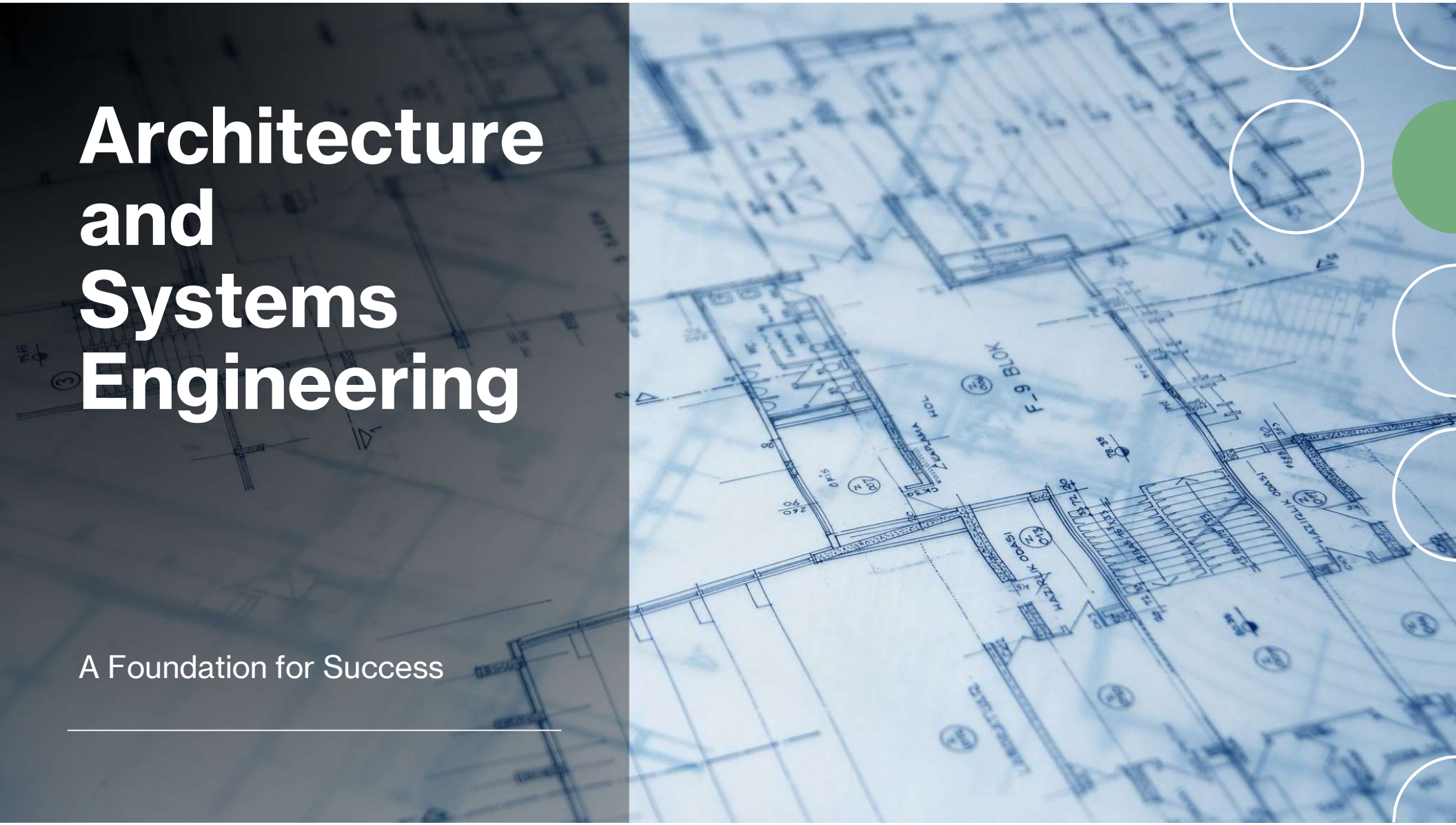
Architecture and Systems Engineering

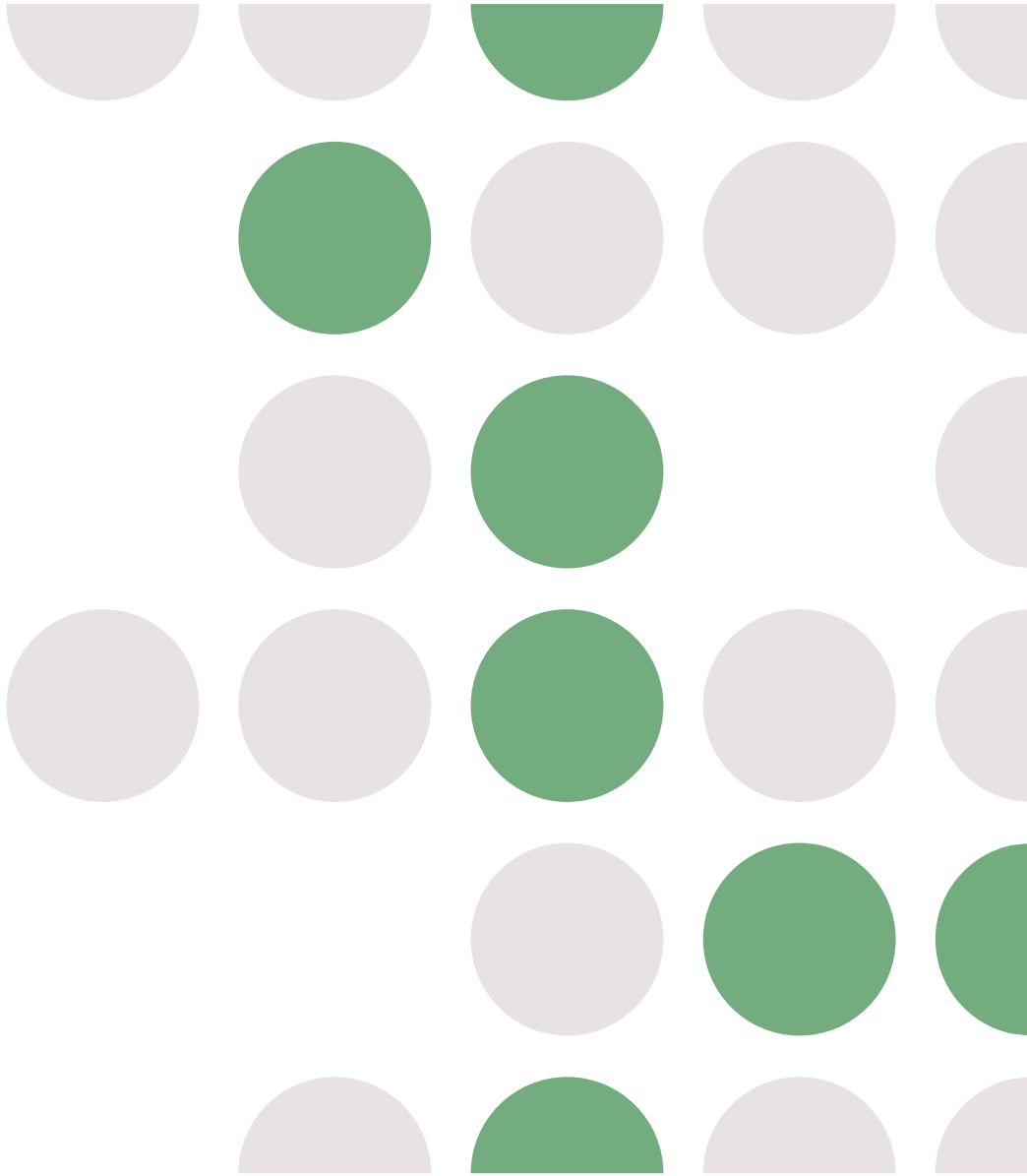
A Foundation for Success



"The boldest of the design decisions, whoever made them, have accounted for a high fraction of the goodness of the outcome. “

Fred Brooks

The Design of Design

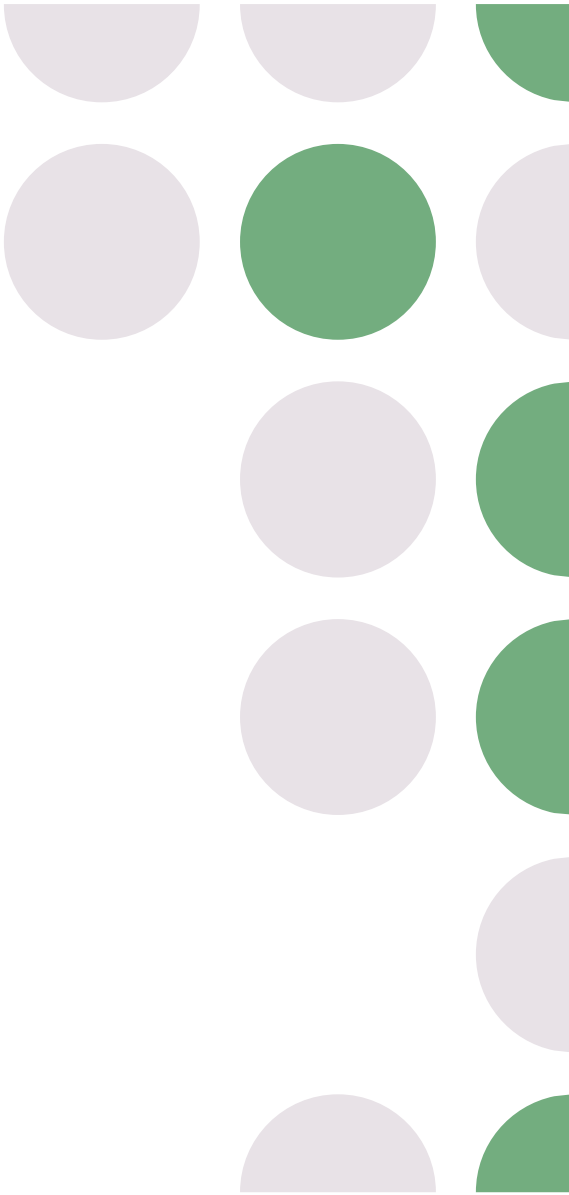


Andy Unterseher

- Software Architect at Don’t Panic Labs
- 10+ years of experience designing and developing software systems
- @unter
- https://dontpaniclabs.com/blog

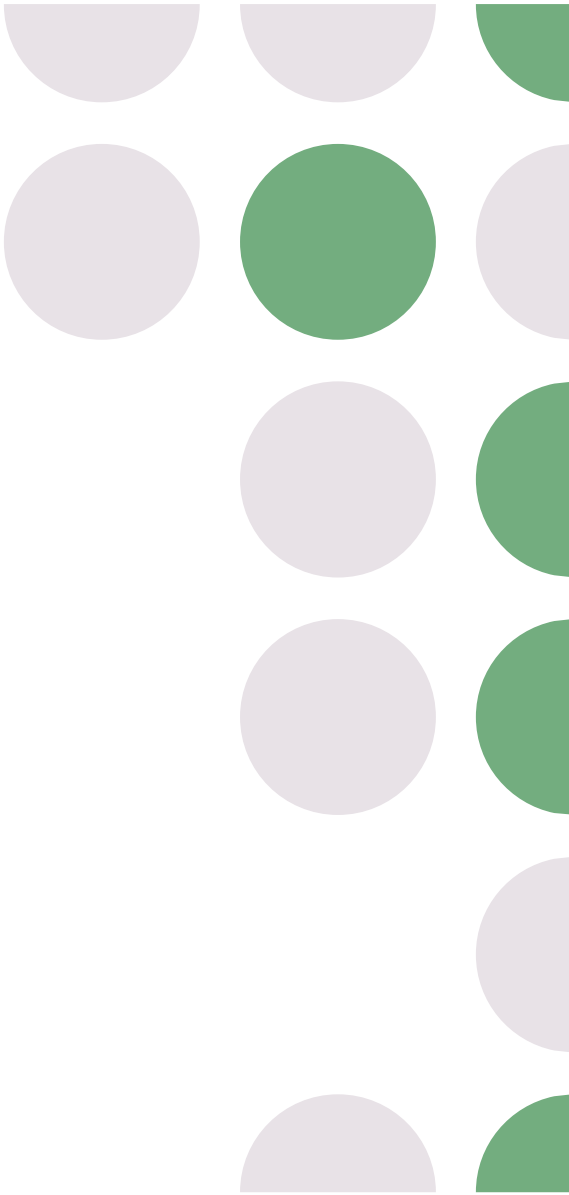
Agenda

- History
- Concepts
- Architectures



Before DPL

- Classic ASP
  - SQL Statements in the HTML
- Object Oriented Systems
- N-Tier Architectures
- All Ball of Mud



Zen of Architecture

For the beginner architect, there are many options

For the master architect, there are only a few

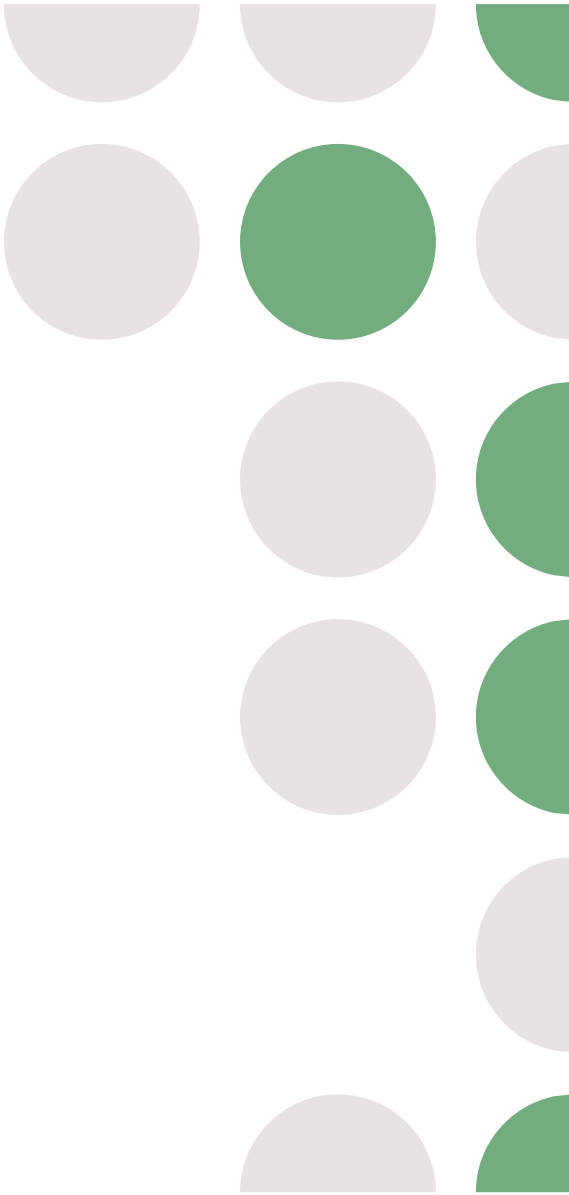
Juval Lowy

---

https://www.youtube.com/watch?v=Jxm2rgeuC6s

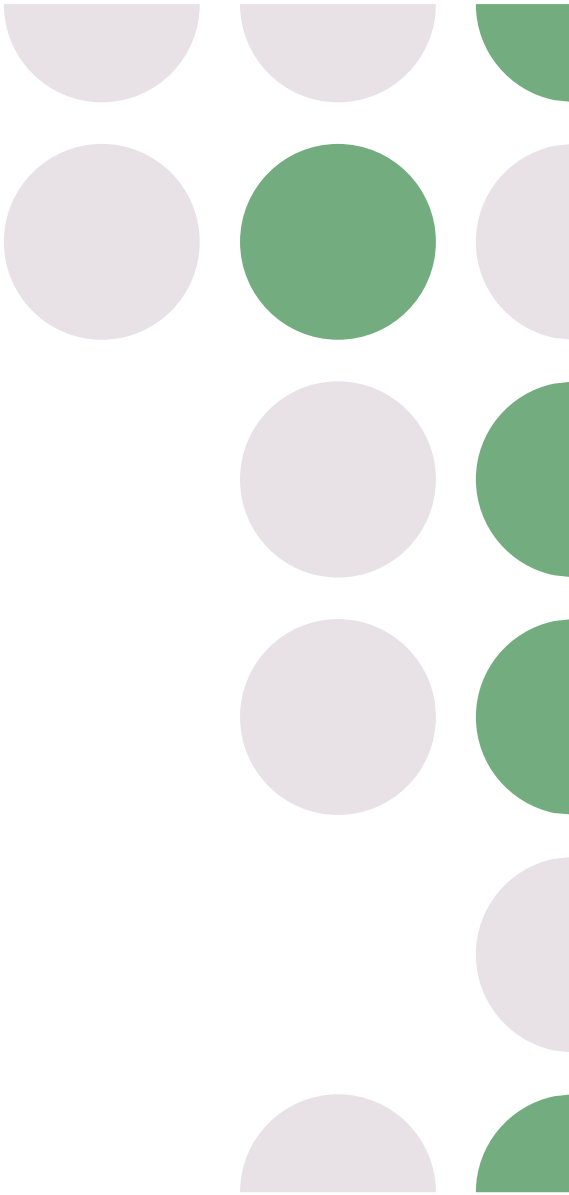
DPL Architecture Goal

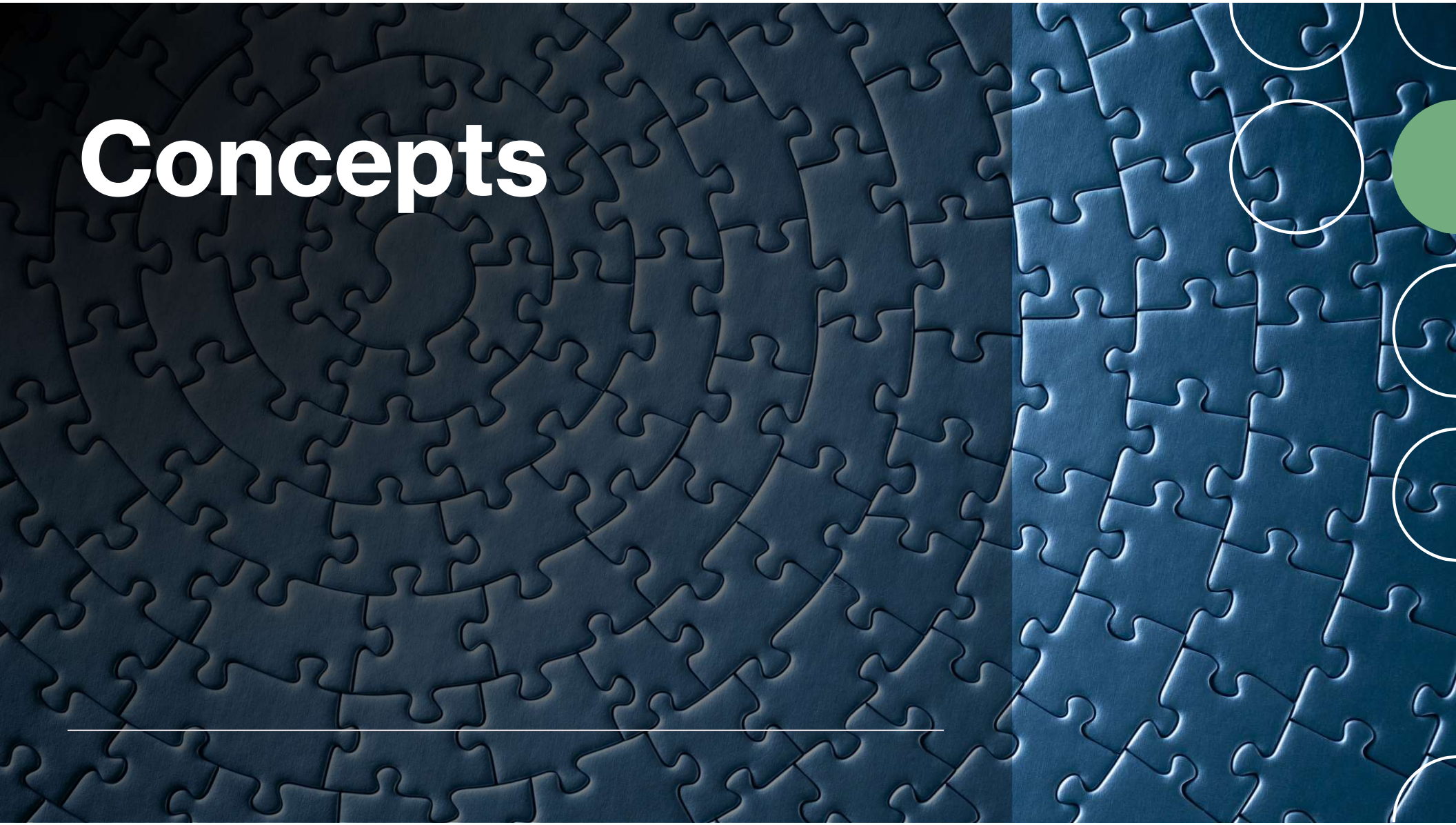
- Build many new companies with
  - A flexible engineering staff
  - Common Design Methodology
  - Repeatable Architecture Practice
  - Sustained agility in development



Experience at DPL

- Service Oriented Architecture
  - Volatility Based Decomposition
- Architected Systems for
  - E-commerce
  - Sports Performance
  - Medical
  - Financial
  - Agriculture



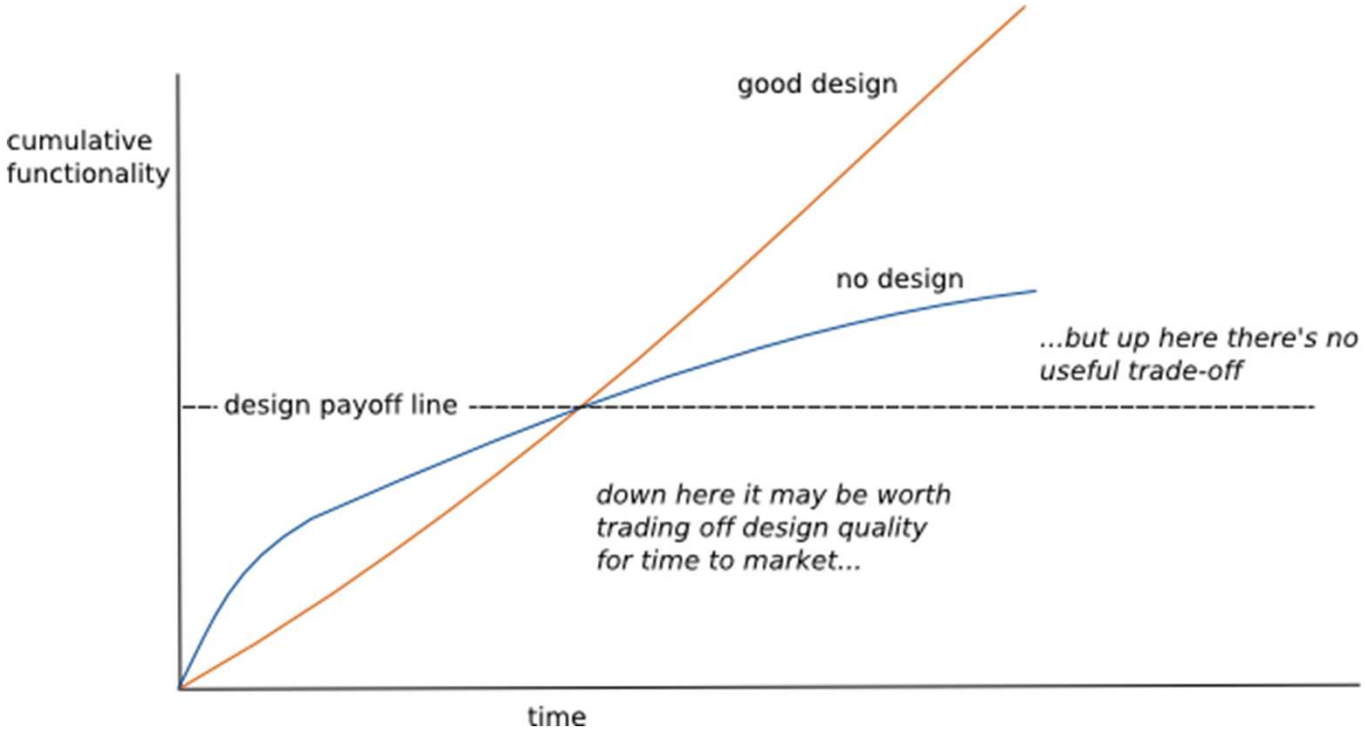

Concepts

Agile Manifesto

“Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. “

https://agilemanifesto.org/principles.html

Design Stamina Hypothesis



https://martinfowler.com/bliki/DesignStaminaHypothesis.html

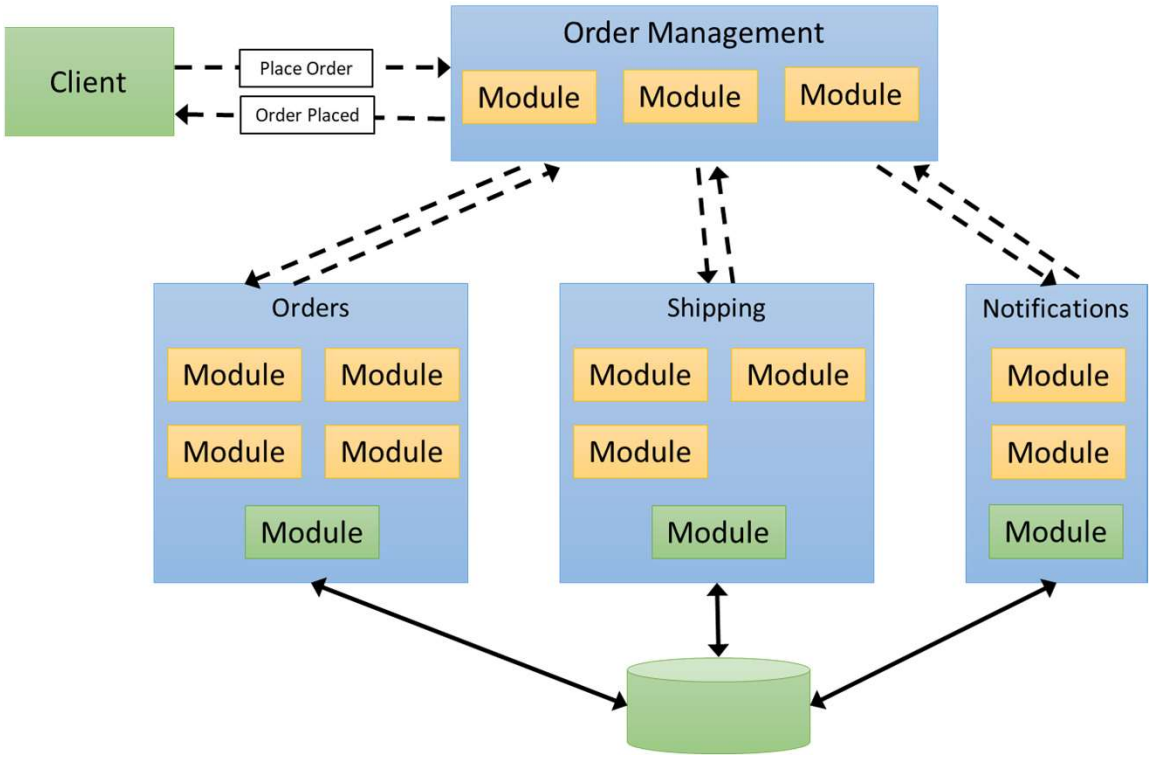
Systems Engineering

“Systems engineering is an interdisciplinary field of engineering and engineering management that focuses on how to design, integrate, and manage complex systems over their life cycles.”

---

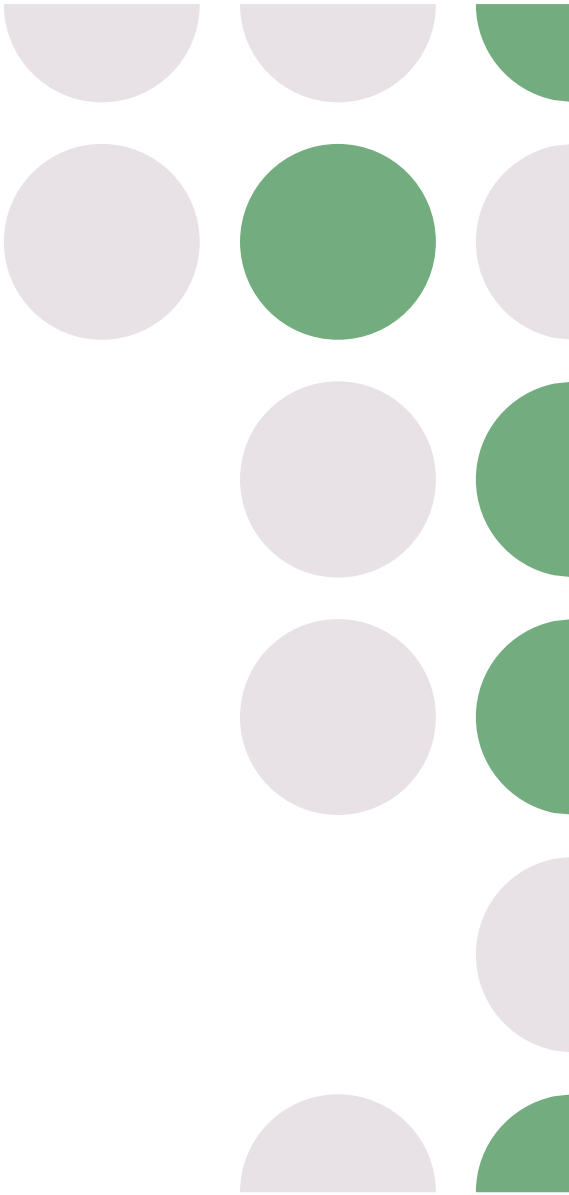
https://en.wikipedia.org/wiki/Systems_engineering

Boxes and Arrows

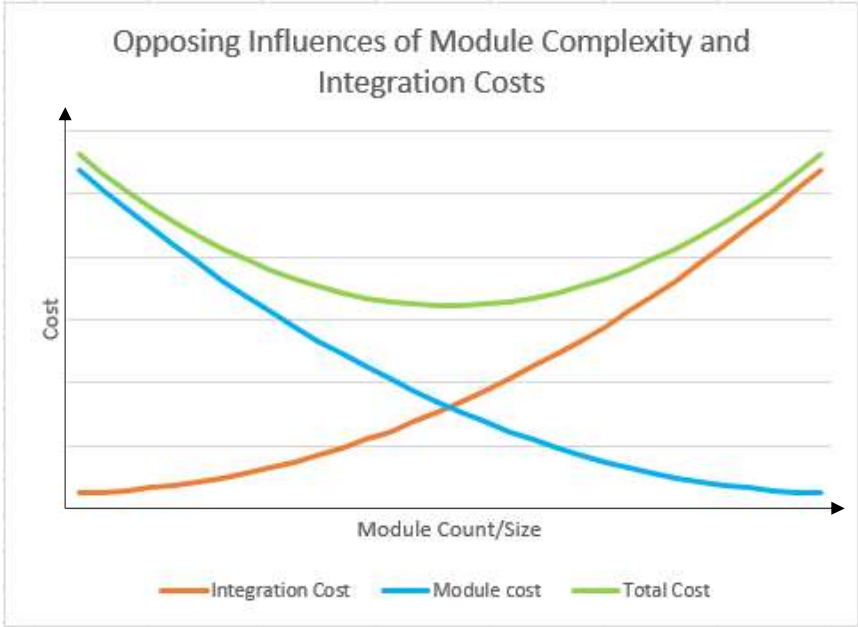


Tradeoffs

- Engineering is tradeoffs
  - There are no right and wrong designs
- Don’t leave Design to chance



Service vs. Integration Cost



Structured Design (Yourdon/Constantine 1979)

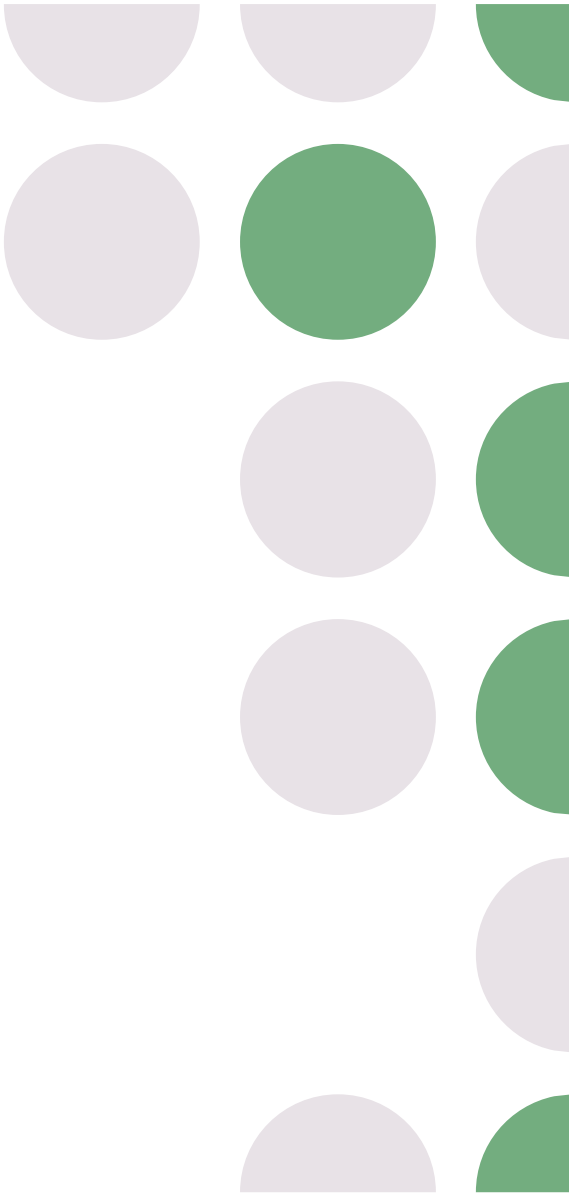

Monolith
Distributed
Increasing functionality, complexity, rate of change, team size...

Decomposition

"Usually nothing is said about the criteria to be used in dividing the system into modules"

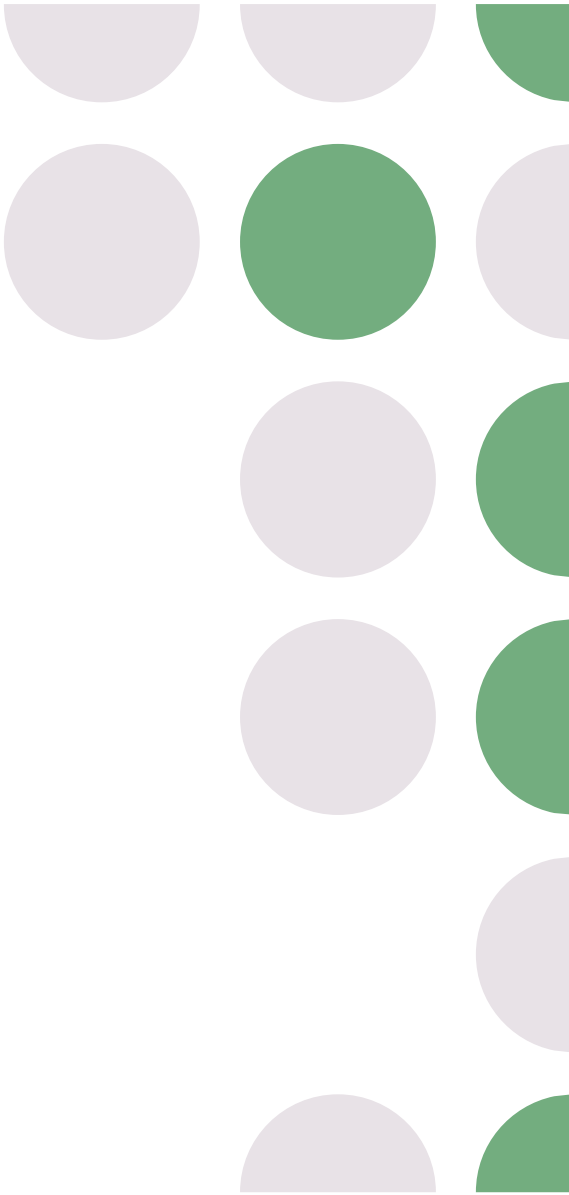
David Parnas

On the Criteria To Be Used in Decomposing Systems into Modules

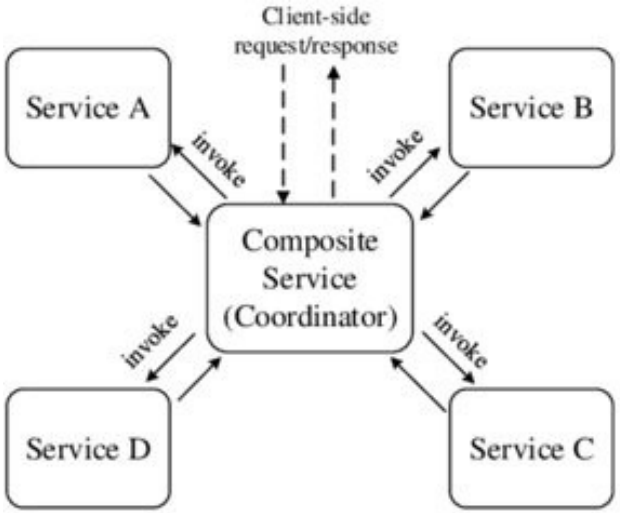


Subsystems / Domains

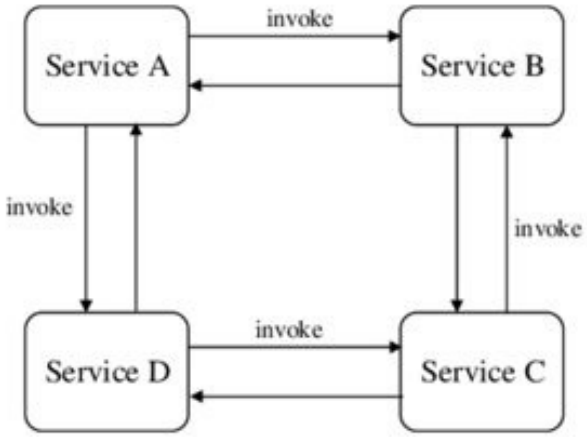
- Independent pieces of a larger system design



Orchestration and Choreography



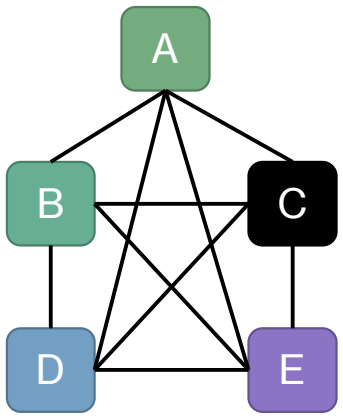
(a) Web Service Orchestration



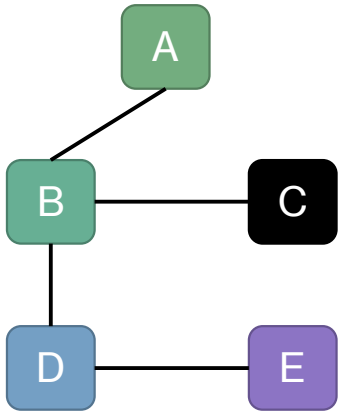
(b) Web Service Choreography

Open to Closed Architecture

- Rules/constraints in module interactions (closed) or the lack of rules/constraints (open)



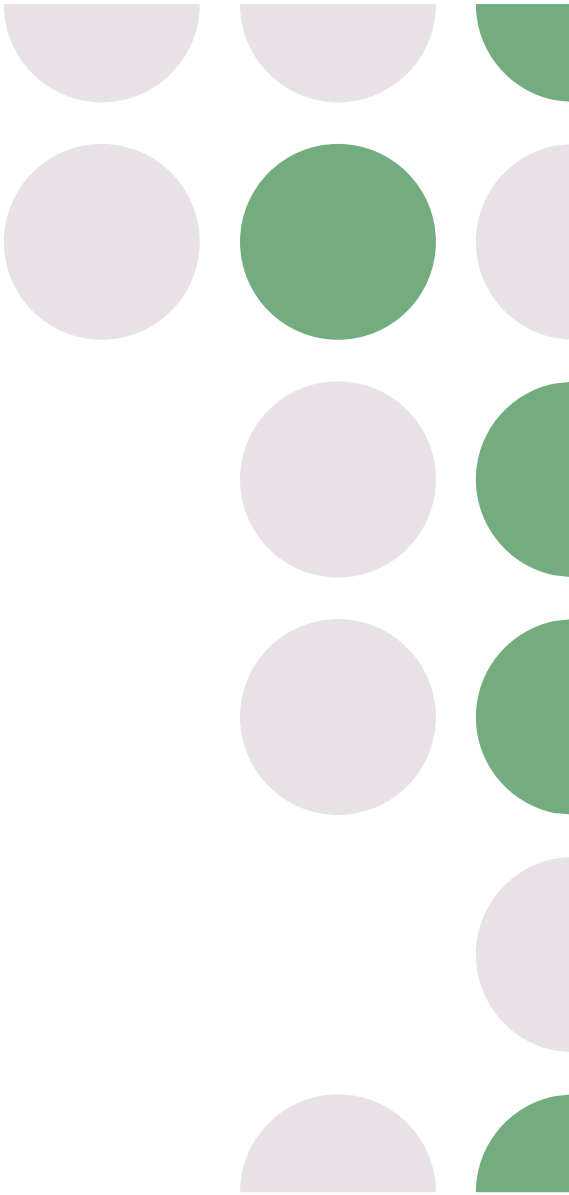
Open



Closed

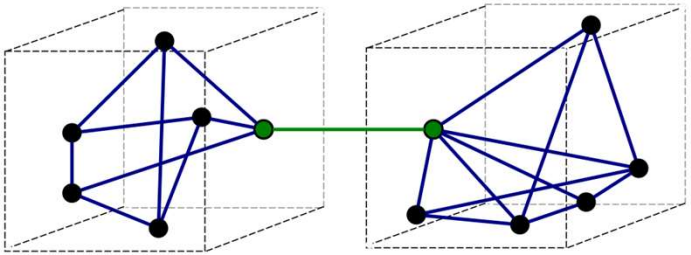
Enterprise Messaging

- API Management
- Enterprise Bus
- Pub / Sub
- Queues

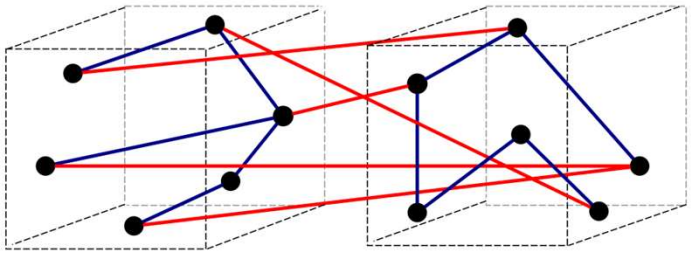


Coupling

- Coupling is the degree of interdependence between software modules



a) Good (loose coupling, high cohesion)

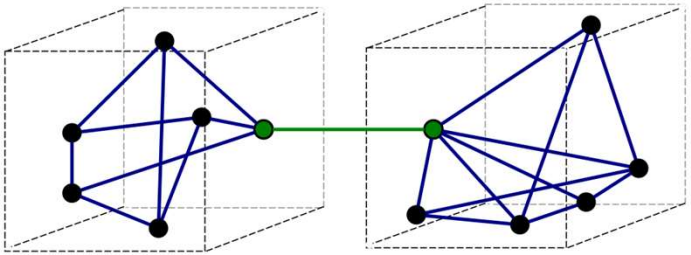


b) Bad (high coupling, low cohesion)

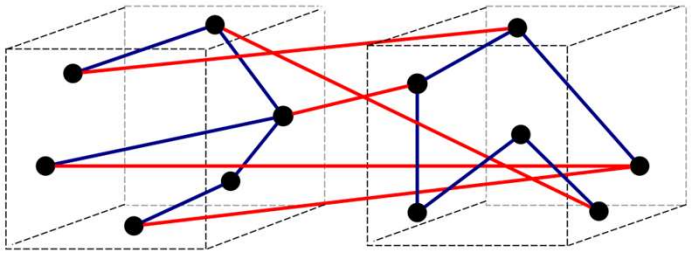
https://en.wikipedia.org/wiki/Coupling_(computer_programming)

Cohesion

- The degree to which elements inside a module belong together



a) Good (loose coupling, high cohesion)

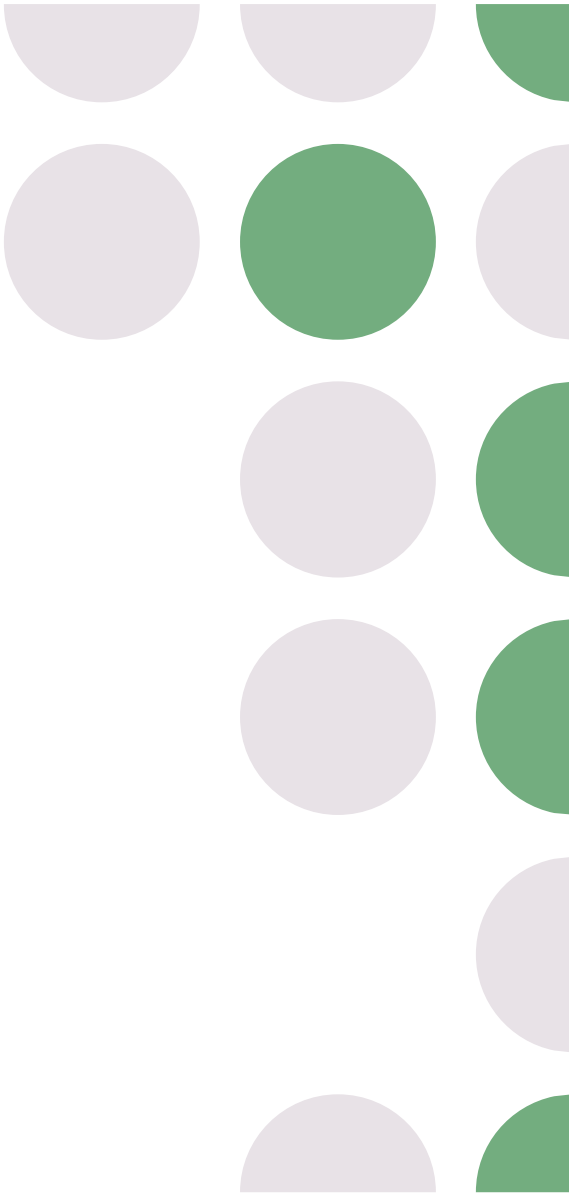


b) Bad (high coupling, low cohesion)

https://en.wikipedia.org/wiki/Cohesion_(computer_science)

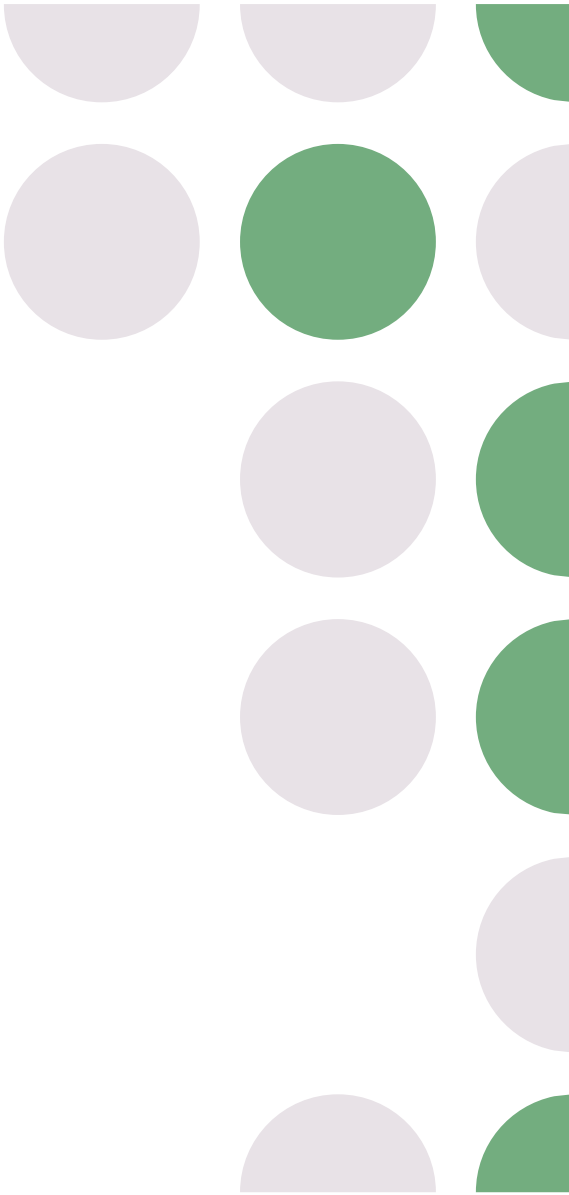
Logical vs. Physical Architecture

- Separate where your code logically belongs with the hosting and physical architecture of that code



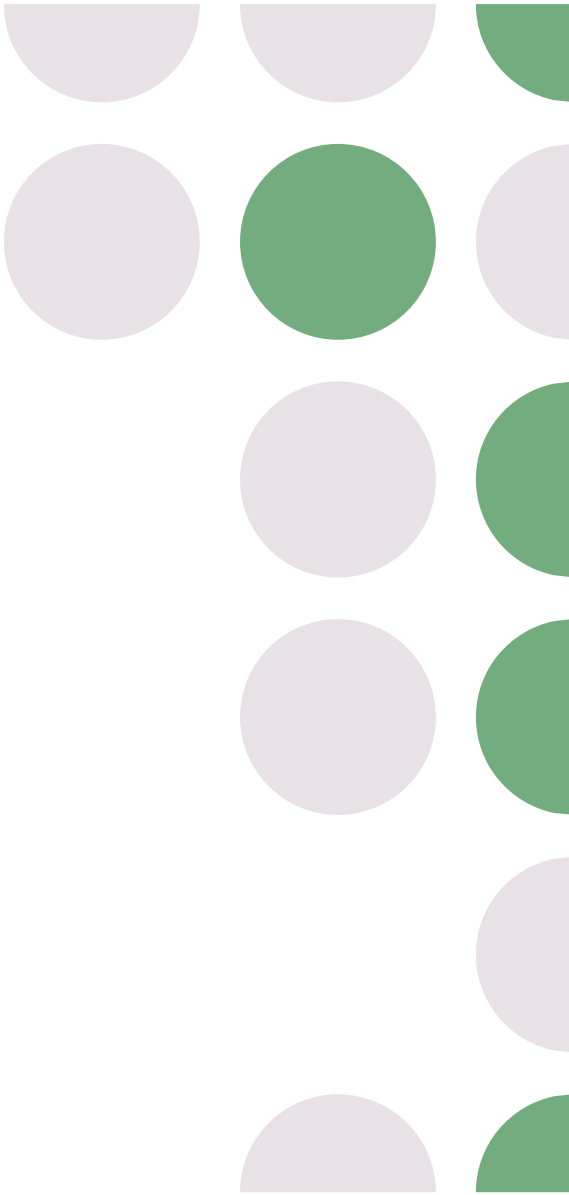
Layered Approach to Quality

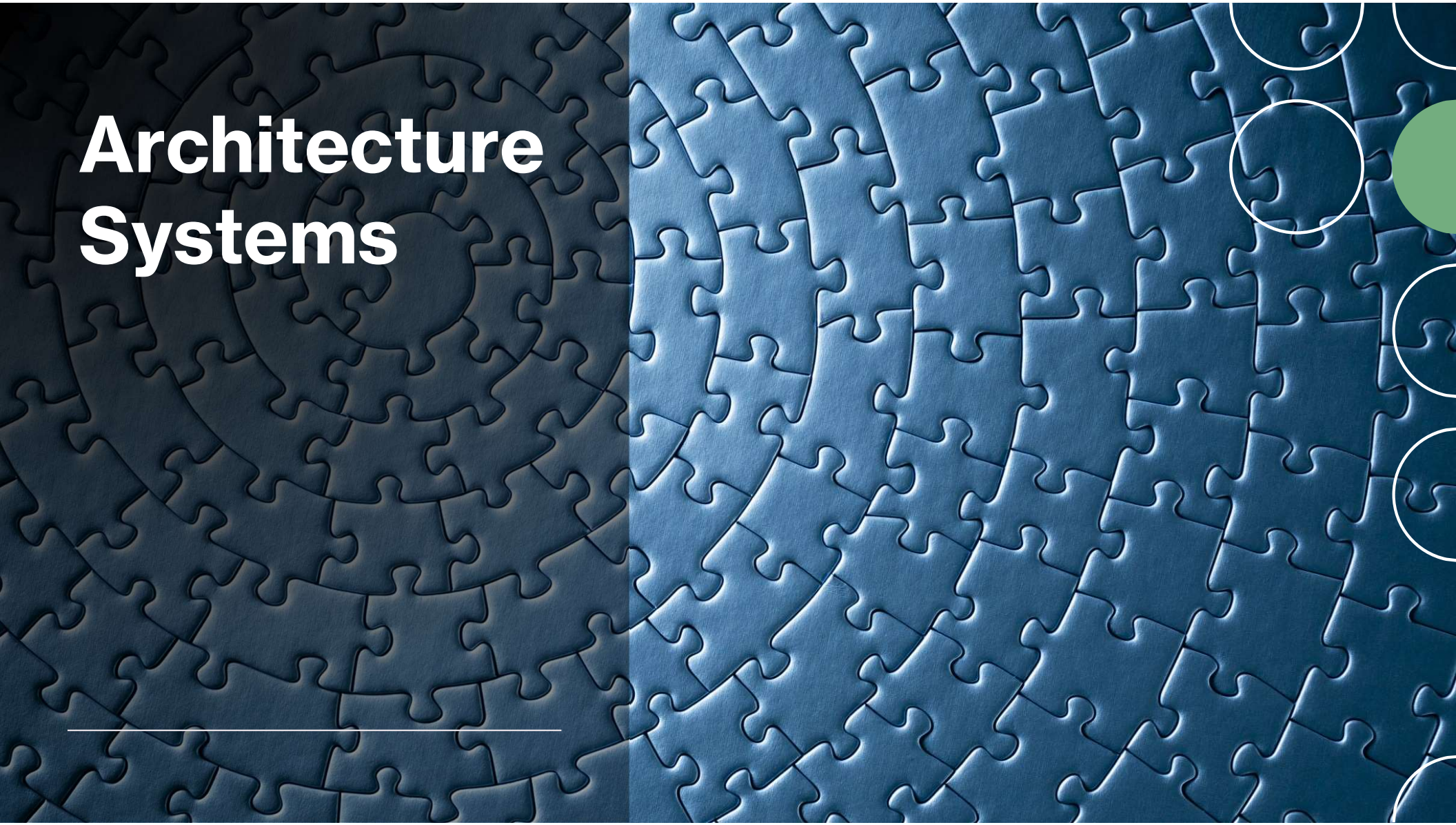
- There is no silver bullet for quality
  - Our System must support Unit, Integration, and other testing.



Other Design Patterns

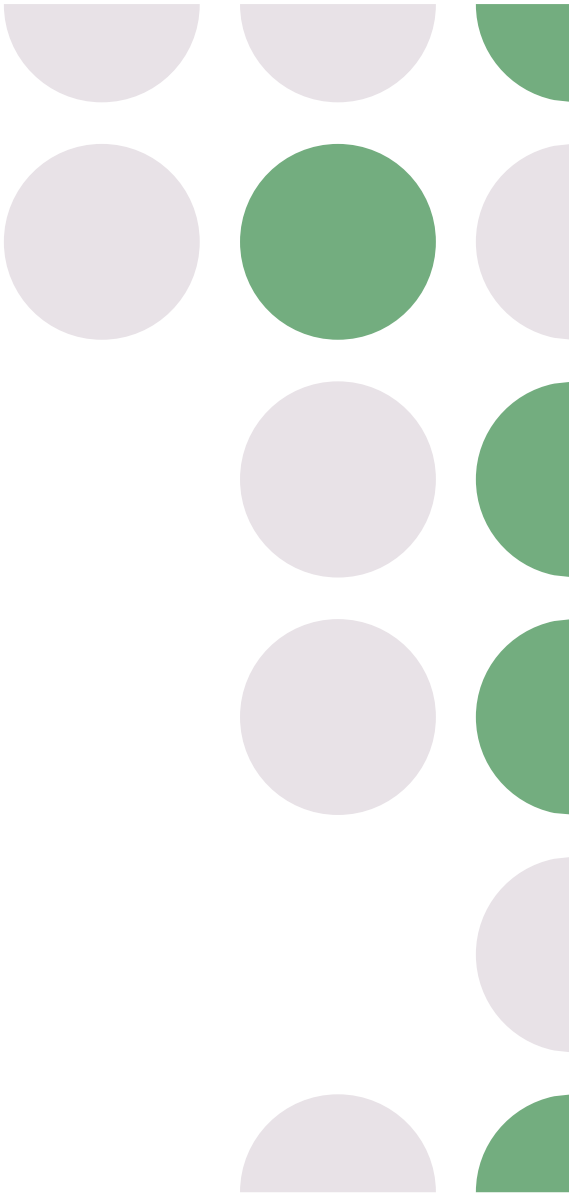
- Gang of 4 Design Patterns
  - https://springframework.guru/gang-of-four-design-patterns/
- Distributed Systems Patterns
  - https://martinfowler.com/articles/patterns-of-distributed-systems/




Architecture
Systems

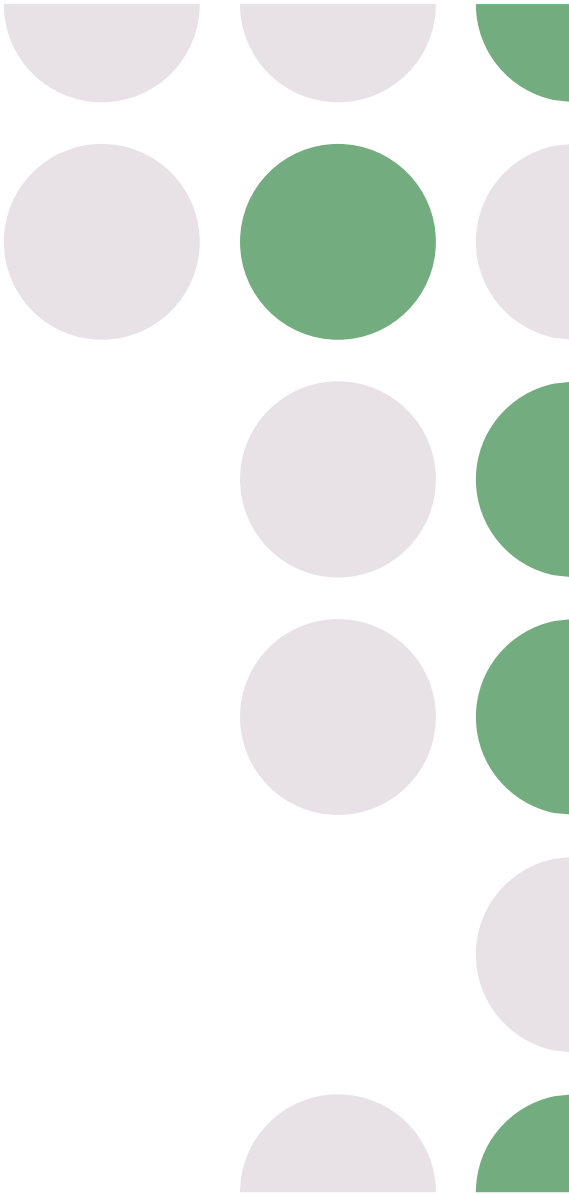
Microservices Architecture

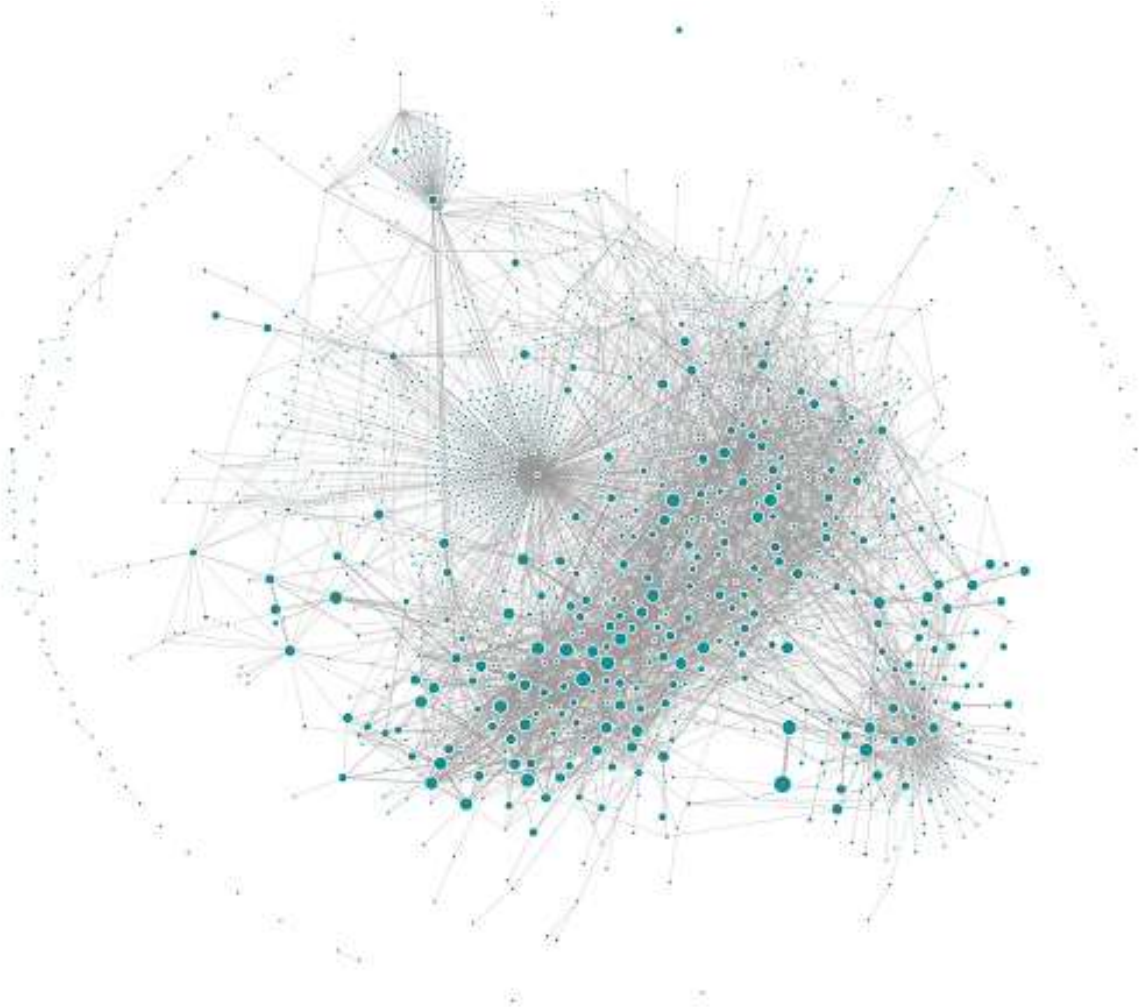
- Small, loosely coupled, independent services that manager their own data or state



Microservices Architecture

- Tradeoffs
  - Loose Coupling
  - Highly Testable
  - Ability to Scale Independently
  - Small blocks to deploy
  - Tracing and Debugging can be harder
  - Hard to constrain granularity
  - Difficult to distribute Workflow and Transactions





Uber’s Domain-Oriented Microservice Architecture

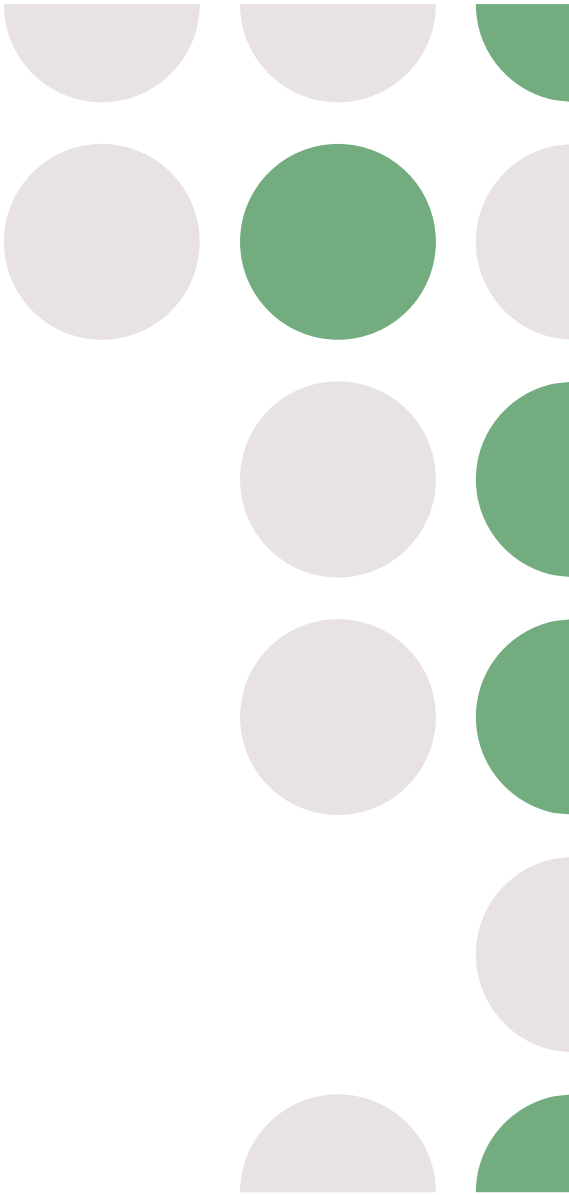
“For instance, engineers had to work through around 50 services across 12 different teams in order to investigate the root cause of the problem.”

---

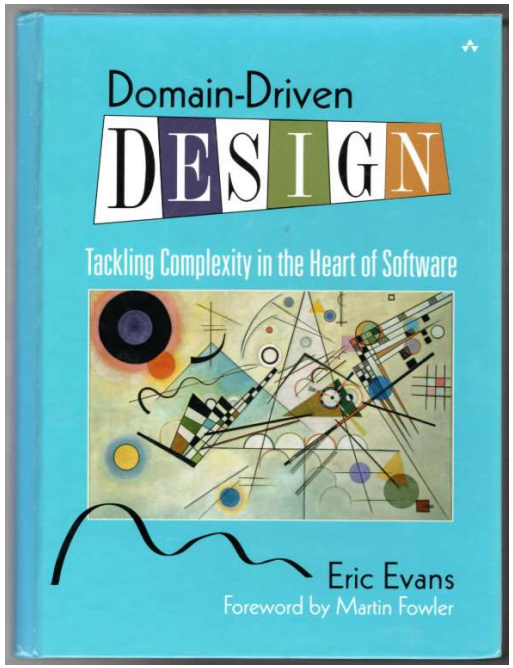
https://eng.uber.com/microservice-architecture/

Domain Driven Design

- Decomposing a software system into domains modeled after the domain itself

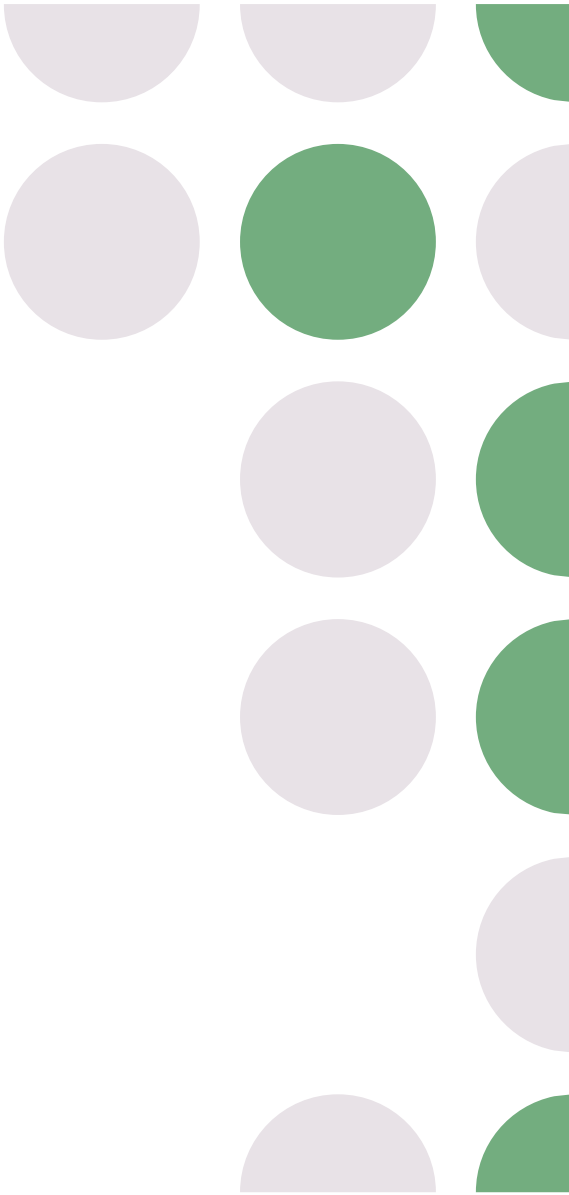


DDD by Eric Evans



Domain Driven Design

- Tradeoffs
  - Ubiquitous language between business and development
  - Flexible with change
  - Divides Systems into smaller domains
  - Requires strong Domain Expertise
  - Higher Costs
  - More suitable for larger projects

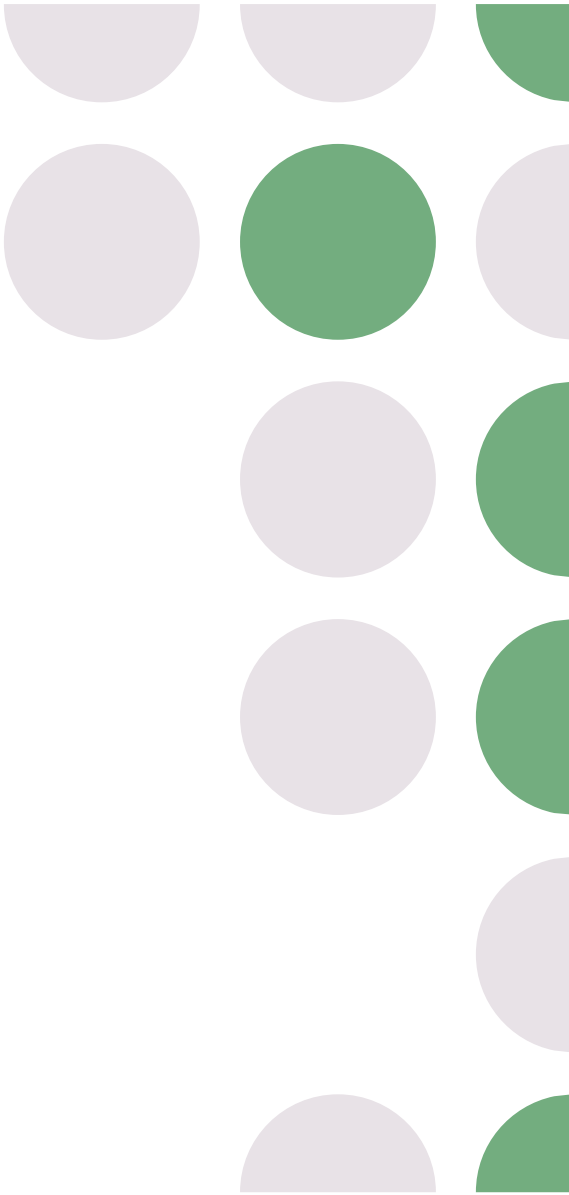


Modular Monoliths

“Microservices surged in popularity in recent years and were touted as the end-all solution to all of the problems arising from monoliths. Yet our own collective experience told us that there is no one size fits all best solution, and microservices would bring their own set of challenges.”

---

https://shopify.engineering/deconstructing-monolith-designing-software-maximizes-developer-productivity

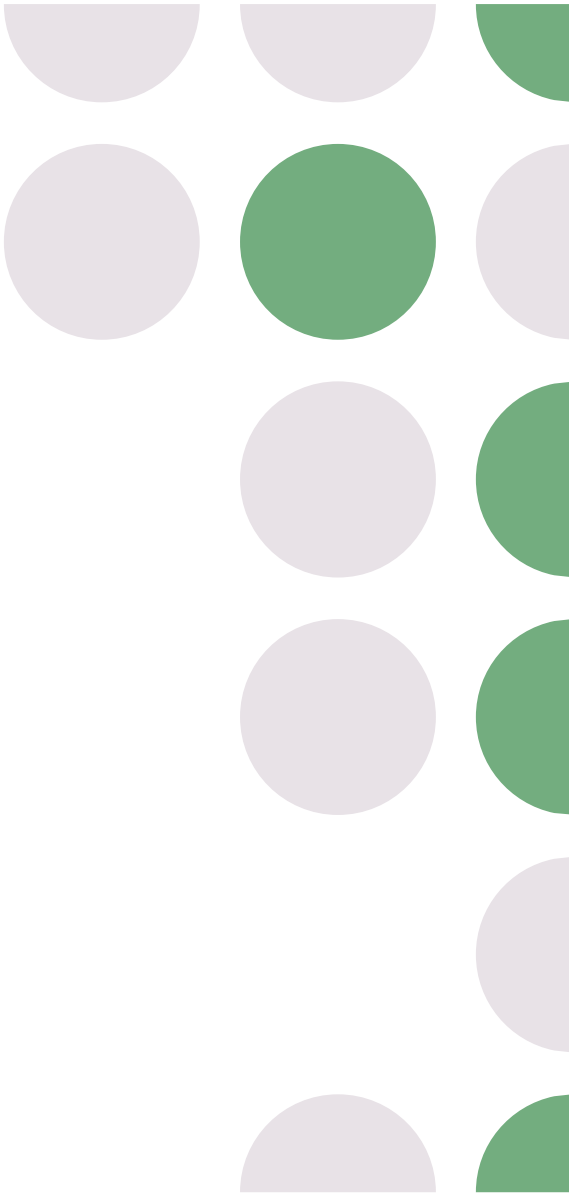


Volatility Based Decomposition

“We propose instead that one begin with a list of difficult design decisions or design decisions which are likely to change. Each module is then designed to hide such a decision from the others.”

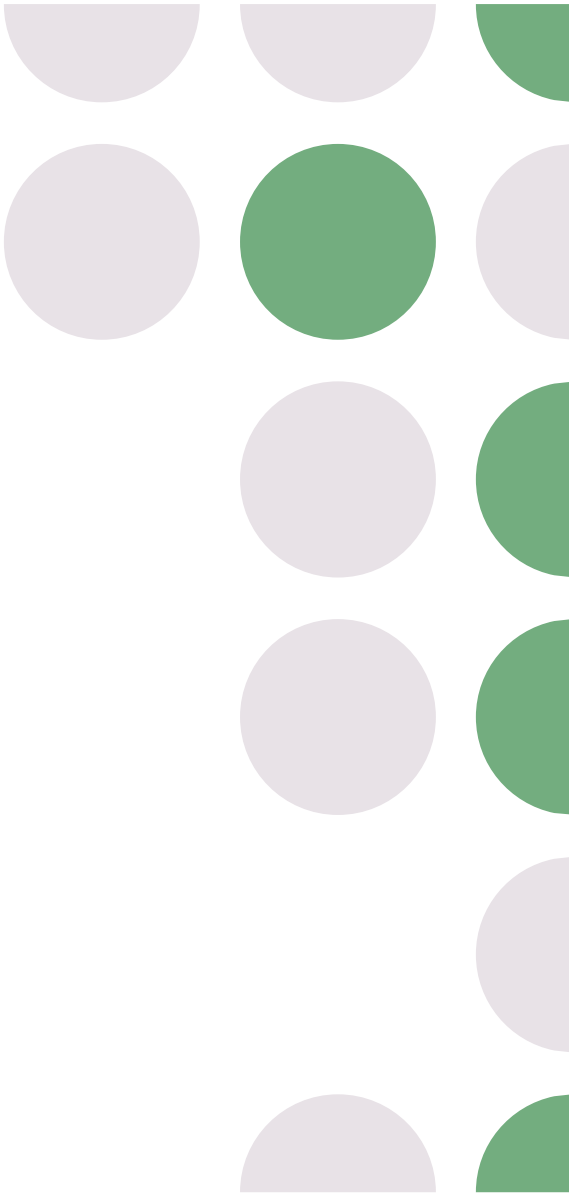
David Parnas

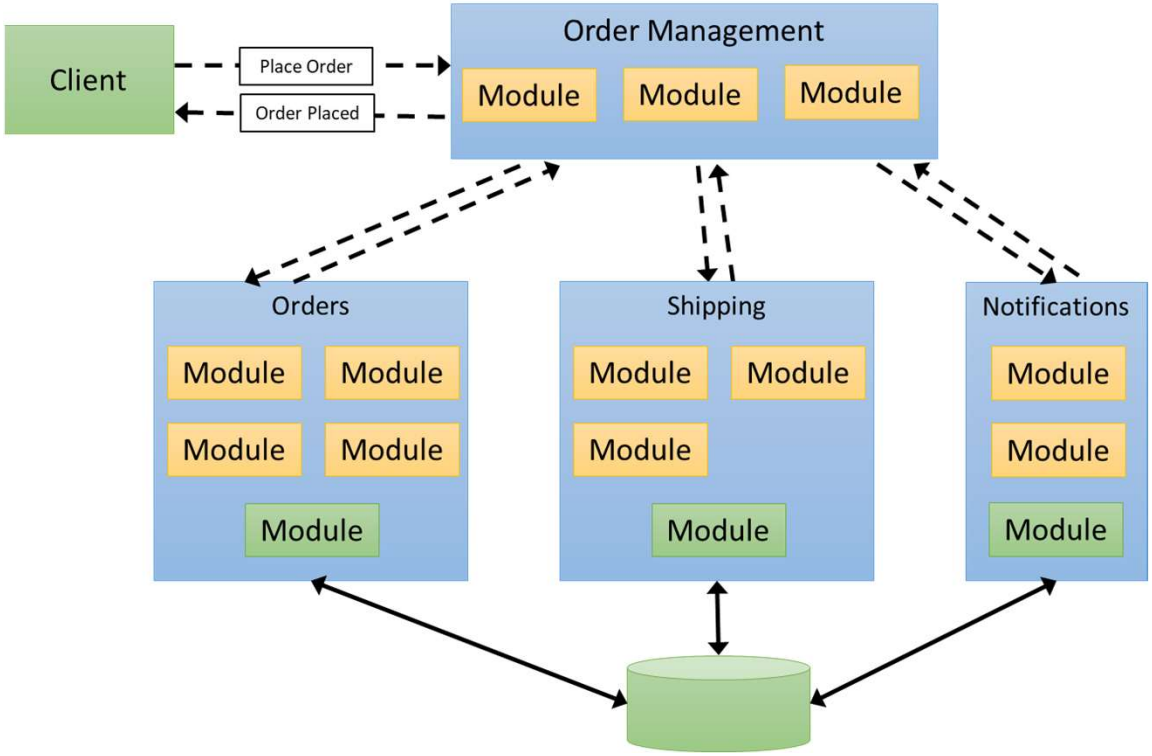
On the Criteria To Be Used in Decomposing Systems into Modules



Volatility Based Decomposition

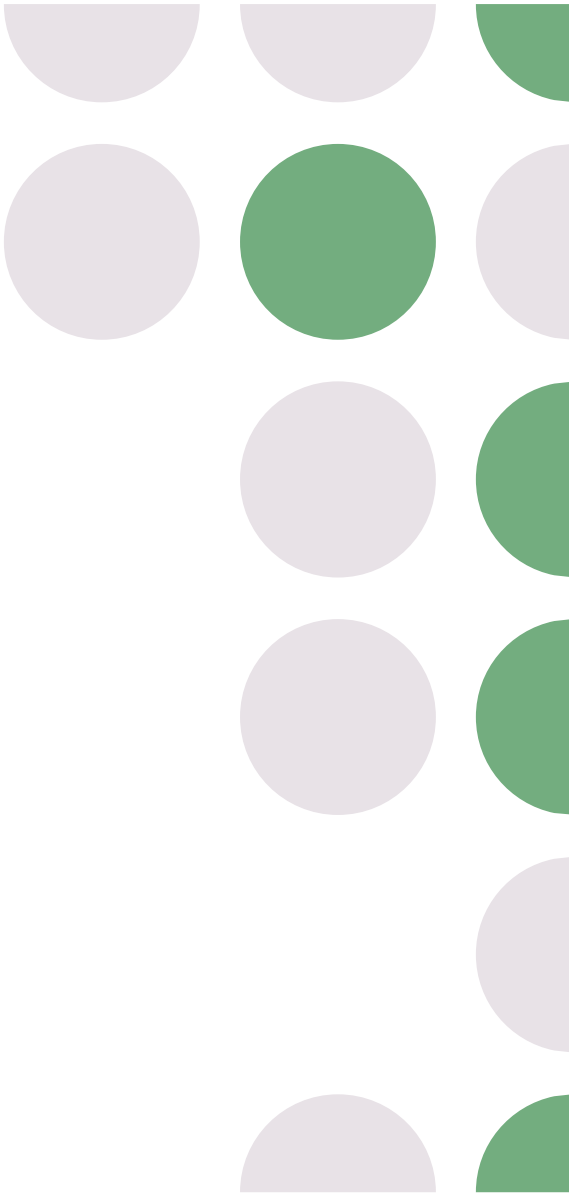
- Tradeoffs
  - Architected Sub-systems
    - Each system manages orchestration
    - Big changes happen inside sub-systems
  - Mostly Closed Architecture
  - Sweet spot of Service vs. Integration Cost
  - More regression testing of larger sub-systems
  - Less scaling options



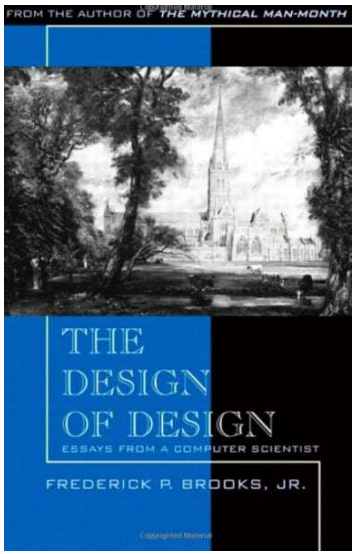

Client
Place Order
Order Placed
Order Management
Module
Module
Module
Orders
Module
Module
Module
Module
Module
Shipping
Module
Module
Module
Module
Notifications
Module
Module
Module

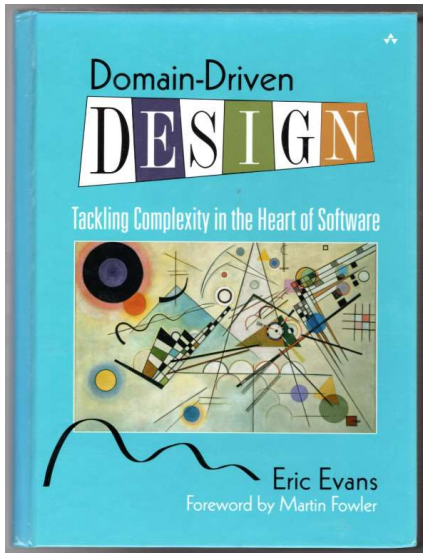
Opinions

- Favor these Tradeoffs
  - Upfront Design
  - Architecting services based on Volatility
  - Sub-systems or domains
  - Closed vs. Open Architecture
  - Orchestration vs. Choreography

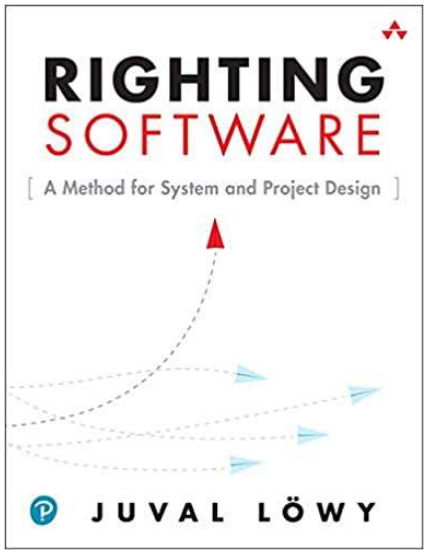


References

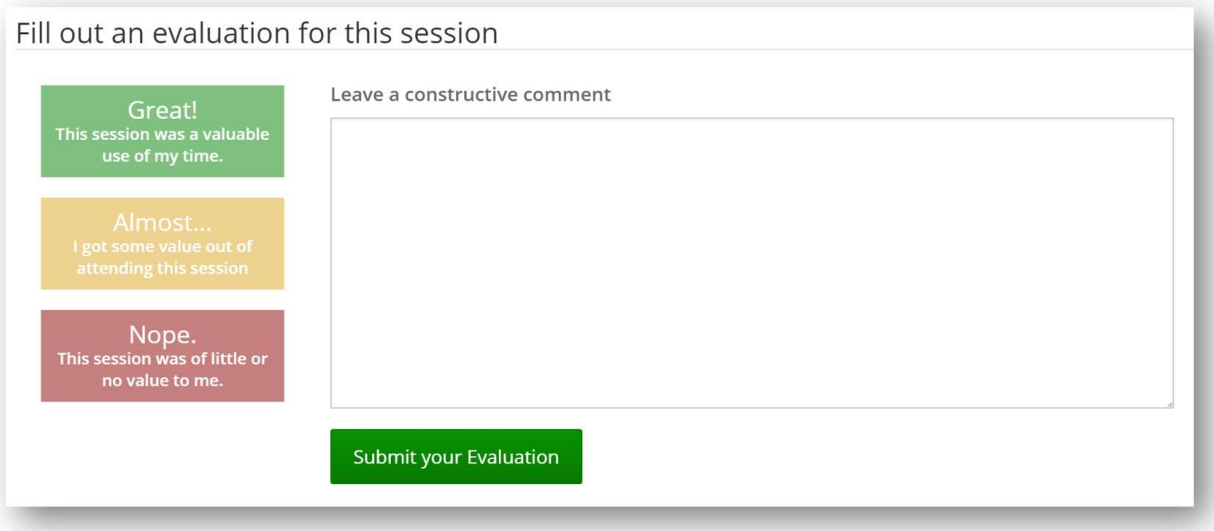

FROM THE AUTHOR OF THE MYTHICAL MAN-MONTH
THE DESIGN OF DESIGN
ESSAYS FROM A COMPUTER SCIENTIST
FREDERICK P. BROOKS, JR.


Domain-Driven
DESIGN
Tackling Complexity in the Heart of Software
Eric Evans
Foreword by Martin Fowler


Robert C. Martin Series
Clean Architecture
A Craftsman's Guide to Software Structure and Design
Robert C. Martin


RIGHTING SOFTWARE
[A Method for System and Project Design]
JUVAL LÖWY

Evaluations



NebraskaCode.amegala.com/Schedule

Contact Info

- @unter
- andy@dontpaniclabs.com
- https://github.com/unter/Presentations