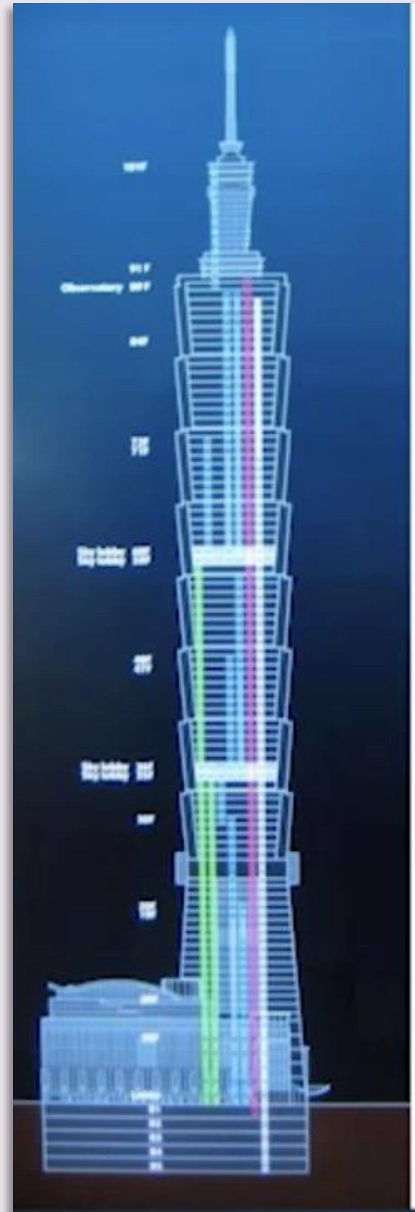
LO5: Build and Use Proof-of-Concepts

- Formulate hypotheses underlying architectural decisions
- Test hypothesis with prototypes
  - (run time) measurements
  - (design time) demonstrators of API suitability
- Critically analyze experimental results and validity of the conclusions



LO6: Communicate Architectural Decisions

- Decisions take time to make
- Develop consensus and support
- Document decisions unambiguously
- Write effectively
- Model at right level of abstraction
- Offer compelling presentations
- Listen and adapt



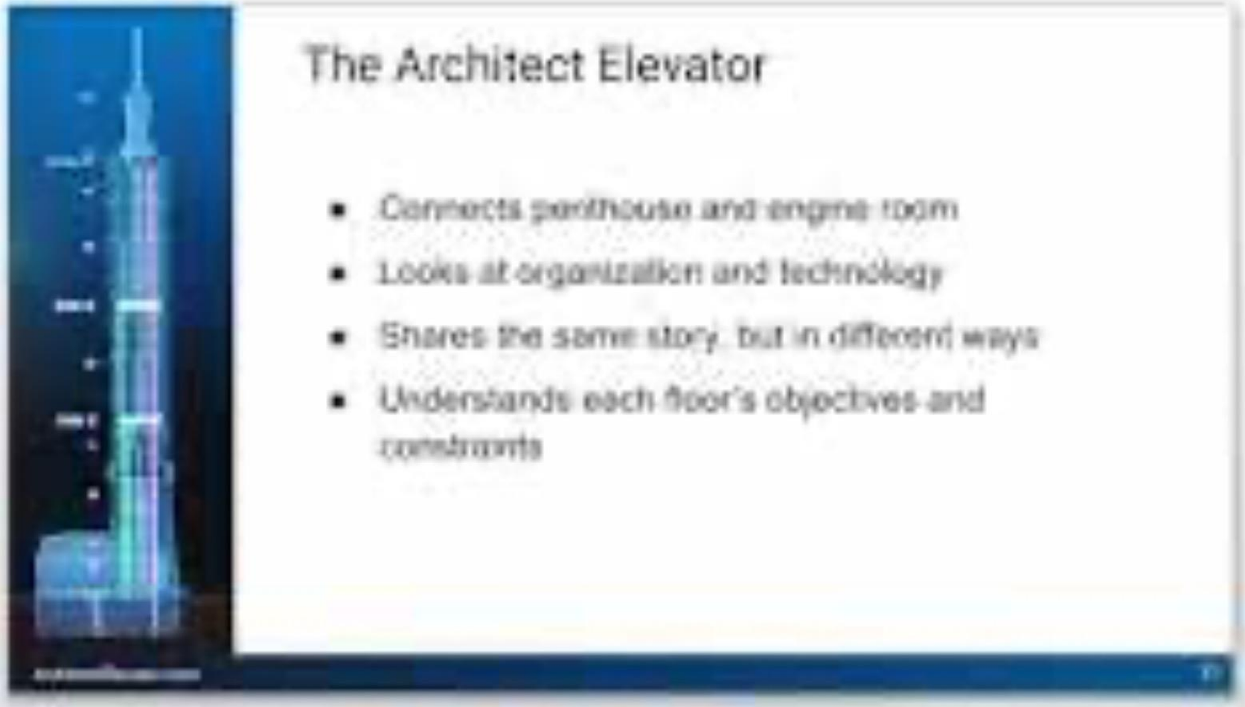
The Architect Elevator

- Connects penthouse and engine room
- Looks at organization and technology
- Shares the same story, but in different ways
- Understands each floor's objectives and constraints

ArchitectElevator.com 13

Gregor Hohpe

https://www.youtube.com/watch?v=Zq2VcRZmz78&t=670s

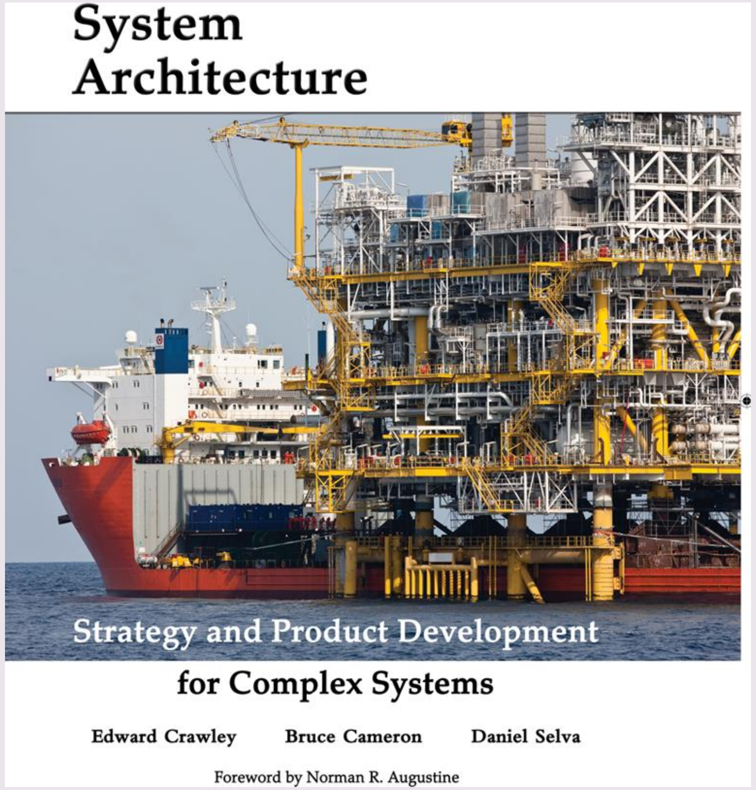


The Architect Elevator

- Connects penthouse and engine room
- Looks at organization and technology
- Shares the same story, but in different ways
- Understands each floor's objectives and constraints

TNG TECHNOLOGY CONSULTING

LO7: (Principle of)
The Role of the Architect



26 Principles of System Architecture

The role of the architect is to

resolve ambiguity,

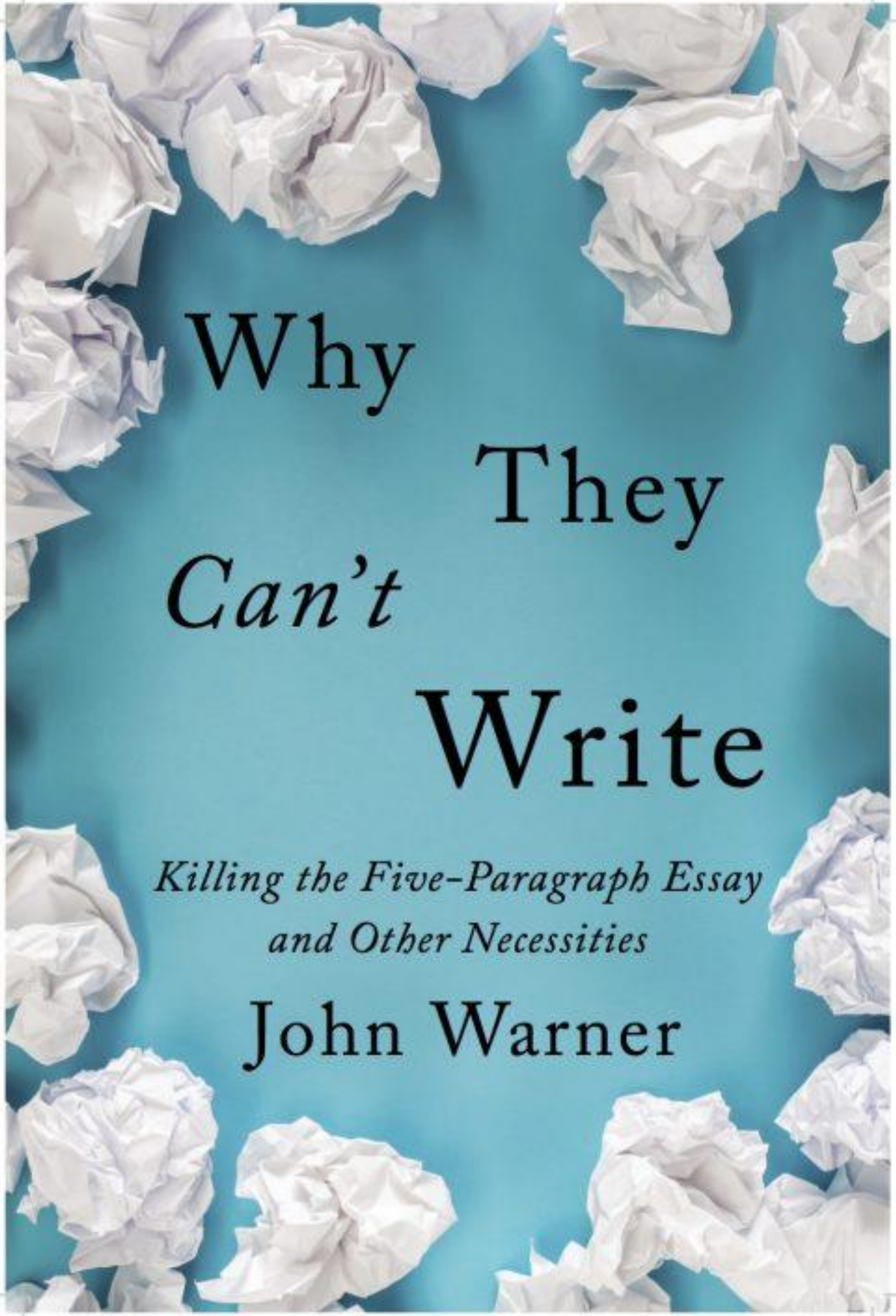
focus creativity,

and simplify complexity

Software Architecture as a *Practice*

A *practice* consists of four primary dimensions:

1. **Knowledge:**
   What do software architects know?
2. **Skills:**
   What can software architects do?
3. **Habits of mind:**
   How do software architects think?
4. **Attitudes:**
   What do software architects believe and value about being a software architect?



Attitudes: *Values* in the Agile Manifesto

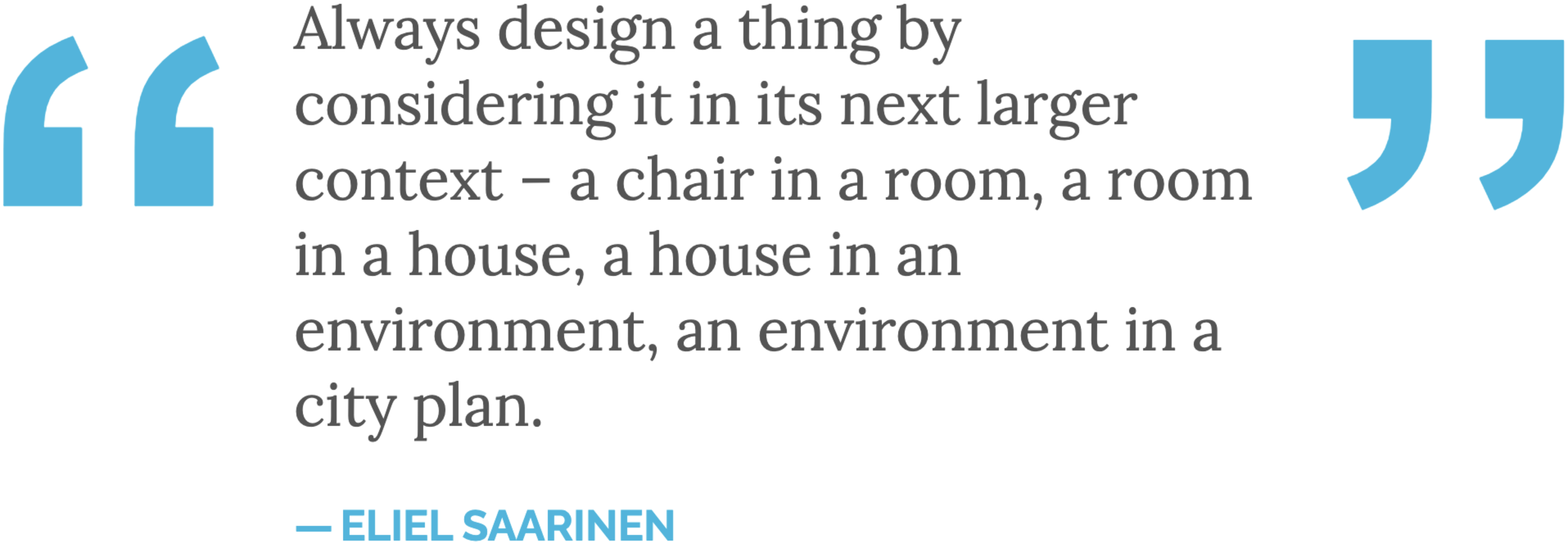
“We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

1. *Individuals and interactions over processes and tools*
2. *Working software over comprehensive documentation*
3. *Customer collaboration over contract negotiation*
4. *Responding to change over following a plan*

That is, while there is value in the items on the right,
we value the items on the left more.”

https://agilemanifesto.org/, 2001

Habits of Mind

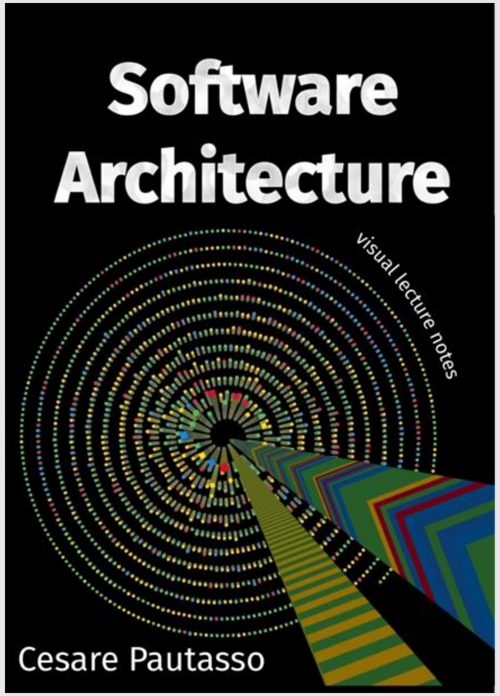


> Always design a thing by considering it in its next larger context – a chair in a room, a room in a house, a house in an environment, an environment in a city plan.
>
> — ELIEL SAARINEN

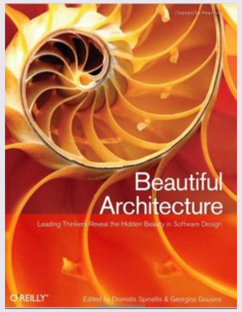
https://quotesondesign.com/eliel-saarinen/

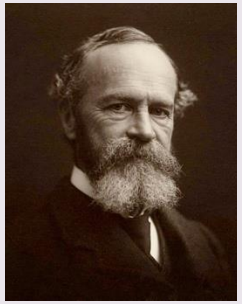
Knowledge:
What do software architects know?

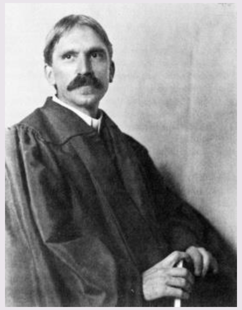
- Implicit:
  - The experience you gain from working with systems
- Explicit:
  - Knowledge codified into patterns, processes, styles, ...
  - Modeling and model analysis techniques
- “Systems Zoo” – learn from great examples
  - The reflective engineer
- Epistemology / what can we know?
  - When do we consider architectural knowledge valid?



The Architecture of Open Source Applications
Volume II: Structure, Scale, and a Few More Fearless Hacks
Edited by Amy Brown & Greg Wilson


Beautiful
Architecture
Leading Thinkers Reveal the Hidden Beauty in Software Design
O'REILLY
Edited by Diomidis Spinellis & Georgios Gousios





Pragmatism / James / Dewey

Technology Strategy Patterns
The Phoenix Project
The Software Architect Elevator
All Othe
Context
Just Enough Software Architecture
Software Systems Architecture
Thinking in Systems
The Model Thinker
Entire System
Building Microservices
Fundamentals of Software Architecture
Enterprise Integration Patterns
Pattern-Oriented Software Architecture
Connected Components
Clean Code
Refactoring
Design Patterns
Pattern-Oriented Software Architecture
Single Component

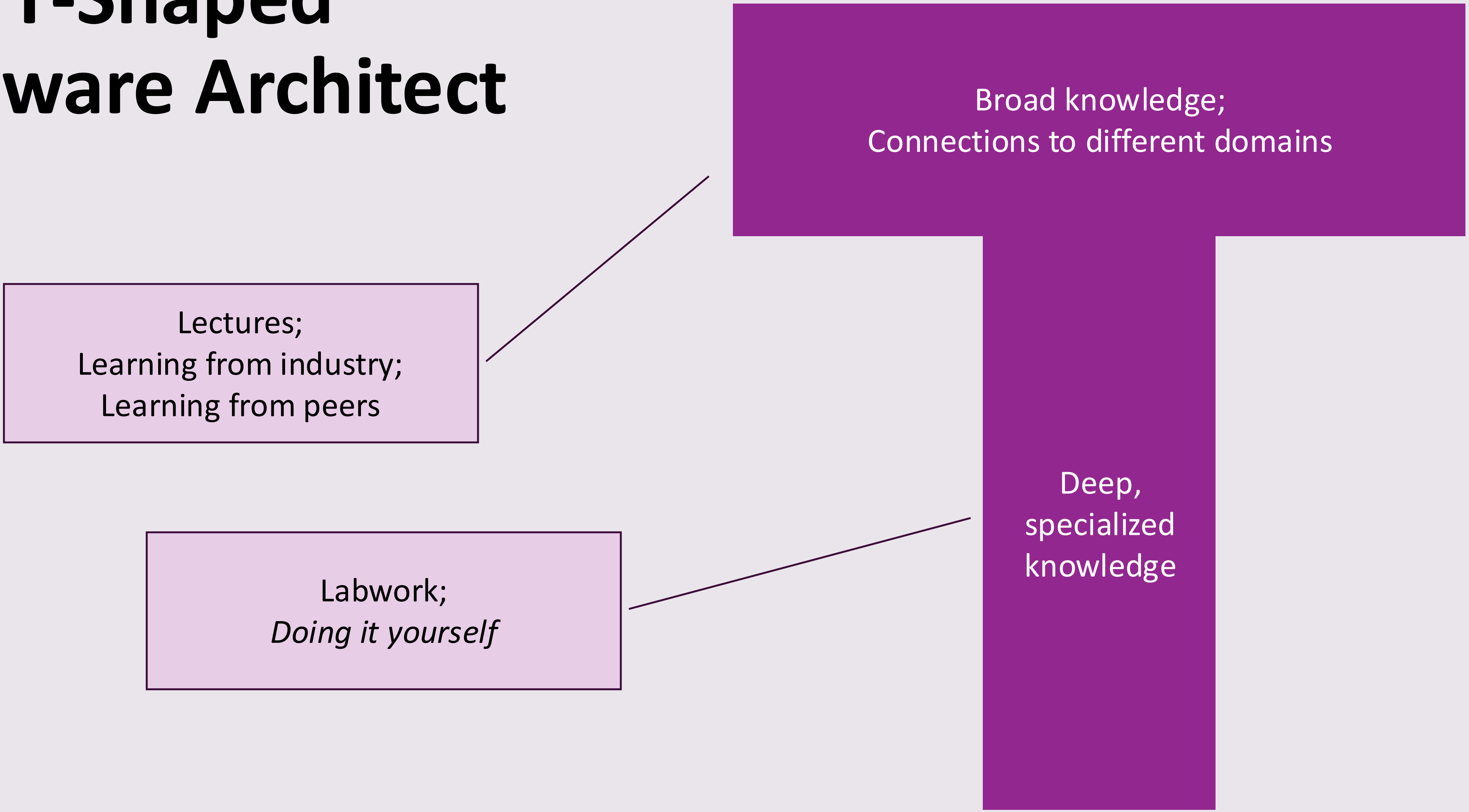
← Gregor Hohpe's "Architect's Bookshelf" -- https://architectelevator.com/architecture/architect-bookshelf/

Architects as Knowledge Crunchers

- A very broad topic
- Mix of people skills, technical skills, and domain sensitivity
- Reusable architectural knowledge often *abstract*
- Architects need ability to make such knowledge *concrete* in their own context
- The architect is never finished learning

My list: https://bookwyrm.social/list/2266/s/software-architecture

The T-Shaped Software Architect



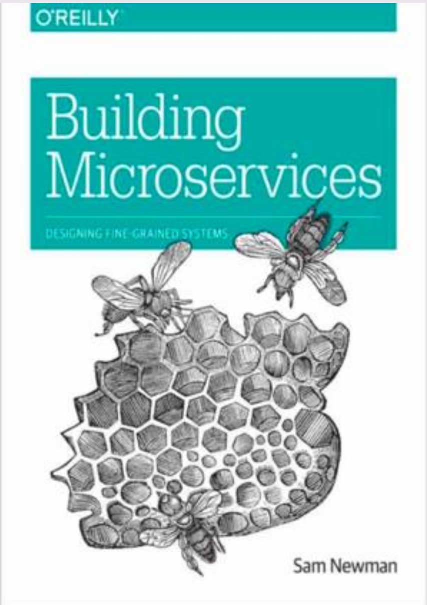
Broad knowledge;
Connections to different domains

Lectures;
Learning from industry;
Learning from peers

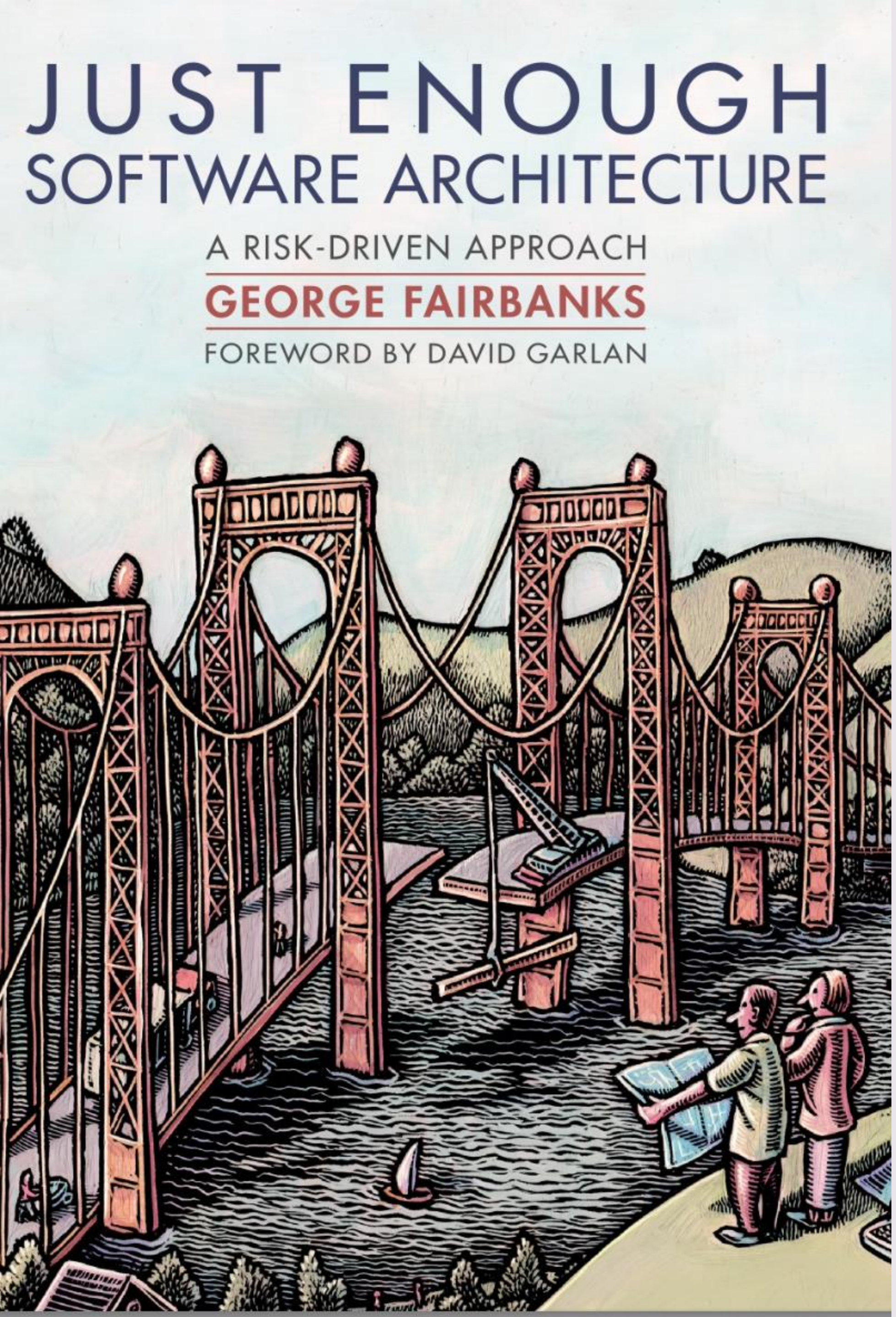
Deep,
specialized
knowledge

Labwork;
Doing it yourself

Core Skills of the “Evolutionary Architect”



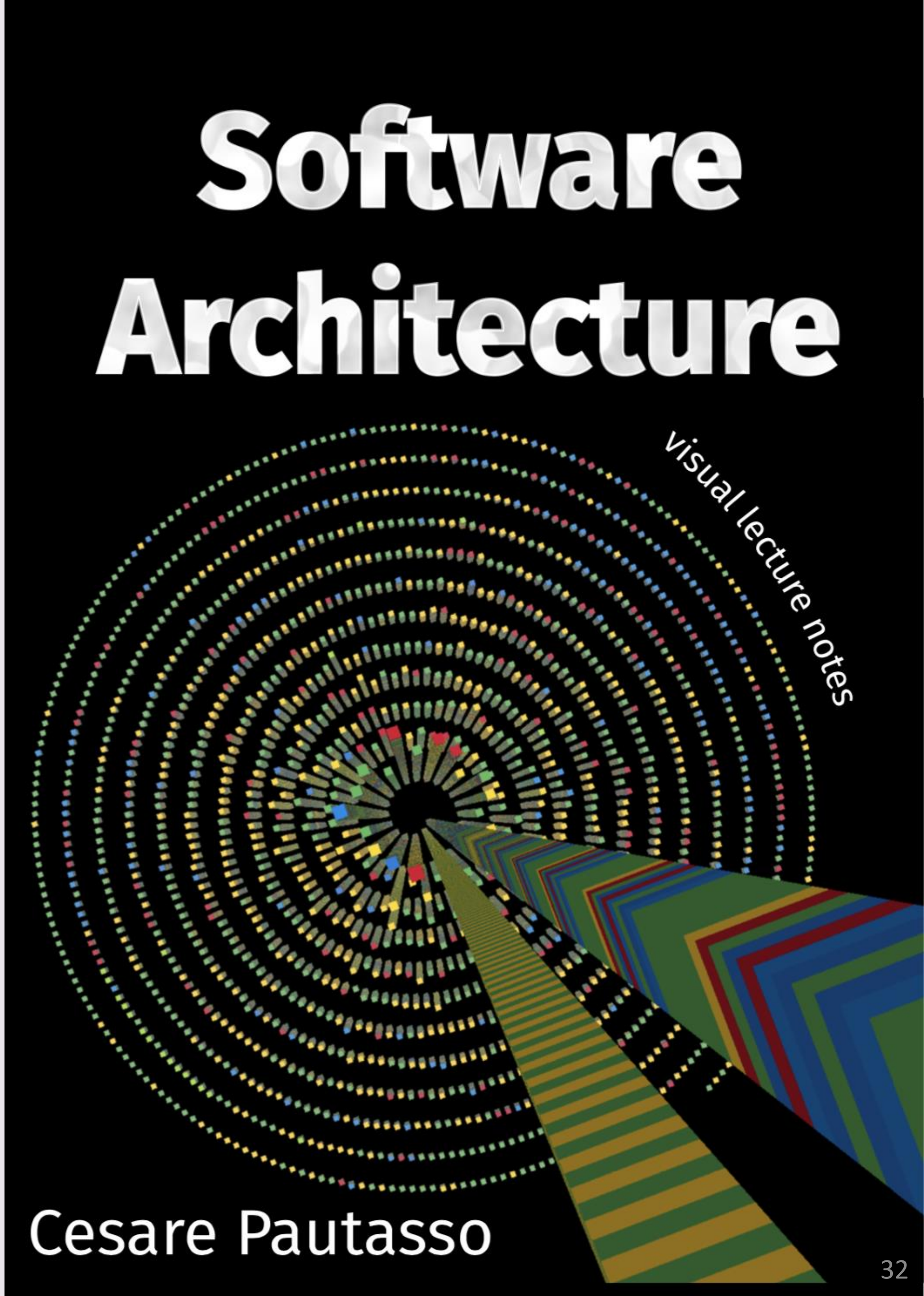
- **Vision**: Ensure there is *a clearly communicated technical vision* for the system that will help your system *meet the requirements of your customers and organization*
- **Empathy:** Understand impact of your decisions on end users and team
- **Collaboration:** Engage with as many people as possible to realize vision
- **Adaptability:** Adjust vision when needed
- **Autonomy:** Balance (giving) autonomy and overall consistency
- **Governance:** Ensure system built meets vision



Core Skills: Knowing “When to Architect”

- Effort should be commensurate with *risk of failure*
- Risk = chance of event * cost of event
- Risk categories:
  - Engineering risks meeting key quality attributes
  - Management risks (e.g., customer rejection, late to market)
- Each project faces different risks;
  no single arch. approach
- Some projects “highly precedented” – almost no risk if you follow proven architecture
- Other projects highly novel / unchartered / high stakes: careful architectural analysis crucial

1. Introduction
2. Quality Attributes
3. Definitions
4. Modeling Software Architecture
5. Modularity and Components
6. Reusability and Interfaces
7. Composability and Connectors
8. Compatibility and Coupling
9. Deployability, Portability and Containers
10. Scalability
11. Availability and Services
12. Flexibility and Microservices



Software Architecture

visual lecture notes

Cesare Pautasso

Lecture Schedule (Tentative)

- W1, Tue: Intro & Labwork
- W1, Wed: Problem analysis, domain modeling
- W2, Tue: Solution analysis, writing for decision makers
- W2, Wed: Architectural styles
- W3, Tue: Quality attributes, scalability (Diomidis Spinellis)
- W3, Wed: The Unix architecture (Diomidis Spinellis)

Lecture Schedule (Tentative)

- W4, Tue: Guest lecture Miro
- W4, Wed: No lecture (R1 Refinement)
- W5, Tue: Guest lecture Exact
- W5, Wed: (Remote) API design
- W6, Tue: Architecture modernization
- W6, Wed: Government IT
- W7, Tue: TBD
- W7, Wed: Guest lecture Adyen

Software Architecture

CS4505, TU Delft, 2024/2025

Arie van Deursen & Diomidis Spinellis

Image credit: Corné Bastiaansen www.rijksvastgoedbedrijf.nl