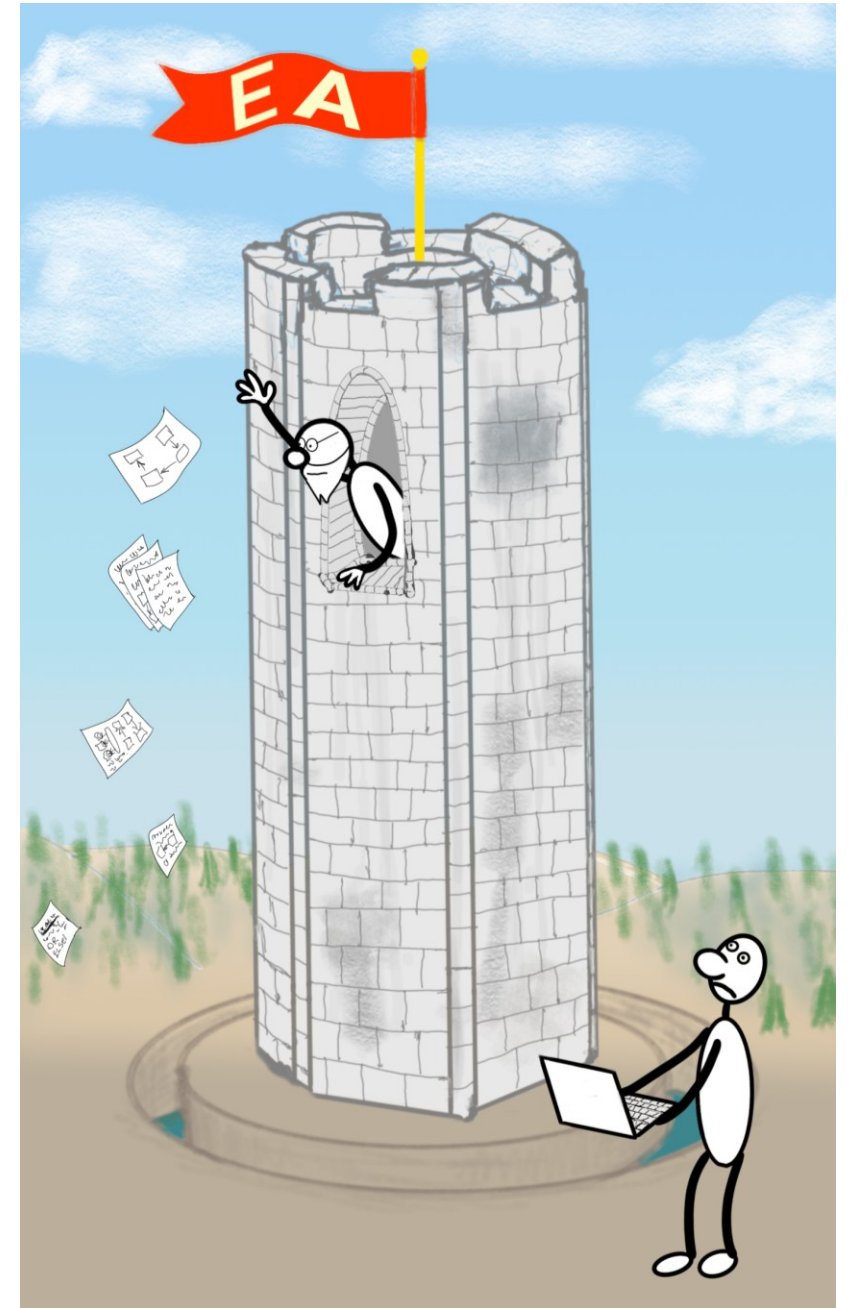


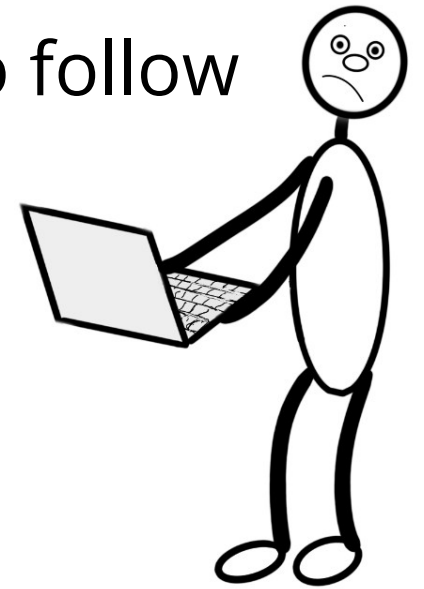
Escaping Architecture Ivory Tower with Architecture Pattern Catalog

dev2next
Colorado Springs, CO
September 29 - October 2, 2025



What Is Architecture?

Unsatisfying attempts at defining Architecture to follow



What Is Architecture? [Wikipedia]

Software architecture is the set of structures needed to reason about a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations.

What Is Architecture? [Martin Fowler]

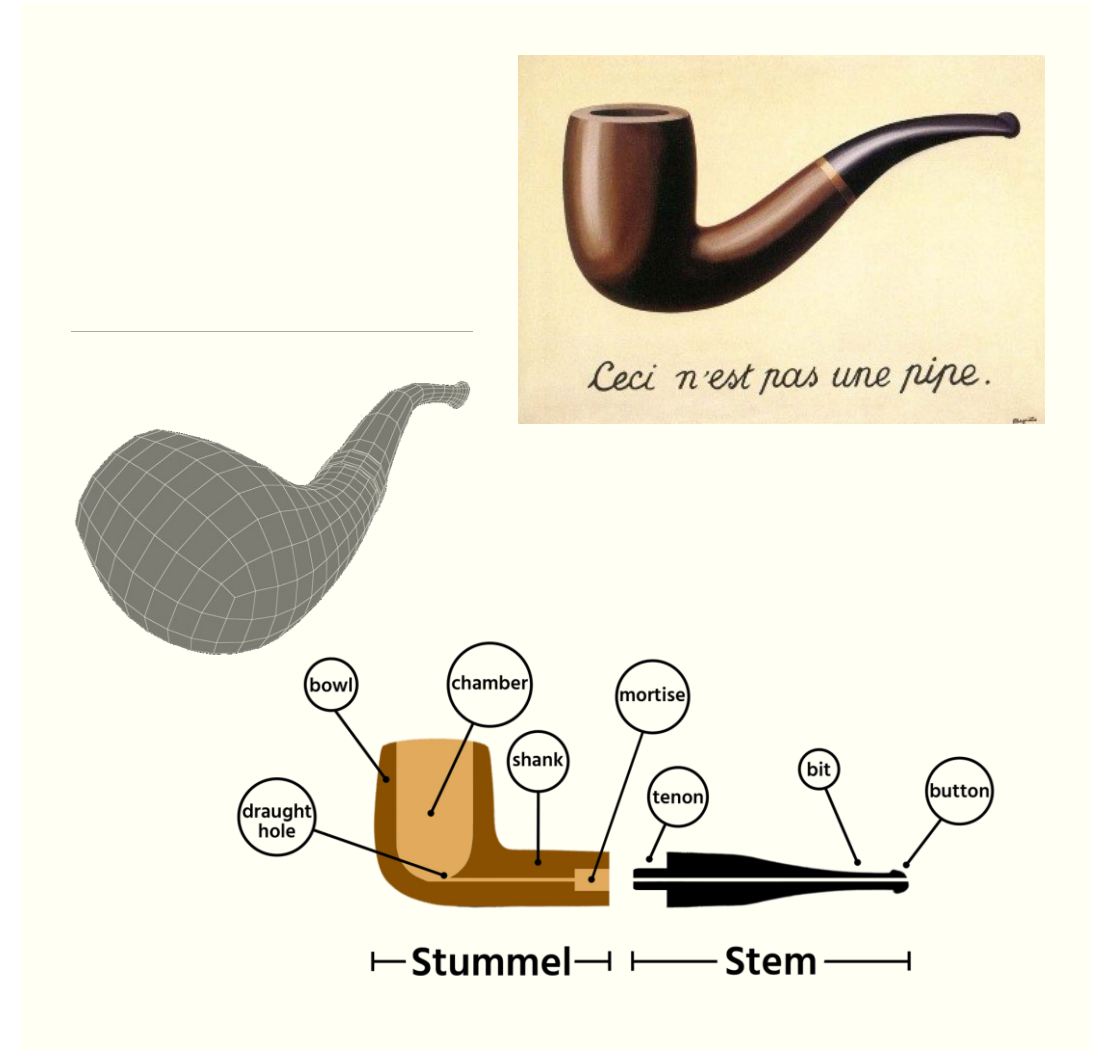
[A] hazily defined notion of the most important aspects of the internal design of a software system.

What Is Architecture? [Grady Booch]

- “Architecture is just a collective hunch, a shared hallucination, an assertion by a set of stakeholders on the nature of their observable world, be it a world that is or a world as they wish it to be”
- Represented by a set of interlocking models
- Architecture captures
 - Significant design decisions
 - Patterns
 - Cross cutting concerns
 - Rationale
 - Tribal memory

What They Say About Models

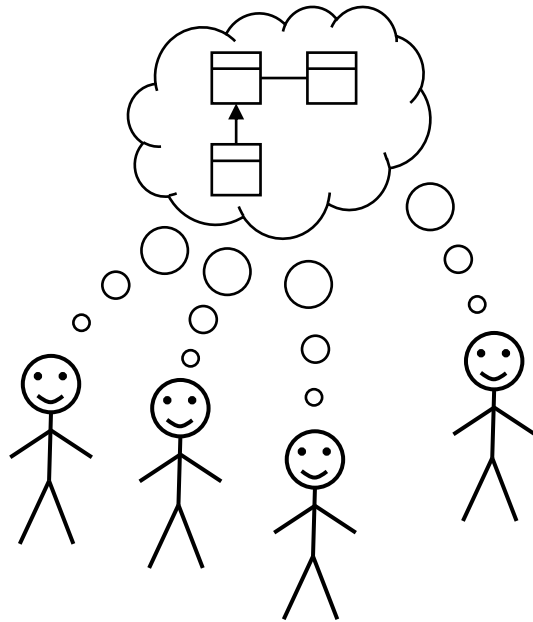
- All models are wrong, but some are useful
- Achieve amplification through simplification
- Comprehensiveness is the enemy of comprehensibility



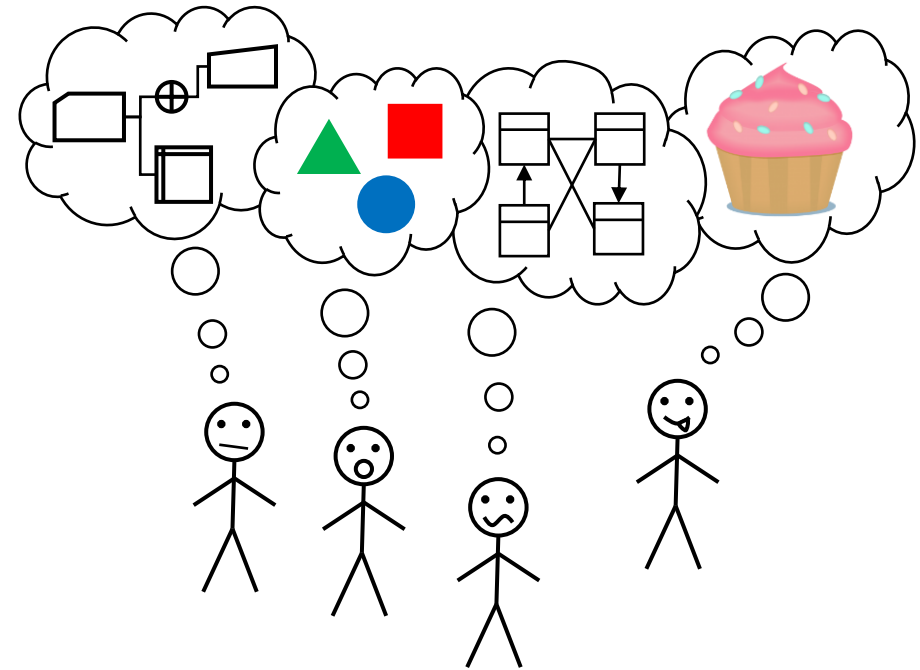
“...a set of interlocking models...”

There are always models. If there is no explicit modeling activity, there will be many implicit models, likely incorrect and inconsistent

Explicit Modeling:



Implicit (No) Modeling:



What Is Architecture? [Fred Brooks]

A clean, elegant programming product must present to each of its users a **coherent mental model** of the application, of strategies for doing the application, and of the user-interface tactics to be used

...

Conceptual Integrity is the most important consideration in system design. It is better to have a system omit certain anomalous features and improvements, but to reflect one set of design ideas, than to have one that contains many good but independent and uncoordinated ideas.

What Is Architecture? [Ralph Johnson]

In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.'

...

architecture is a social construct

...

Whether something is part of the architecture is entirely based on whether the developers think it is important.

...

Architecture is about the important stuff. Whatever that is.

So, What Is Architecture?

No good definition given—promise kept!



What Is Architect?

- A facilitator—of no value if all alone all by themselves
- Ensures Conceptual Integrity
- “A keeper of the Project Vision”
- Follows a defined process for organizing our thinking in a way that allows to solve systemic as well as discrete problems
- Curates a repository of information that is useful to other participants
 - Produces artifacts in code and in other mediums
 - Designs knowledge flow
- “guide, as in mountaineering” [Fowler]
- A storyteller

Why Is Architect?

- The storytellers in a tribe tell stories
- Stories explain why the world the way it is
- Running software is the ultimate truth of a software project
- ...but it lacks creation stories (“why it is the way it is”)
- Some stories can be inferred from the code
- The rest can be in tribal memory living in heads of storytellers
- Relying on informal storytelling to share important decisions is costly
- When the stories are gone, recreating those decisions is very costly
- Codifying tribal memory makes evolving systems materially easier

Where Is Architect?



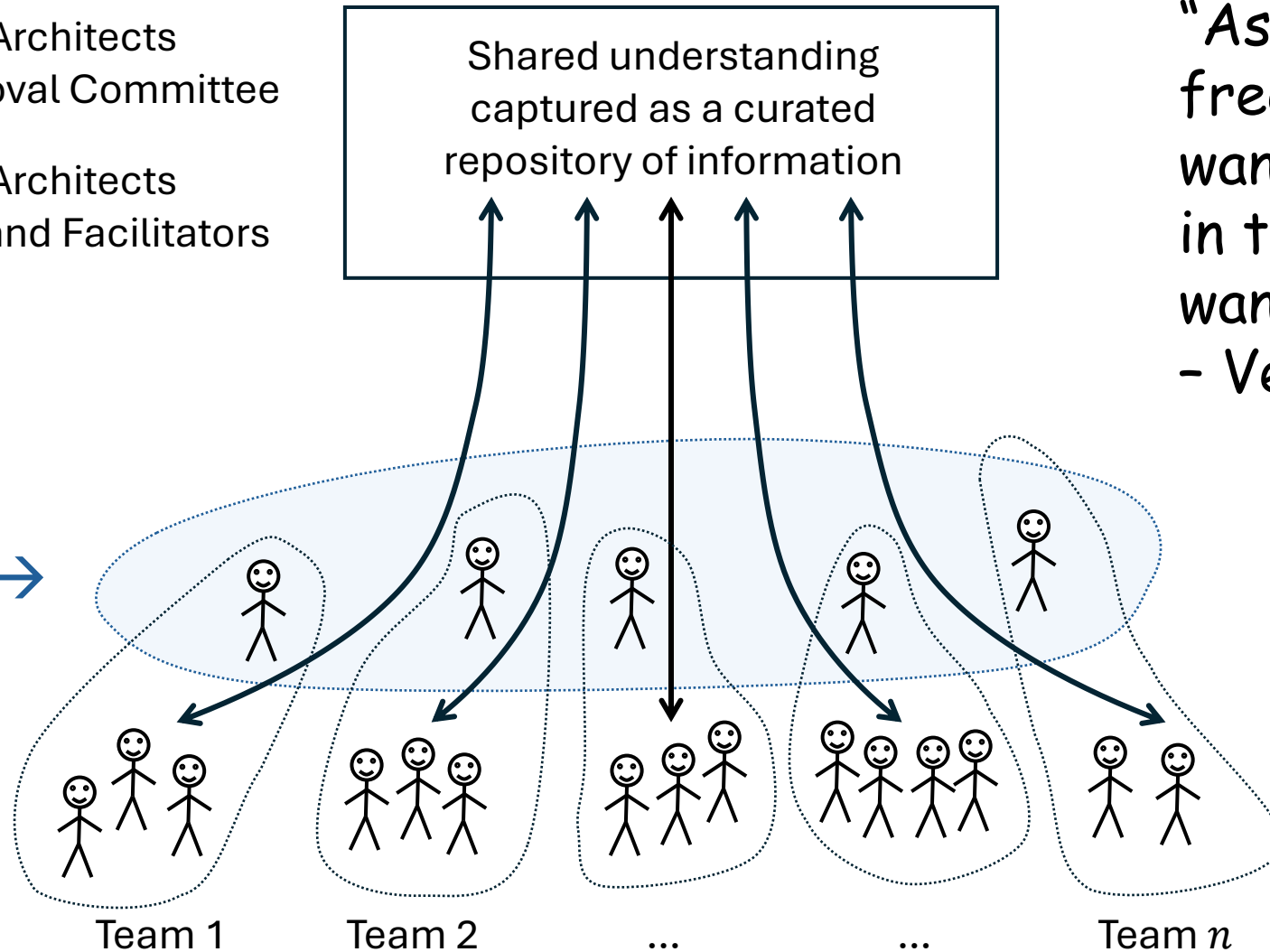
Enterprise Architects
as an Approval Committee



Enterprise Architects
as Guides and Facilitators

"Ask early and frequently if you want feedback. Ask in the end if you want blessings"
- Venkat S.

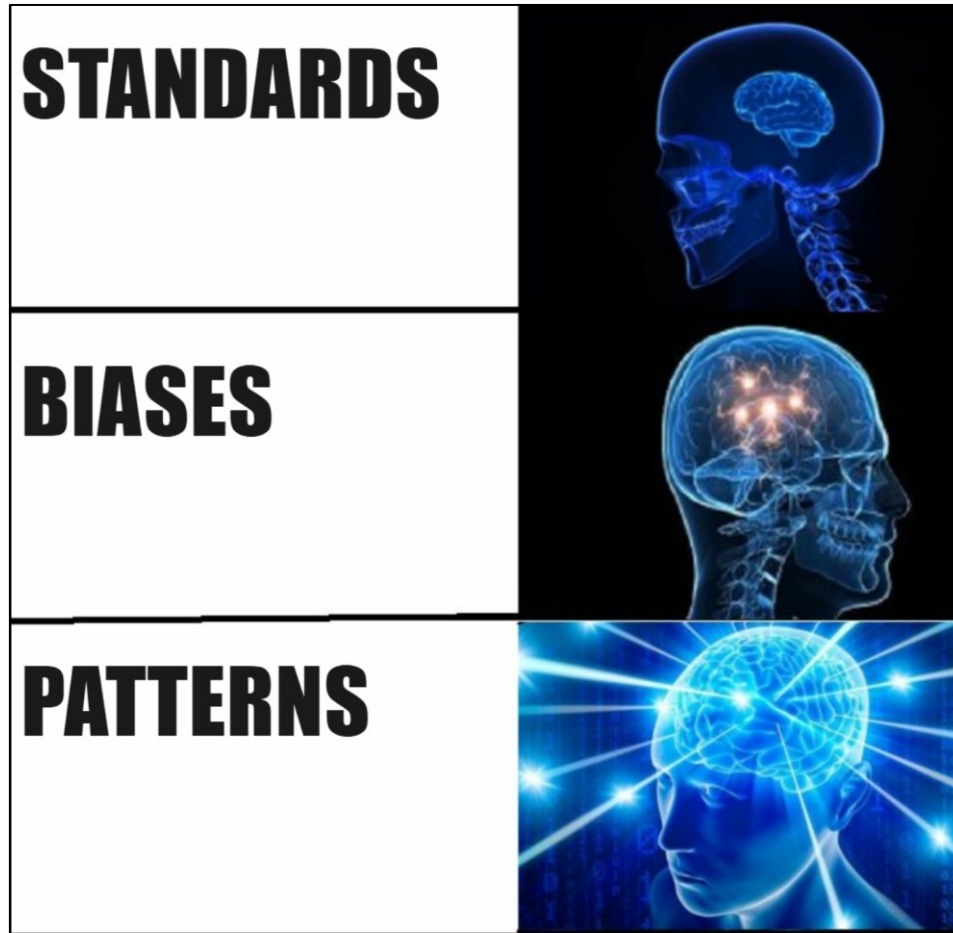
Architects →



Benefits of Shared Understanding

- Architectural/Conceptual Integrity across Organization
- Collectively owned
- Goal: same solutions for same problems in same contexts
- To evolve systems, we need to understand the trade-offs and the context and the compromises
 - See: https://en.wikipedia.org/wiki/G._K._Chesterton#Chesterton's_fence
- Write down the reasons for doing things
 - What is the rationale behind the design decisions?
 - Why we chose this, why we didn't choose this?
- “The code is the truth, but the code is not the whole truth”

How to Capture Shared Understanding



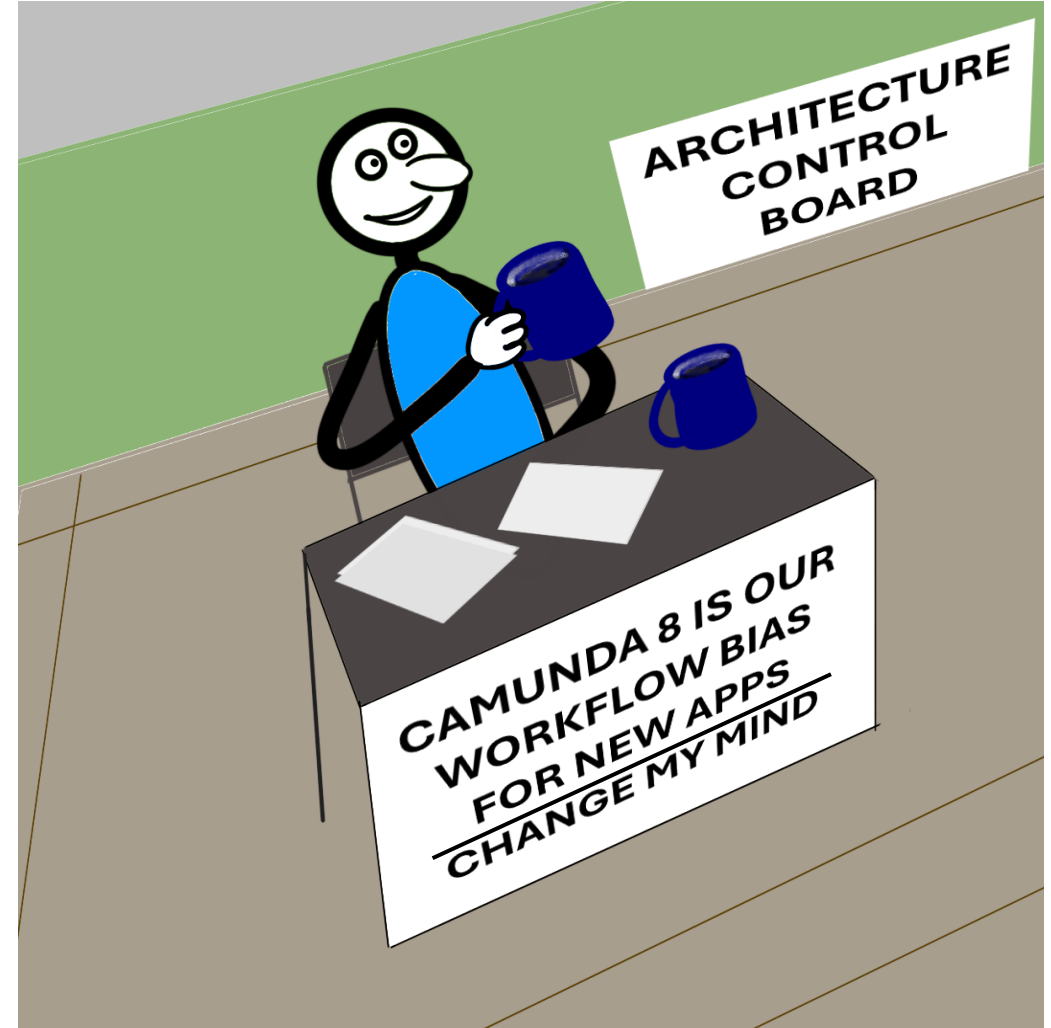
Standards

- A list of “approved” software things, e.g.:
 - Camunda 5.1
 - Kafka 3.2
 - Drools 8.43
 - ...
- Approved for what? Why? Why not? Unclear
- Software is fungible and many of tools are Turing complete, so:
 - Kafka is a database
 - A workflow engine is a rules engine
 - A rules engine is a workflow engine
- Standardization can lead to stagnation
 - What is the motivation for changing a standard?



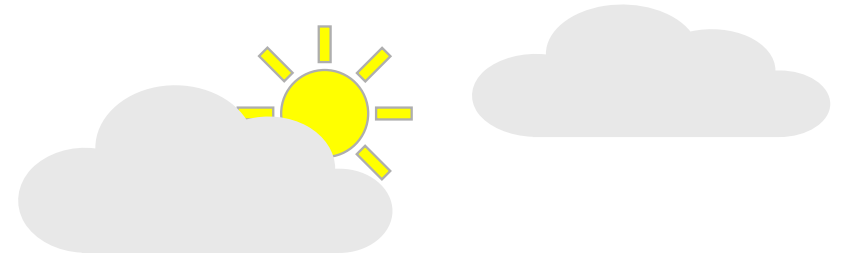
Biases – Standards Refined

- “Our bias is X. Why will it not work for your requirements?”
- Introduces a notion of context
 - Functional
 - Workflow
 - Database
 - etc.
 - Lifecycle
 - Legacy apps
 - Maintained apps
 - New development



Problems With Standards and Biases

- A catalog of standards or biases navigates from a solution
- We want to navigate from a problem
- ...in a context
- ...to a solution
- ...understanding tradeoffs
- ...specific to our organization
- ...with working examples to inspire and reuse
- ...and pointers to related problems and their solutions



Solution?



What Are Patterns?

"Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves." - *Richard Gabriel*

"A pattern language defines a collection of patterns and the rules to combine them into an architectural style. Pattern languages describe software frameworks or families of related systems."

Why Do Patterns Make Sense?

- Patterns capture solutions, not just abstract principles or strategies
- Each specific solution tries to resolve the forces acting upon it
- Each engineering problem has an associated solution space
- Solution space is limited
 - Laws of physics
 - Business issues (cost, skillset, etc.)
- Patterns capture solutions with a track record, not theories or speculation
- Enough systems have been built to observe that a [relatively] small number of patterns exist in each given domain that enumerate all [or most] suitable solutions

⇒ *Patterns describe structures that are useful, useable, and used!*

Pattern Examples

Category	Pattern
Messaging	Publisher/Subscriber
	Guaranteed Messaging
	Queue with Keys (Sequential Convoy)
	Dead Letter Queue
Storage	Relational Database
	Key-Value Store
	Two-Dimensional Key-Value Store
	Cache
Resiliency	Bulkhead
	Stateless Services
	Horizontal Scalability
Data	Bi-temporal Milestoning
	High Water Mark Filters

Pattern Description Template

Category
Sub-category

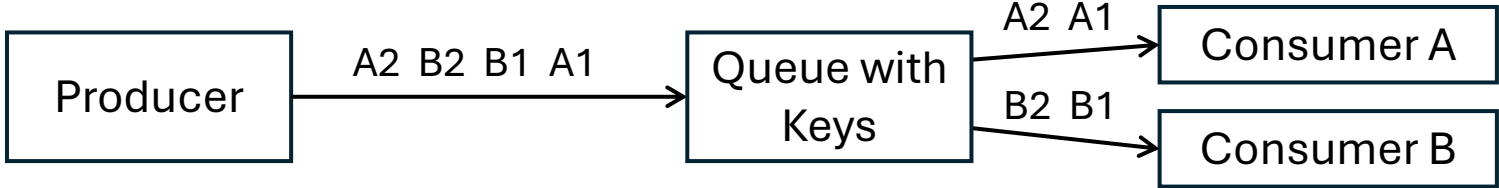
Pattern: [Pattern Name - a descriptive and unique name]

Description	A description of the goal behind the pattern and the reason for using it
Motivation	A scenario consisting of a problem and a context in which this pattern can be used
Applicability	Situations in which this pattern is usable; the context for the pattern
Structure	A graphical representation of the pattern
Consequences	A description of the results, side effects, and trade offs caused by using the pattern
Implementation	A description or an example of an implementation of the pattern
Related Patterns	Other patterns that have some relationship with the pattern
Known uses	Examples of real usages of the pattern

Pattern Description Example

Messaging

Pattern: Sequential Convoy [from MSFT]

Description	Process related messages in a defined order, without blocking processing of other messages
Motivation	Maintain FIFO order within subsets of messages, while still being able to scale out to handle load
Applicability	You have messages that arrive in order and must be processed in the same order. Arriving messages can be "categorized" in such a way that the category becomes a unit of scale for the system.
Structure	 <pre>graph LR; Producer[Producer] -- "A2 B2 B1 A1" --> Queue[Queue with Keys]; Queue -- "A2 A1" --> ConsumerA[Consumer A]; Queue -- "B2 B1" --> ConsumerB[Consumer B]</pre> <p>The diagram illustrates the Sequential Convoy pattern structure. A 'Producer' box sends a sequence of messages 'A2 B2 B1 A1' to a 'Queue with Keys' box. The queue then routes these messages to two consumers: 'Consumer A' and 'Consumer B'. Messages with key 'A' (A2, A1) are sent to Consumer A, and messages with key 'B' (B2, B1) are sent to Consumer B, ensuring each consumer processes its subset of messages in order.</p>
Consequences	Scale out by Category; Category sizes may differ by a lot; throughput limitations; evolving categories; defining category (session) boundaries
Implementation	Azure Service Bus Message Sessions
Related Patterns	Guaranteed Messaging, Competing Consumers
Known uses	Intraday P&L Accounting System [links, contact info]

Pattern Misuse

- Causes
 - Exuberance when patterns are discovered
 - Not understanding or ignoring the context
 - Not keeping the system simple
 - “Anticipatory Design”
 - Ambiguous definition of a pattern
- Treatment
 - Only use a pattern if
 - you have the problem it solves
 - the positives outweigh the negatives
 - Ensure that the entire team understands the patterns in use
 - Oversight and Communication – Architecture Function Involvement



Where Do Patterns Come From?

- Harvested from within the organization
 - Collected from outside the organization
 - Many hard problems have been solved, solutions have been codified
 - Cloud vendors
 - Harvested solutions implemented on their respective clouds
 - Published pattern catalogs
 - Defined different architecture pattern languages
 - Sign of maturity: Algorithms → Design Patterns → Architectural Patterns
 - Cloud Architecture Patterns are Architecture Patterns
- ⇒ Collect, select, adapt, and curate*

Pattern Catalog Examples

- Microsoft

<https://learn.microsoft.com/en-us/azure/architecture/patterns/>

- Amazon

<https://docs.aws.amazon.com/prescriptive-guidance/latest/cloud-design-patterns/introduction.html>

<https://aws.amazon.com/blogs/compute/understanding-asynchronous-messaging-for-microservices/>

- Google

<https://cloud.google.com/architecture>

Use Case: Stock Trading Accounting

A stock brokerage

...buys/sells stocks

...across many accounts

...using FIFO accounting

...calculates position quantities and P&L

...used for real time reporting and compliance



Need to implement a system that captures trades and calculates quantities and P&L in real time for each trade and for each account

Disclaimer

Nothing that follows is accounting advice, nor legal advice,
nor technology implementation advice

Always consult your accountant, lawyer, engineer
as appropriate

Stock Trading Scenario

Time	Account	Sec	Buy/Sell	Qty	Price	Cost Basis	Trade Profit/Loss	Sec Qty	Realized Profit/Loss
9:30 AM	123	MSFT	Buy	100	\$ 450.00		\$ -	100	\$ -
9:35 AM	123	IBM	Buy	40	\$ 240.00		\$ -	40	\$ -
10:10 AM	123	MSFT	Buy	50	\$ 480.00		\$ -	150	\$ -
10:15 AM	123	MSFT	Sell	-50	\$ 475.00	\$ 450.00	\$ 1,250.00	100	\$ 1,250.00
10:40 AM	123	IBM	Buy	60	\$ 260.00		\$ -	100	\$ 1,250.00
11:30 AM	123	MSFT	Sell	-50	\$ 445.00	\$ 450.00	\$ (250.00)	50	\$ 1,000.00
11:45 AM	123	IBM	Sell	-20	\$ 250.00	\$ 240.00	\$ 200.00	20	\$ 1,200.00

Why FIFO Matters

Time	Account	Sec	Buy/Sell	Qty	Price	Cost Basis	Trade Profit/Loss	Sec Qty	Realized Profit/Loss
9:30 AM	123	MSFT	Buy	100	\$ 450.00		\$ -	100	\$ -
9:35 AM	123	IBM	Buy	40	\$ 240.00		\$ -	40	\$ -
10:10 AM	123	MSFT	Buy	50	\$ 480.00		\$ -	150	\$ -
10:15 AM	123	MSFT	Sell	-50	\$ 475.00	\$ 480.00	\$ (250.00)	100	\$ (250.00)
10:40 AM	123	IBM	Buy	60	\$ 260.00		\$ -	100	\$ (250.00)
11:30 AM	123	MSFT	Sell	-50	\$ 445.00	\$ 450.00	\$ (250.00)	50	\$ (500.00)
11:45 AM	123	IBM	Sell	-20	\$ 250.00	\$ 240.00	\$ 200.00	20	\$ (300.00)

Why FIFO Matters

Time	Account	Sec	Buy/Sell	Qty	Price	Cost Basis	Trade Profit/Loss	Sec Qty	Realized Profit/Loss
9:30 AM	123	MSFT	Buy	100	\$ 450.00		\$ -	100	\$ -
9:35 AM	123	IBM	Buy	40	\$ 240.00		\$ -	40	\$ -
10:10 AM	123	MSFT	Buy	50	\$ 480.00		\$ -	150	\$ -
10:15 AM	123	MSFT	Sell	-50	\$ 475.00	\$ 480.00	\$ (250.00)	100	\$ (250.00)
10:40 AM	123	IBM	Buy	60	\$ 260.00		\$ -	100	\$ (250.00)
11:30 AM	123	MSFT	Sell	-50	\$ 445.00	\$ 450.00	\$ (250.00)	50	\$ (500.00)
11:45 AM	123	IBM	Sell	-20	\$ 150.00	\$ 180.00	\$ 200.00	20	\$ (300.00)

BIG SAVINGS!

Why FIFO Matters



Time	Account	Sec				Loss	Sec Qty	Realized Profit/Loss
9:30 AM	123	MSFT				-	100	\$ -
9:35 AM	123	IBM				-	40	\$ -
10:10 AM	123	MSFT	Buy	50	\$ 480.00	-	150	\$ -
10:15 AM	123	MSFT	Sell	-50	\$ 475.00	\$ 480.00	100	\$ (250.00)
10:40 AM	123	IBM	Buy	60	\$ 260.00	-	100	\$ (250.00)
11:30 AM	123	MSFT	Sell	50	\$ 445.00	\$ 450.00	50	\$ (500.00)
11:45 AM	123	IBM	Sell	20	\$ 15.00	\$ 200.00	20	\$ (300.00)

BIG SAVINGS!

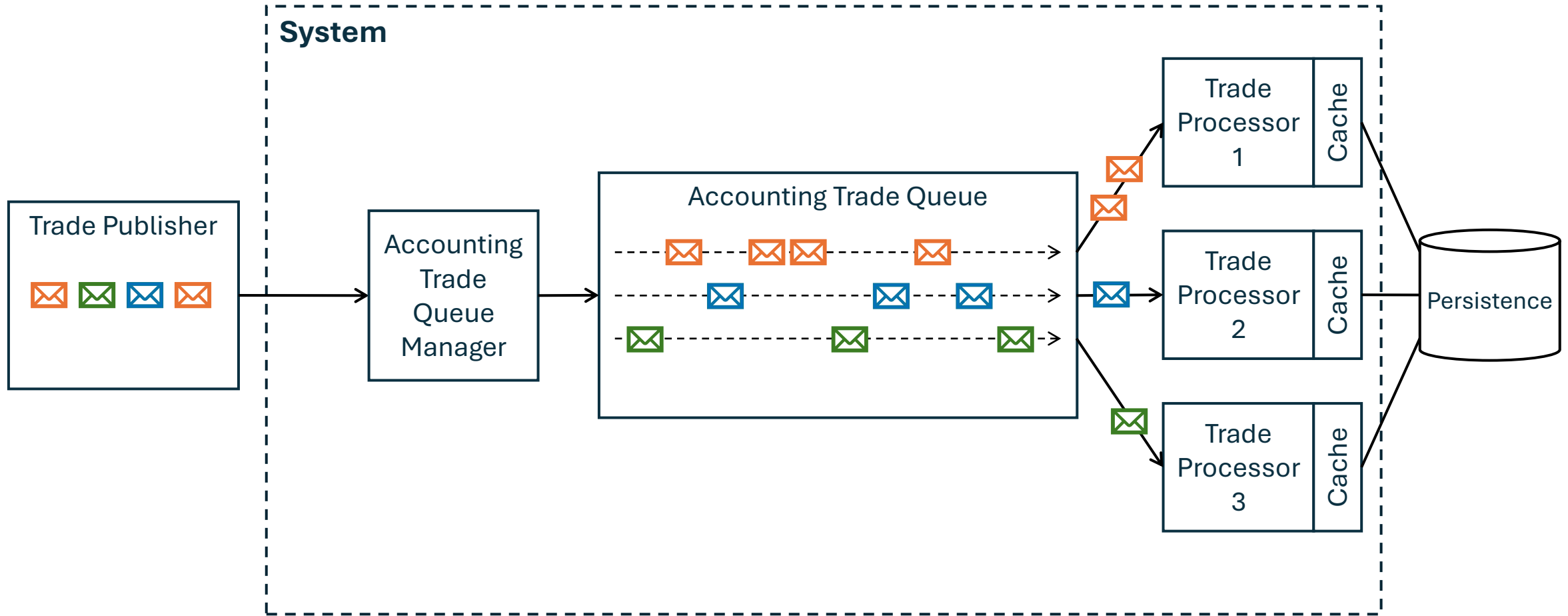
Requirements

- Functional
 - FIFO trade processing for Account/Security combination
 - Can't lose a trade
 - Real time processing for reporting and for portfolio compliance
- Non-functional
 - Cost Efficiency
 - Commodity hardware
 - Horizontal Scalability/Elasticity
 - Reliability/Recoverability
 - Volumes
 - 100,000 accounts
 - 8,000 stocks
 - 10,000 trades a minute

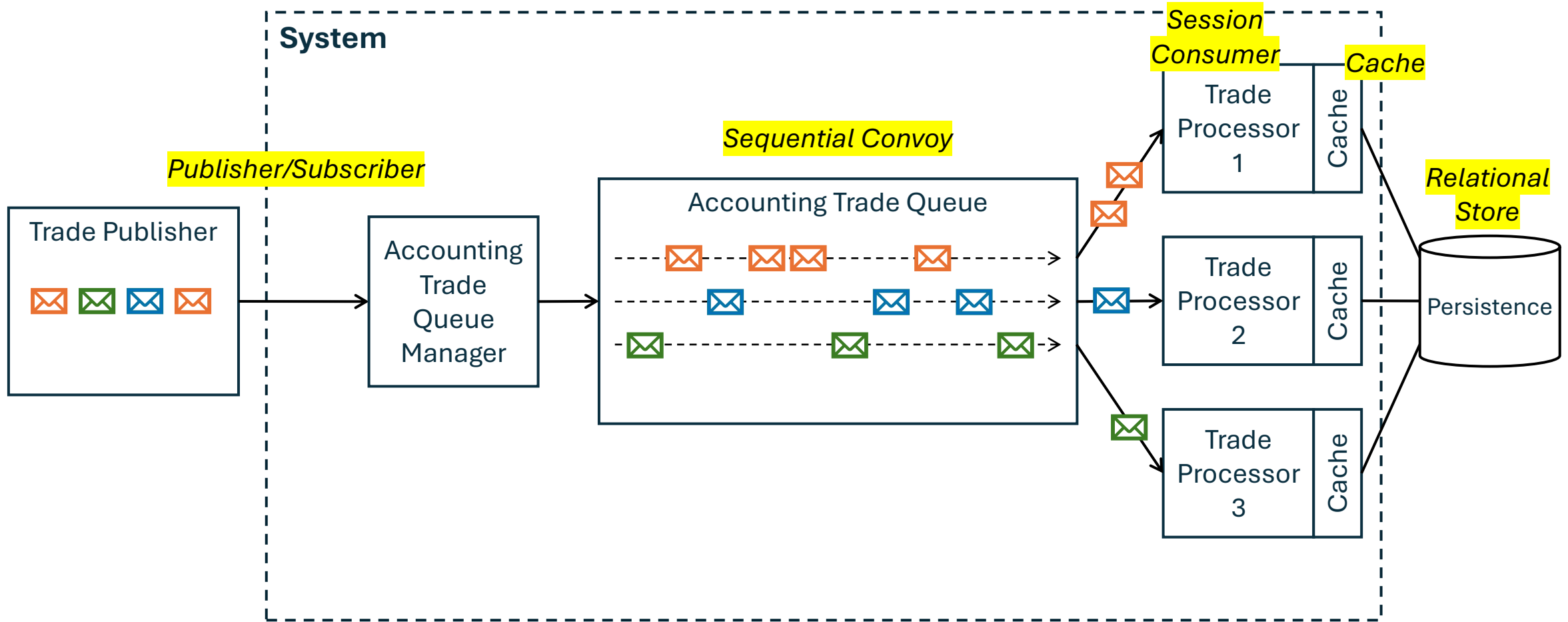
Requirements: Context

- Guaranteed delivery messaging from the trading system
- Multiple consumers
 - Parallel processing for performance
 - The number of consumers scaling up/down as needed
 - FIFO within a single Account/Product combination
 - Partition Key = Account + Product or Account or Product
- Persistence Layer
 - Structured data
 - Relationships with reference data
 - If not performant enough, caching with periodic writes
 - Need to worry about recovery

From Requirements to Patterns



From Requirements to Patterns

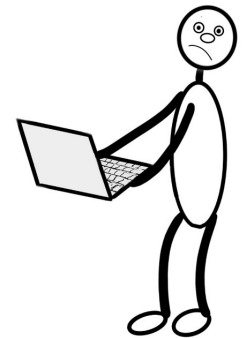


From Patterns to Implementation

Pattern	Implementation Choices
Publisher/Subscriber (Broker/Consumer)	Azure Service Bus or Azure Event Hub
Sequential Convoy	Azure Service Bus Message Sessions
Category (Session) Consumer	Logic App with the Service Bus Peek-Lock Connector or Azure Functions with the Service Bus trigger
Cache	Azure Cache for Redis
Relational Store	Azure SQL Managed Instance

What Now?

- Have yourselves a Pattern Catalog
- Review and reuse what's out there, fine tune to fit your org
- Within your scope of influence:
 - Refactor your architecture function to own and curate Pattern Catalog
 - Build a community
 - Get a conversation going



Thank You

Appendix

Sources

- Wikipedia, https://en.wikipedia.org/wiki/Software_architecture
- Martin Fowler, *"Software Architecture Guide"*
- Grady Booch, *"Architecture as a Shared Hallucination"*
- Fred Brooks, *"The Mythical Man-Month"*
- Martin Fowler, *"Who Needs an Architect?"*
- Ralph Johnson, as quoted by Martin Fowler in *"Who Needs an Architect?"*