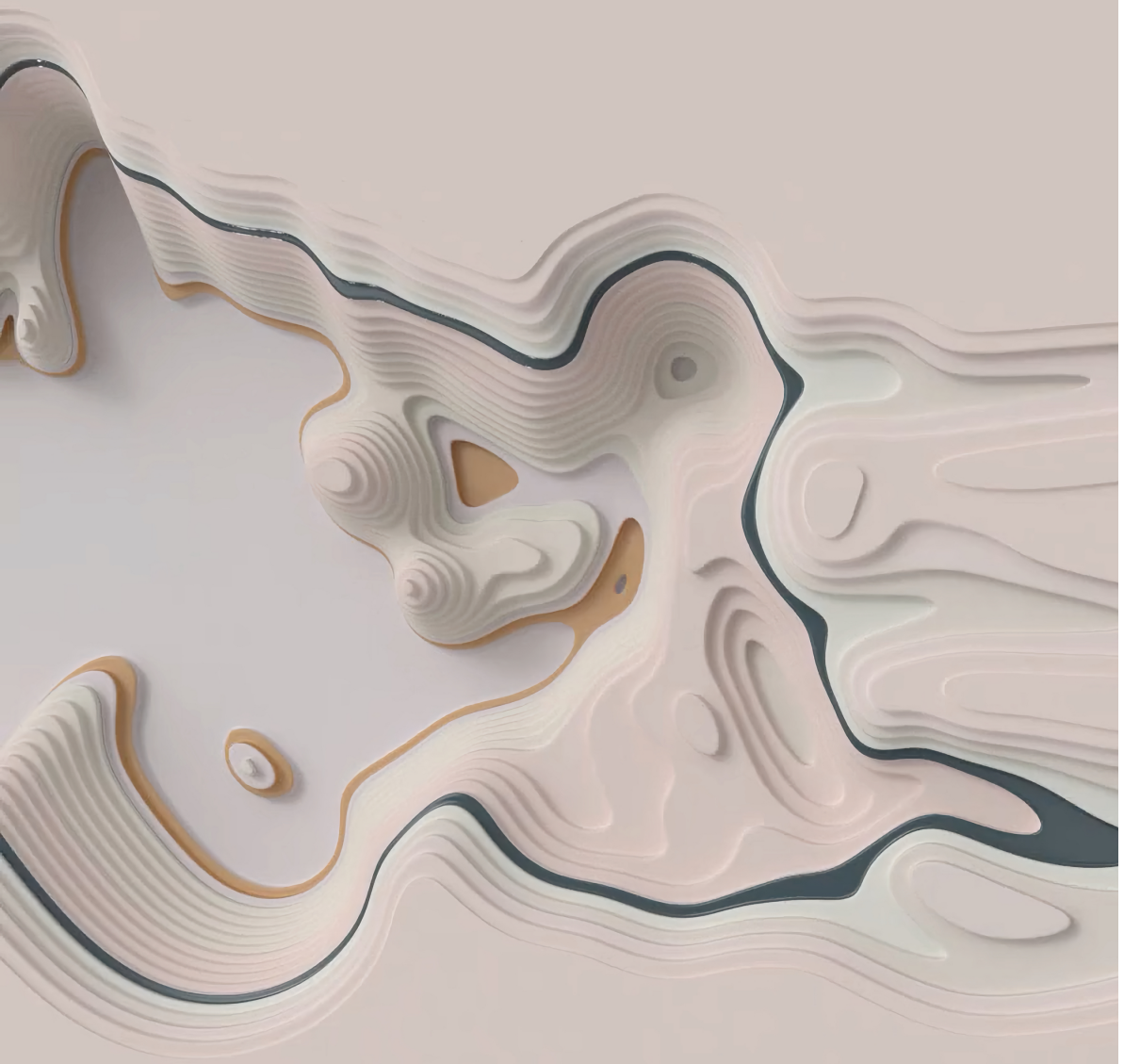


GOTO Amsterdam 2023

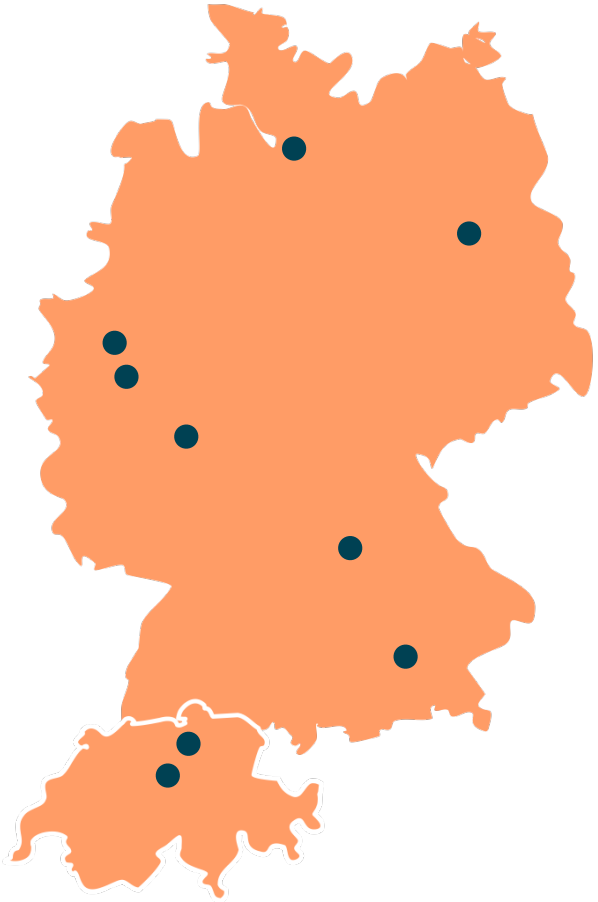
Practical (a.k.a. actually useful) architecture

Stefan Tilkov
stefan.tilkov@innoq.com
@stilkov{@innoq.social}

INNOQ

INNOQ

Founded	Employees	Customers	Annual sales 2022
1999	175+	300+	~ € 22 M



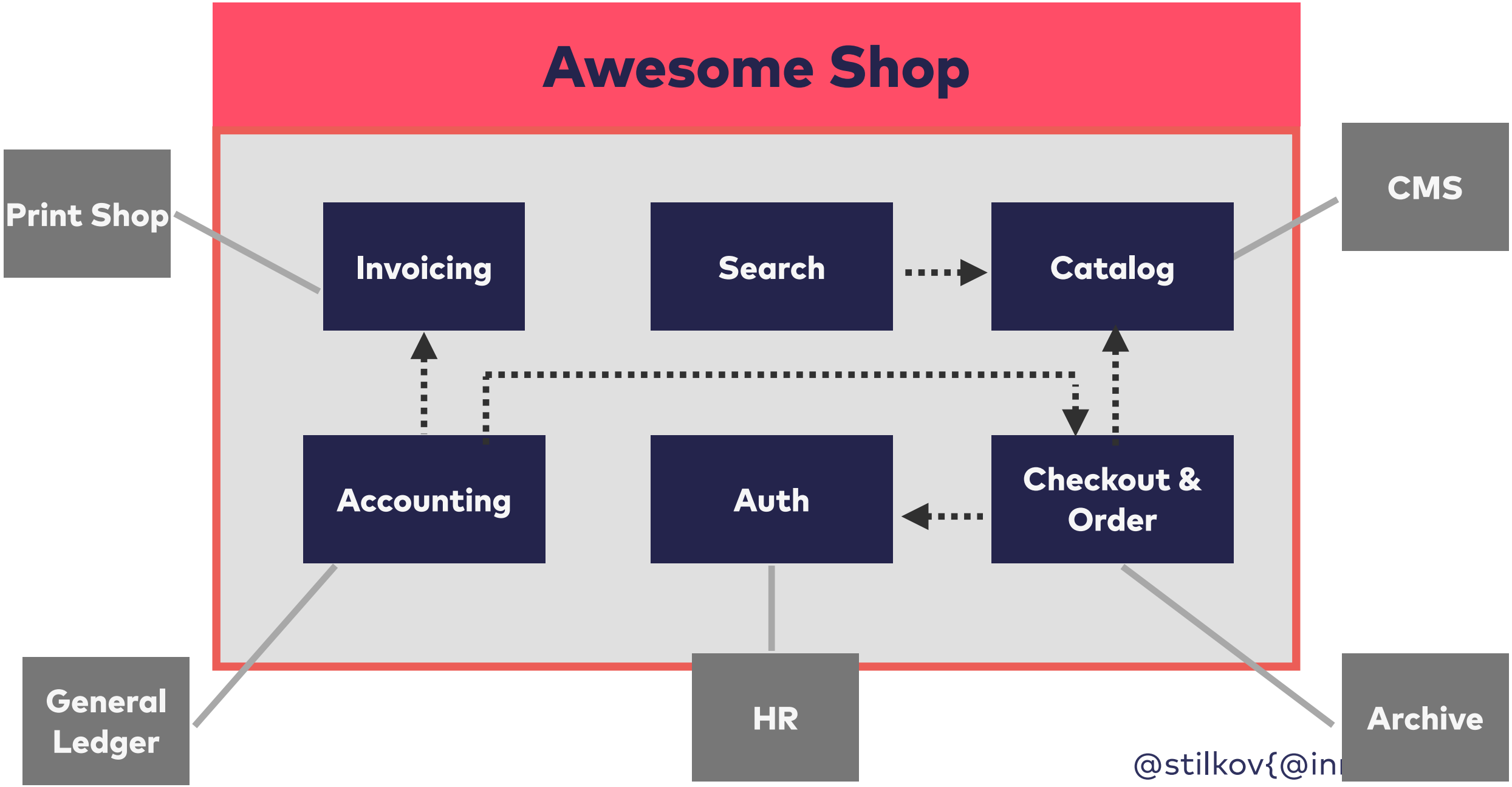
SECTORS Finance | TC | Logistic | E-commerce | Fortune 500 | SME | Start-ups

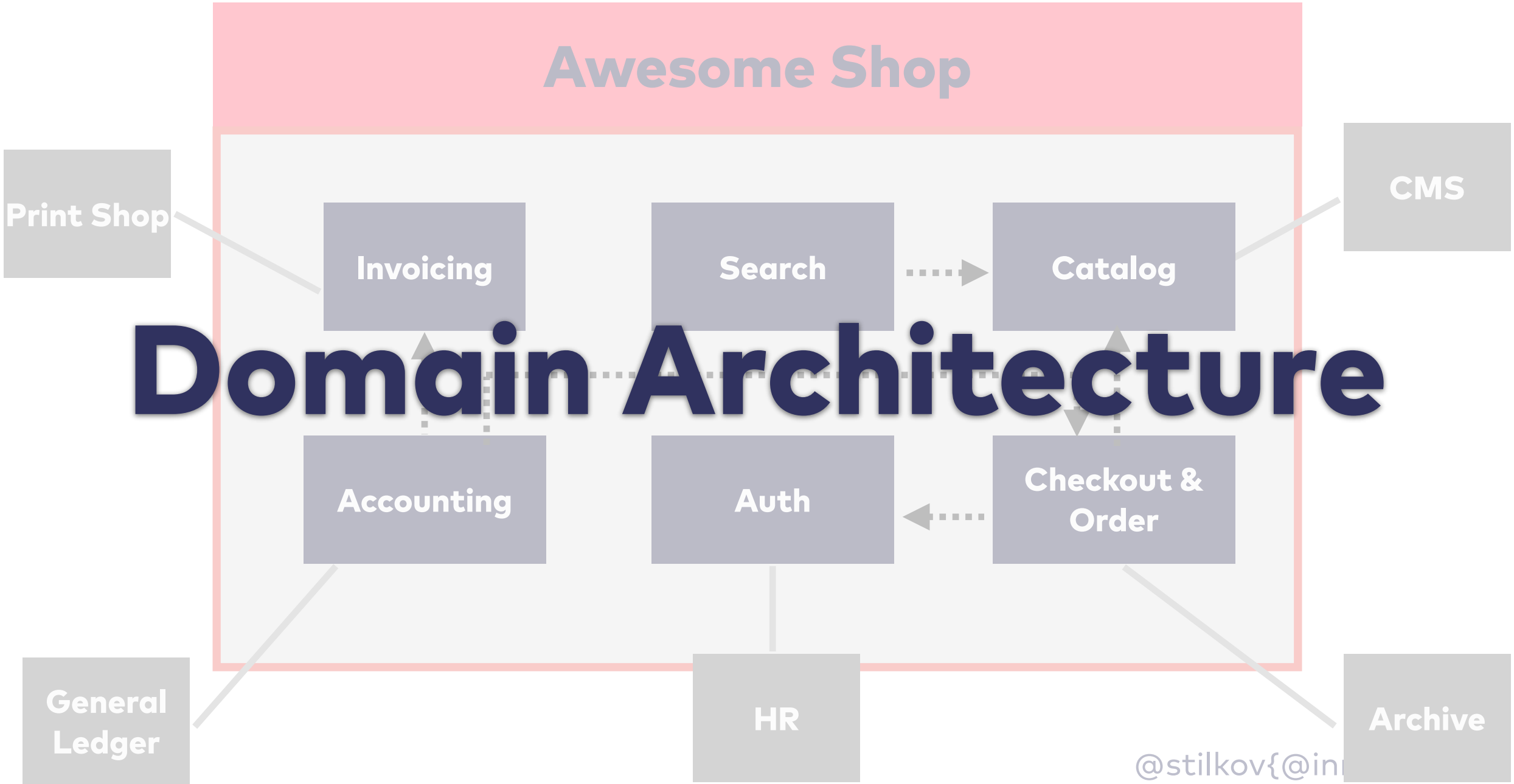
AXA SACAC ebay VORWERK breuninger mobile.de

Ten Recommendations for pragmatic architecture work

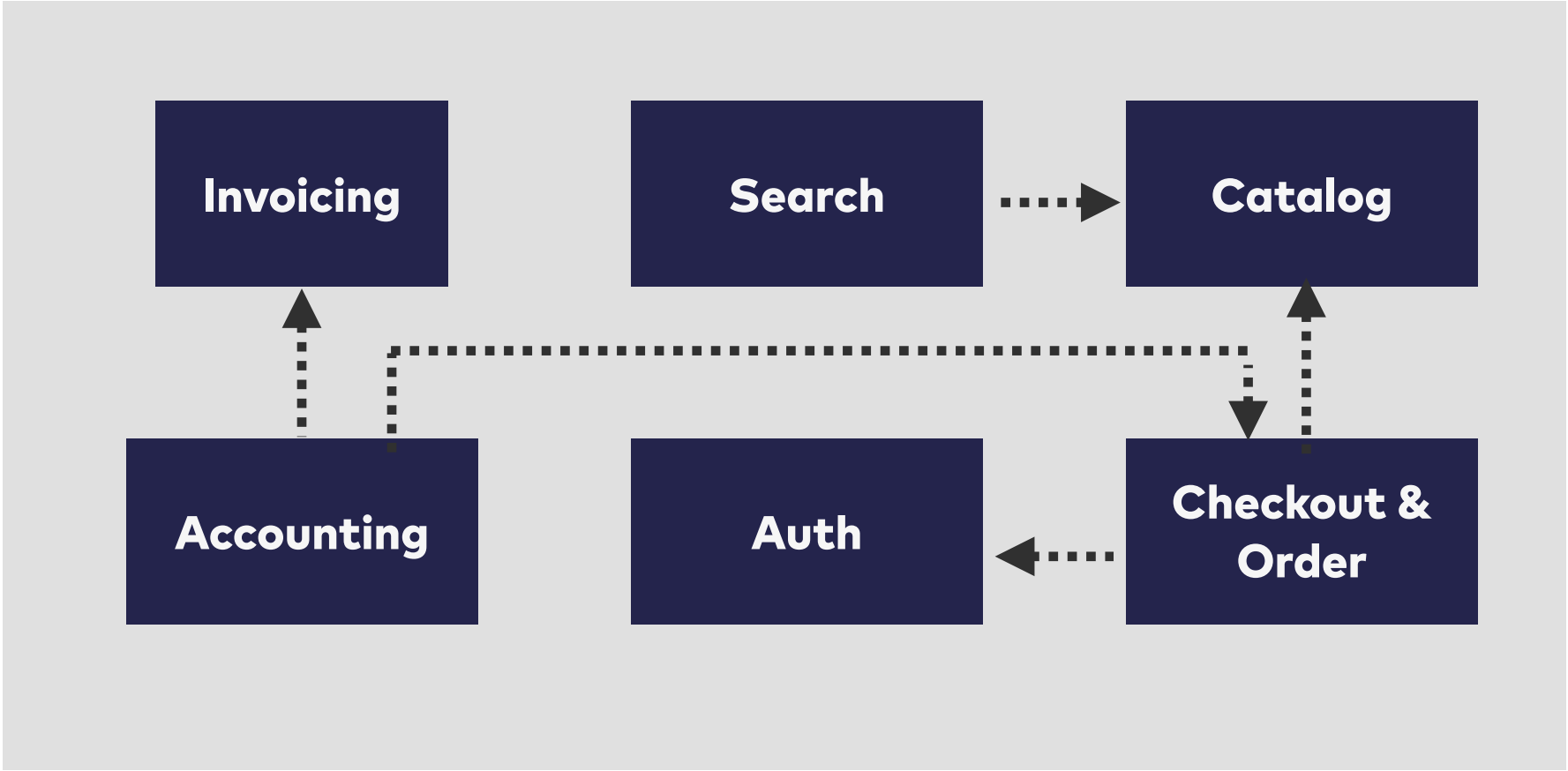
1.
Choose your perspective(s) consciously

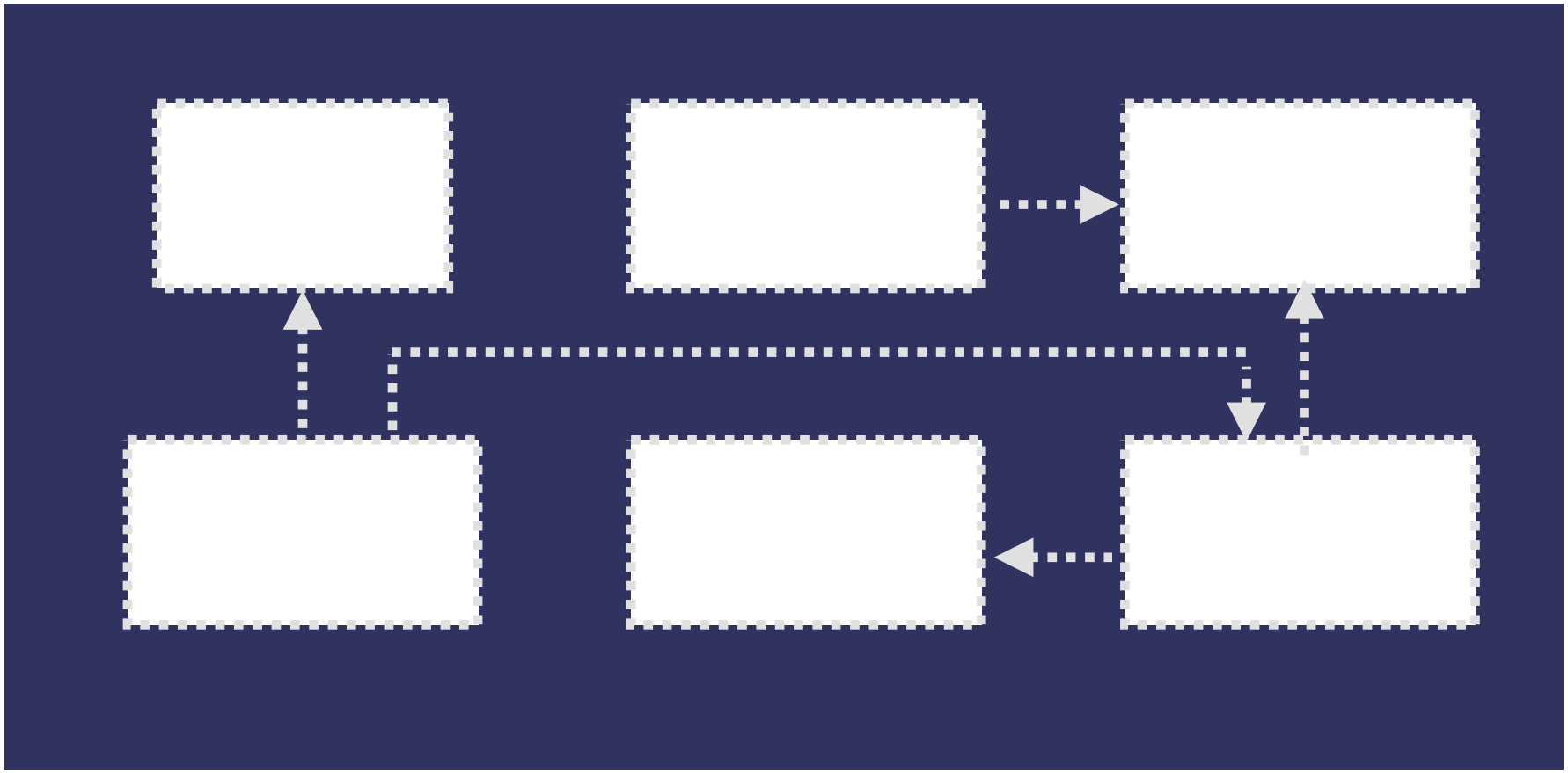

Print Shop
CMS
Awesome Shop
General Ledger
Archive
HR


Awesome Shop
Print Shop
Invoicing
Search
Catalog
CMS
Accounting
Auth
Checkout & Order
General Ledger
HR
Archive



Domain Architecture


Invoicing
Search
Catalog
Accounting
Auth
Checkout &
Order



Macro Architecture

Ruby on Rails
MySQL

NodeJS
ElasticSearch

COTS

Java
Spring Boot

OSS Product

Java
Spring Boot

Ruby on Rails
MySQL

NodeJS
ElasticSearch

COTS

Micro Architecture

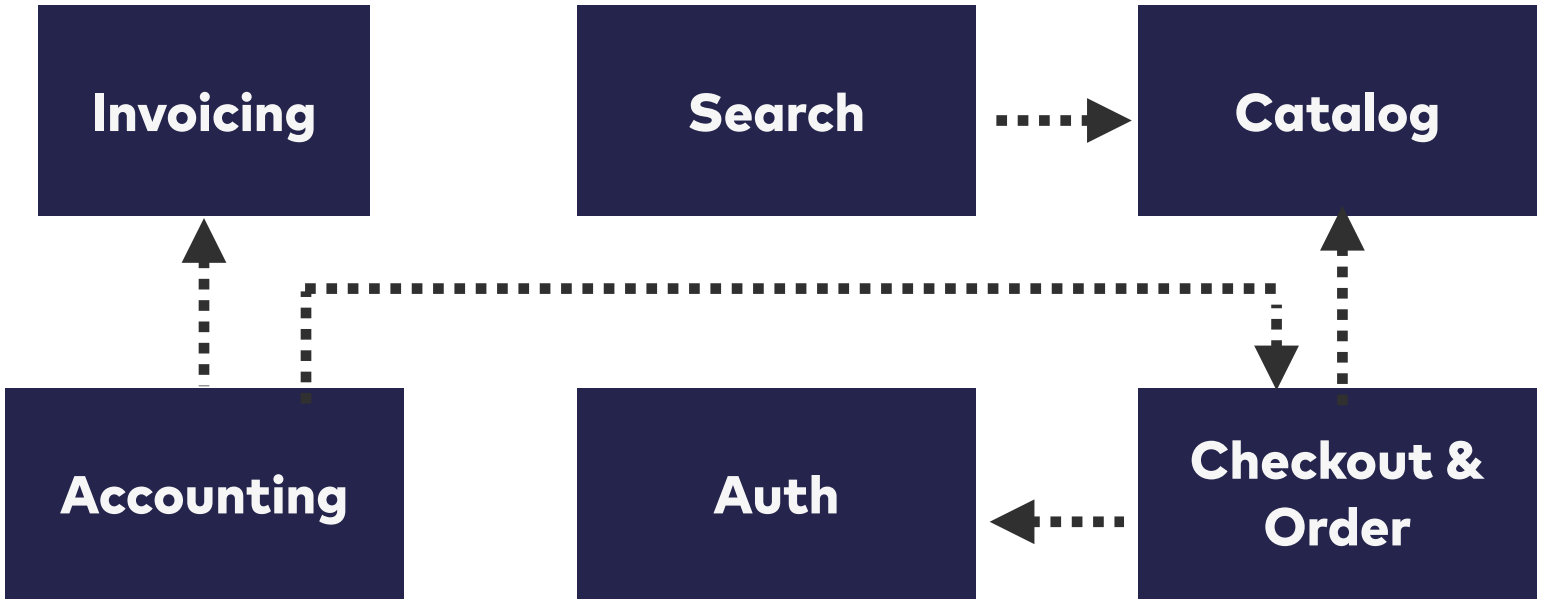
Java
Spring Boot

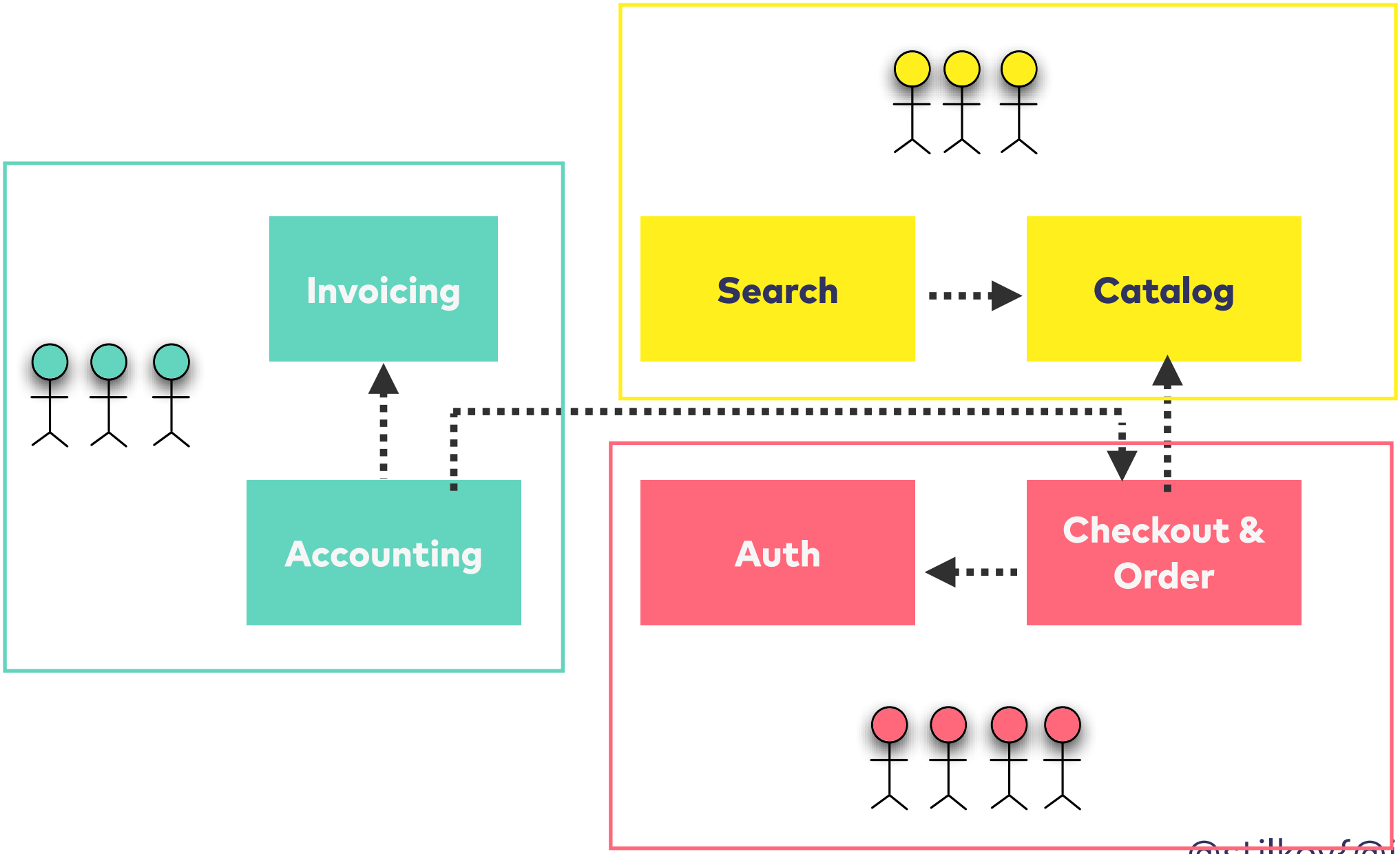
OSS Product

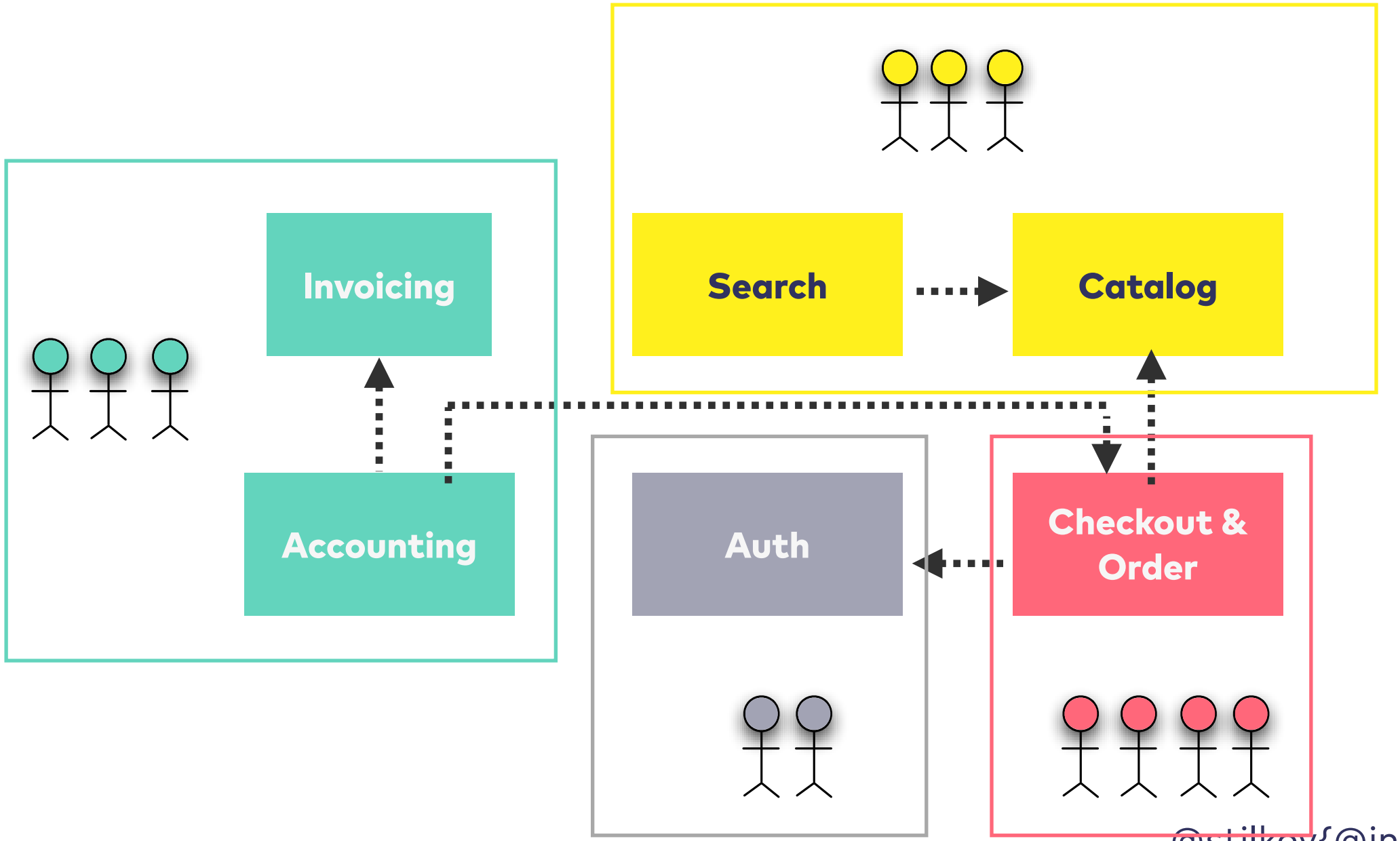
Java
Spring Boot

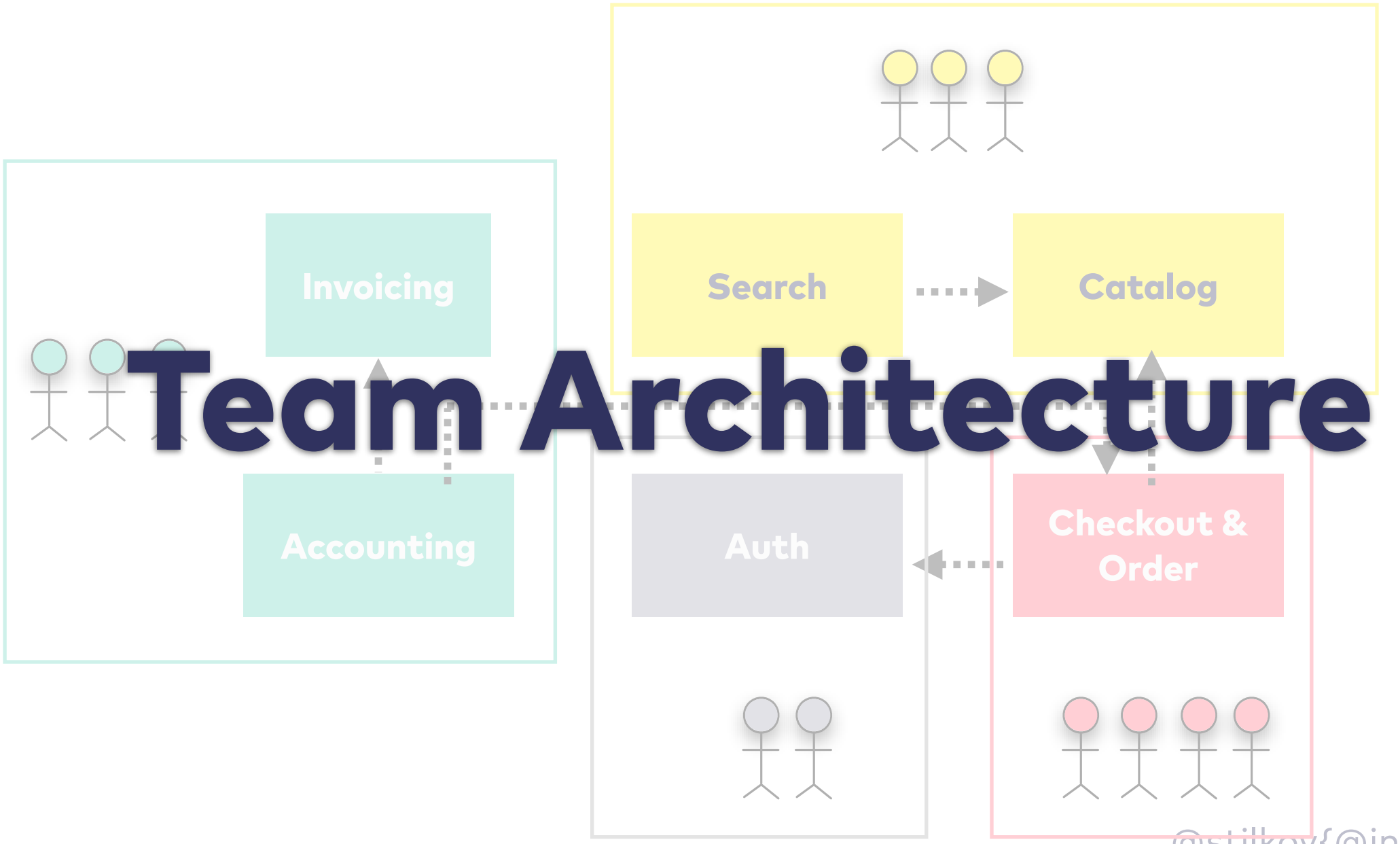
Be aware of each architecture perspective's focus

2.
Explicitly architect your team setup


Invoicing
Search
Catalog
Accounting
Auth
Checkout &
Order


Invoicing
Accounting
Search
Catalog
Auth
Checkout &
Order


Invoicing
Search
Catalog
Accounting
Auth
Checkout &
Order

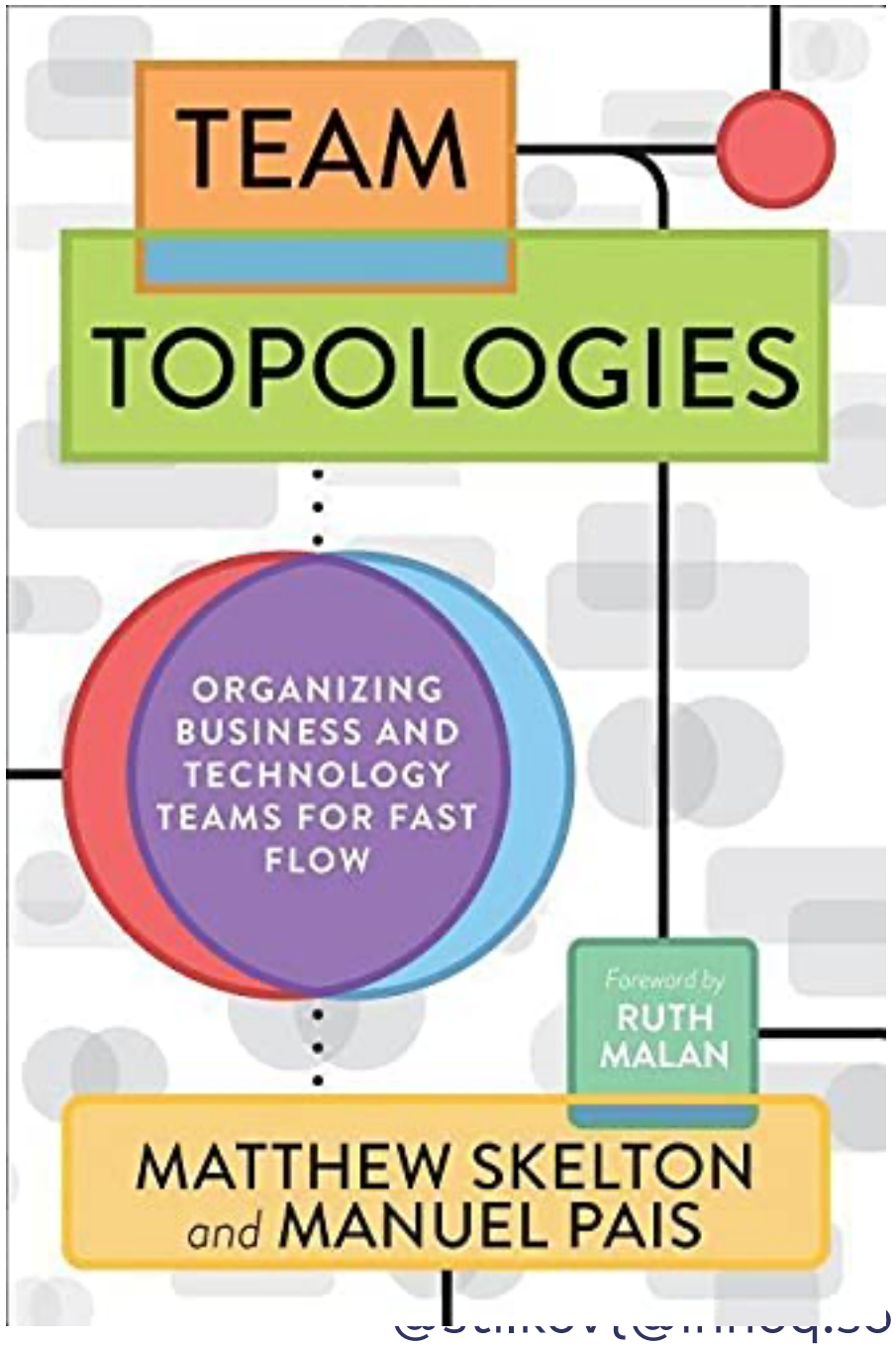


Team Architecture

About Team Topologies

Team Topologies is the leading approach to organizing business and technology teams for fast flow, providing a practical, step-by-step, adaptive model for organizational design and team interaction.

https://teamtopologies.com/



Fundamental topologies

- Stream-aligned team
- Enabling team
- Platform team
- Complicated-subsystem team

Interaction modes

- Collaboration
- X-as-a-Service
- Facilitating

Disagreements first:

“With a team-first approach, the team’s responsibilities are matched to the cognitive load that the team can handle. […] For effective delivery and operations of modern software systems, organizations should attempt to minimize intrinsic cognitive load and eliminate extraneous cognitive load altogether, leaving more space for germane cognitive load.”

Mathew Skelton, Manuel Pais: Team Topologies

You keep using that word.
I don't think it means what you think it means.

“Building and running a software system can be achieved using only four team types. Other team types can be actively harmful to an organization.”

Mathew Skelton, Manuel Pais: Team Topologies

Bold statement!

“Team interactions outside these three core interaction modes are wasteful and indicative of poorly chosen team responsibility boundaries and poorly understood team purposes.”

Mathew Skelton, Manuel Pais: Team Topologies

Another one!

Things that are great about team topologies

- Explicit team-first approach
- Autonomy at the heart of value creation
- Technology-agnostic
- Long-lived teams instead of project thinking
- Based on actual experience, research, collaboration

3.
Match your organizational setup to your project size, context, and culture

Single team

- Some tasks are architecture tasks
- No explicit architecture roles

Small number of collaborating teams

- Architecture board
- Teams send members

Large number of independent teams

- Explicit roles
- Separate arch team
- Arch team sends members to teams

4.
Don't be afraid to decide things centrally

Things that need to be decided centrally

- What to centralize and what to leave to individual teams
- Which teams exist and what their responsibilities are
- Anything relevant* at the seams of more than one team
- Necessary* global policies and strategic* aspects

*for appropriate values of relevant, necessary, and strategic

5.
Pick your battles wisely

Don't do everything at once

- Change programming language and development environment
- Switch from RBDMS to alternative models
- Move to the cloud
- Replace desktop UIs with web frontends
- Introduce asynchronous messaging
- Switch to microservices or whatever this year's hype might be

6.
Enforce the least viable amount of rules, rigidly

“If you love somebody
set them free”

Sting

Be careful with standardization

	Standardized	Team decision
Programming Language	x	
Implementation frameworks & libraries		x
IDE		x
API standard	x	
Frontend stack		x
UI integration approach	x	
Operating environment	x	

Be careful with standardization

	Standardized	Team decision
Programming Language		x
Implementation frameworks & libraries		x
IDE		x
API standard	x	
Frontend stack	x	
UI integration approach	x	
Operating environment		x

Be careful with standardization

	Standardized	Team decision	Best practice
Programming Language		x	x
Implementation frameworks & libraries		x	x
IDE		x	x
API standard	x		
Frontend stack		x	x
UI integration approach		x	x
Operating environment	x		

Things you as an architect might want to enforce

- Interoperability
- Integration
- Operability
- Data analysis
- …

- Privacy
- Accessibility
- Security
- Compliance
- …

7.
Balance prescriptive vs. descriptive architecture

A well-documented architecture, with wonderful, up-to-date descriptions, explicit ADRs, beautiful diagrams, consistent models, created with state-of-the-art tools, can still be bad.

Documenting what exists is a great start, not an end goal

Good decisions will not necessarily satisfy everyone; most decisions that do are likely bad.

Compromising on essential aspects will likely avoid conflicts now, but hurt really bad later

8.
Don't aim for perfection – iterate

"Changes you're starting now will co-exist with changes that started last year and the year before [...] If you adopt that perspective, then you stop trying to rip up the pavement and do something completely new, and you focus a lot more on incremental change."

Michael Nygard – Architecture without an end state

9.
Architect for delivery flow as much as for runtime quality

The quality of a system's architecture is inversely proportional to the number of bottlenecks limiting its evolution, development, and operations

Autonomy helps speed, unmanaged chaos will slow you down

As an architect, your goal is to get out of the way of the people who do the actual work

10.
Be boring and do more with less

Theory vs practice

In theory, architects …

- analyze quality requirements and figure out the actual business motivation
- pick a suitable development method, architecture, and implementation technology
- continuously re-evaluate whether past decisions still make sense given updated information

In practice, …

A reasonable software architecture for an e-commerce system, based on the customer's preferences for React, Kafka, and Microservices, could include the following components:

1. User interface: The user interface layer should be built using React, which is a popular and powerful JavaScript library for building user interfaces. React can be used to create dynamic and interactive web applications that are responsive to user input.
2. Application layer: The application layer should be built using a microservices architecture. Microservices are small, independently deployable services that communicate with each other through lightweight protocols. Each microservice should be responsible for a specific task, such as order processing, inventory management, or payment processing.
3. Messaging layer: The messaging layer should be built using Kafka, which is a distributed streaming platform that can handle large volumes of data in real-time. Kafka can be used to enable communication between microservices and to process events, such as order updates or inventory changes.

4. Database layer: The database layer should be designed to support a microservices architecture. Each microservice should have its own database, which can be optimized for its specific needs. This can improve performance and scalability, and reduce the risk of data corruption or loss.
5. Security layer: The security layer should be designed to protect the e-commerce system from malicious attacks and unauthorized access. This can include measures such as authentication and authorization, data encryption, and secure communication protocols.
6. Analytics layer: The analytics layer should be designed to provide insights into customer behavior, sales trends, and other key performance indicators. This can include tools such as data visualization, machine learning, and predictive analytics.

Overall, this architecture is designed to be scalable, flexible, and resilient, and can help the e-commerce system handle large volumes of traffic and data.

but what about data mesh?

Data Mesh is a newer architectural approach that emphasizes the decentralization of data ownership and management, as well as the creation of cross-functional data teams within an organization. In a Data Mesh architecture, data is treated as a product and is owned by the teams that produce and consume it.

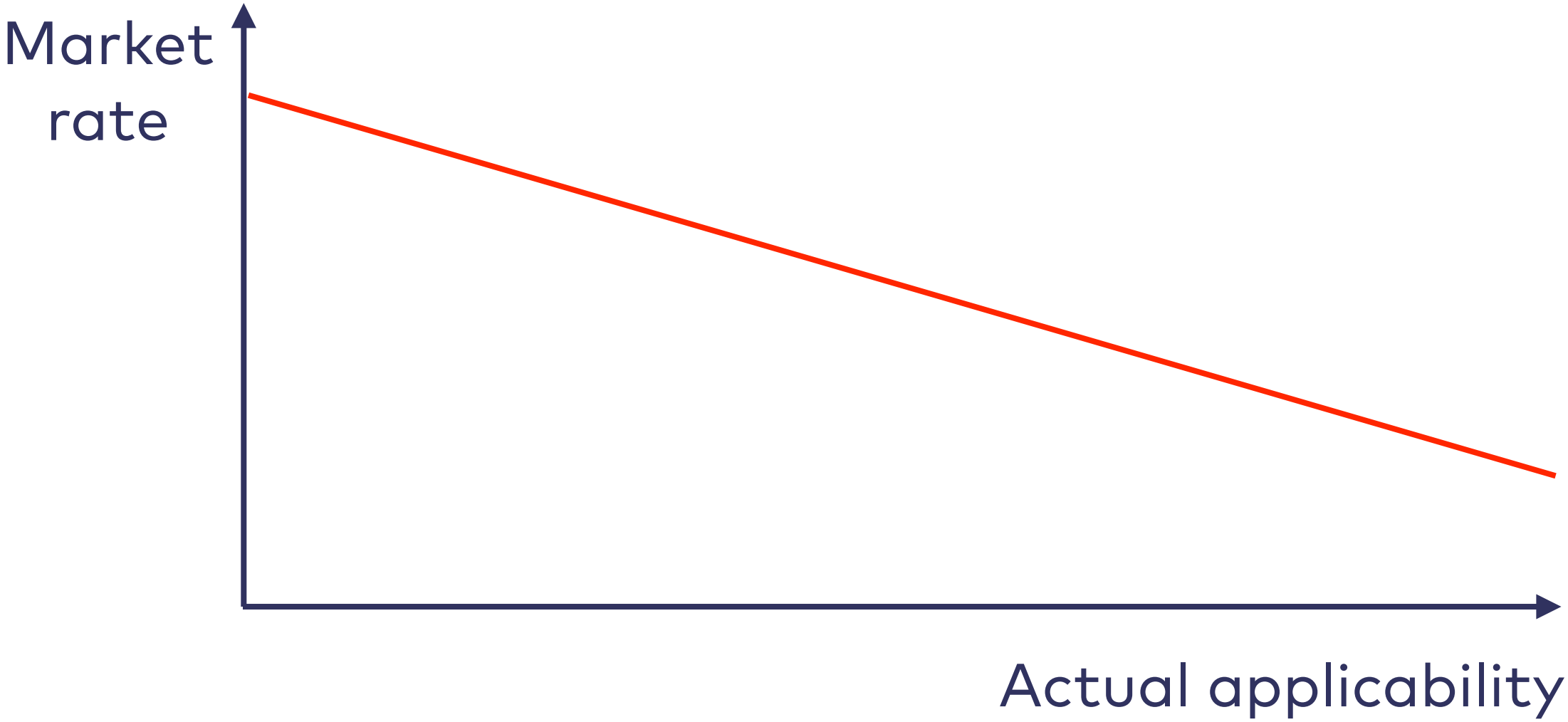
For an e-commerce system, a Data Mesh architecture could be applied to the data layer. Each microservice could have its own data store, and each data store could be owned and managed by the team responsible for that microservice. This can help to reduce dependencies and improve autonomy, as well as improve the quality and accuracy of the data.

In addition, Data Mesh can help to promote collaboration and communication between different teams, which can be especially important for an e-commerce system that involves multiple functions and stakeholders. Data Mesh can also enable faster and more efficient data processing and analysis, which can help to improve the overall performance and effectiveness of the system.

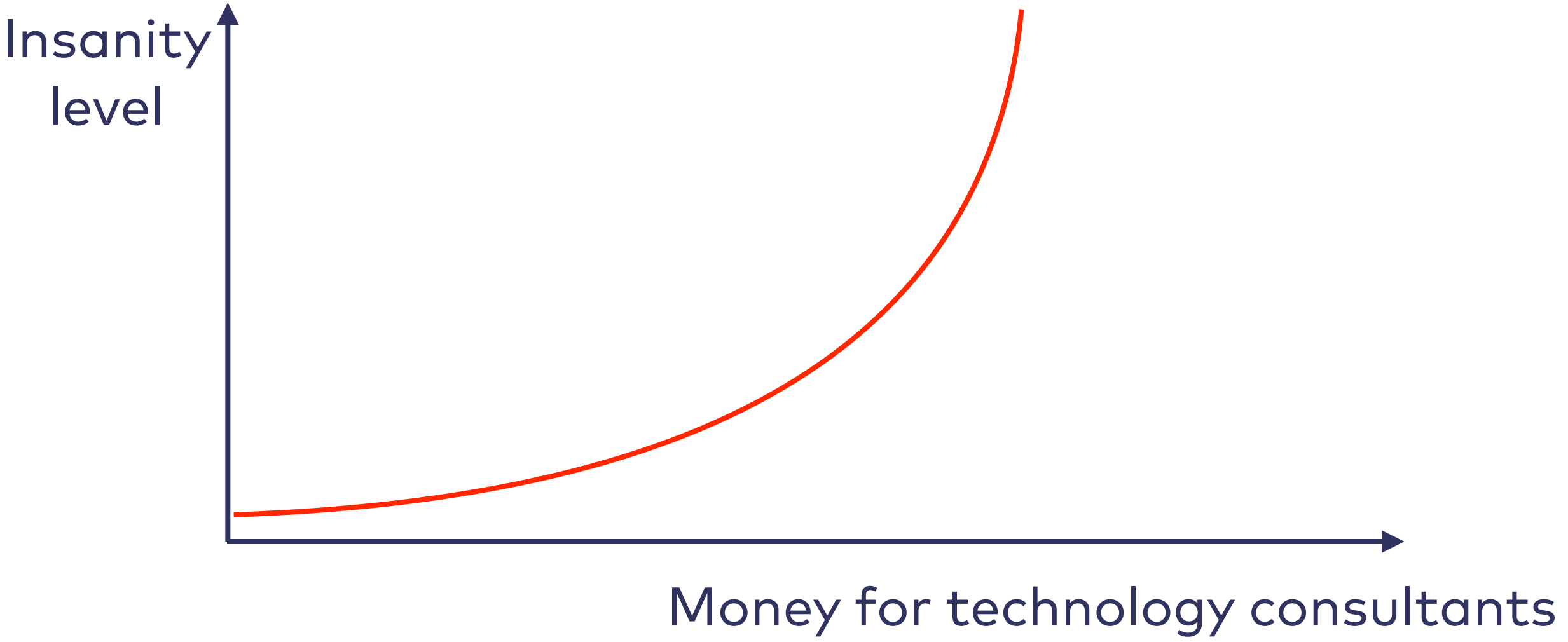
It's worth noting that implementing a Data Mesh architecture can be more complex than a traditional microservices architecture, and can require additional resources and expertise. However, for organizations that are willing to invest in the necessary tools and processes, Data Mesh can be a powerful approach for improving data quality, agility, and collaboration.

If we think stochastic parroting is bad, maybe we should apply that criticism to ourselves, too

Technology suitability



Company insanity



Prefer simple, straightforward solutions to overly clever and complicated, cool approaches

Focus innovation on the domain, not technology

Conclusion

Architecture remains the most interesting and fascinating area in IT because it matters. If you get it wrong, everything breaks down – yet even if you get it right, success is not guaranteed

Whatever you do, consider the context and beware of one-size-fits-all solutions

Don't aim for perfection – aim for evolvability

That's all I have

Stefan Tilkov
stefan.tilkov@innoq.com
+49 170 471 2625
@stilkov – @stilkov@innoq.social

innoQ Deutschland GmbH

Krischerstr. 100
40789 Monheim
+49 2173 3366-0

Ohlauer Str. 43
10999 Berlin

Ludwigstr. 180E
63067 Offenbach

Kreuzstr. 16
80331 München

Wendenstr. 130
20537 Hamburg

Spichernstr. 44
50672 Köln

Königstorgraben 11
90402 Nürnberg

innoQ Schweiz GmbH

Gewerbestr. 11
6330 Cham

Hardturmstr. 253
8005 Zürich