

Modular Monolith Architectural Style



Thang Chung

December 06, 2024



**VIETNAM
WEB
SUMMIT**

**Nash
Tech.**

Who am I

Thang Chung

- Technical Manager at NashTech VN
- Microsoft Azure MVP
- Creator of Vietnam Microservices Group on Facebook (>20k members): <https://www.facebook.com/groups/645391349250568>
- Experience: >18 years in software consult, design, development, and deployment software for outsourcing, product, and startup companies.
- Leading architectural design area at NashTech VN.
- Expertise in cloud computing, cloud-native platform, serverless, and WebAssembly/WASI.
- Blog: <https://dev.to/thangchung>
- GitHub: <https://github.com/thangchung>
- LinkedIn: <https://www.linkedin.com/in/thangchungatwork>
- X (former Twitter): @thangchung

Agenda

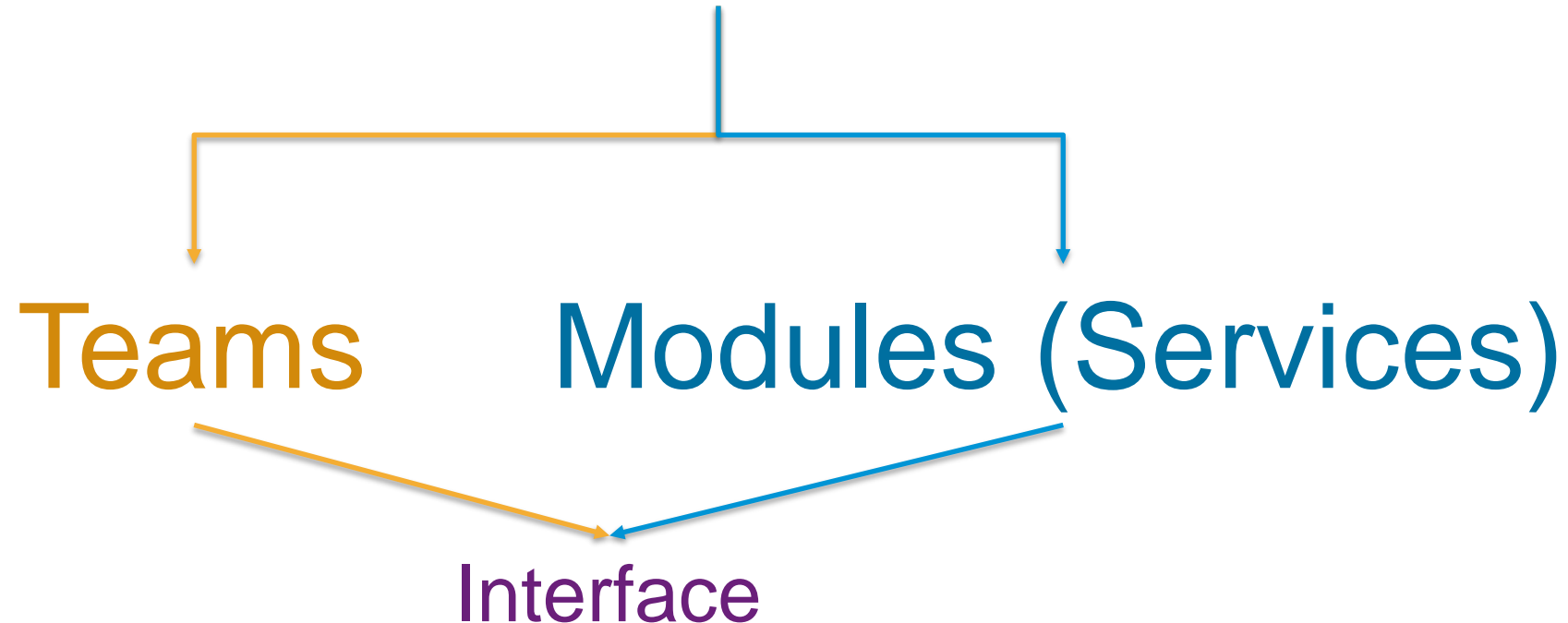
1. **Socio-technical Topologies**
2. **Modular Monolith**
3. **Some design patterns to support maintainability and extensibility**

Socio-technical Topologies



Software Development

Socio-technical Topologies



Communication (social) + module interaction (technical)
Information hiding – reduce cognitive load

Now let's look at a metaphor scenario to demonstrate the situation drastically...

Conway's law

Architecture copies communication structures of the organization

https://www.melconway.com/Home/Conways_Law.html

Setup the software team (planning)

Inverse Conway's law



Identity and Access Management
(IAM) domain

Inverse Conway's law



CoffeeShop domain

Setup the software team (reality)

Covid pandemic, economic crisis, war,
etc. from 2019 to now...

Inverse Conway's law

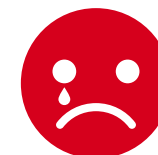


Identity and Access Management
(IAM) domain

Inverse Conway's law

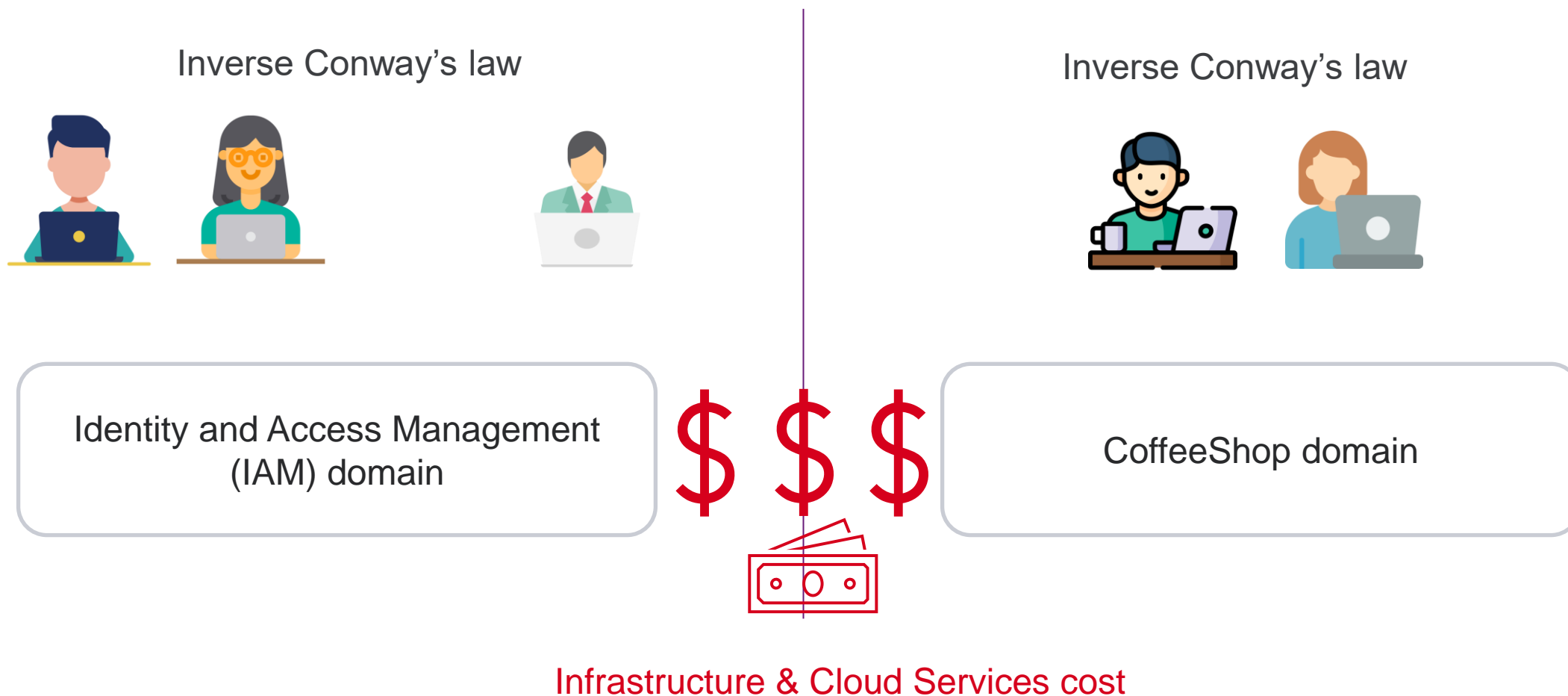


CoffeeShop domain



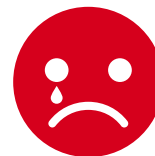
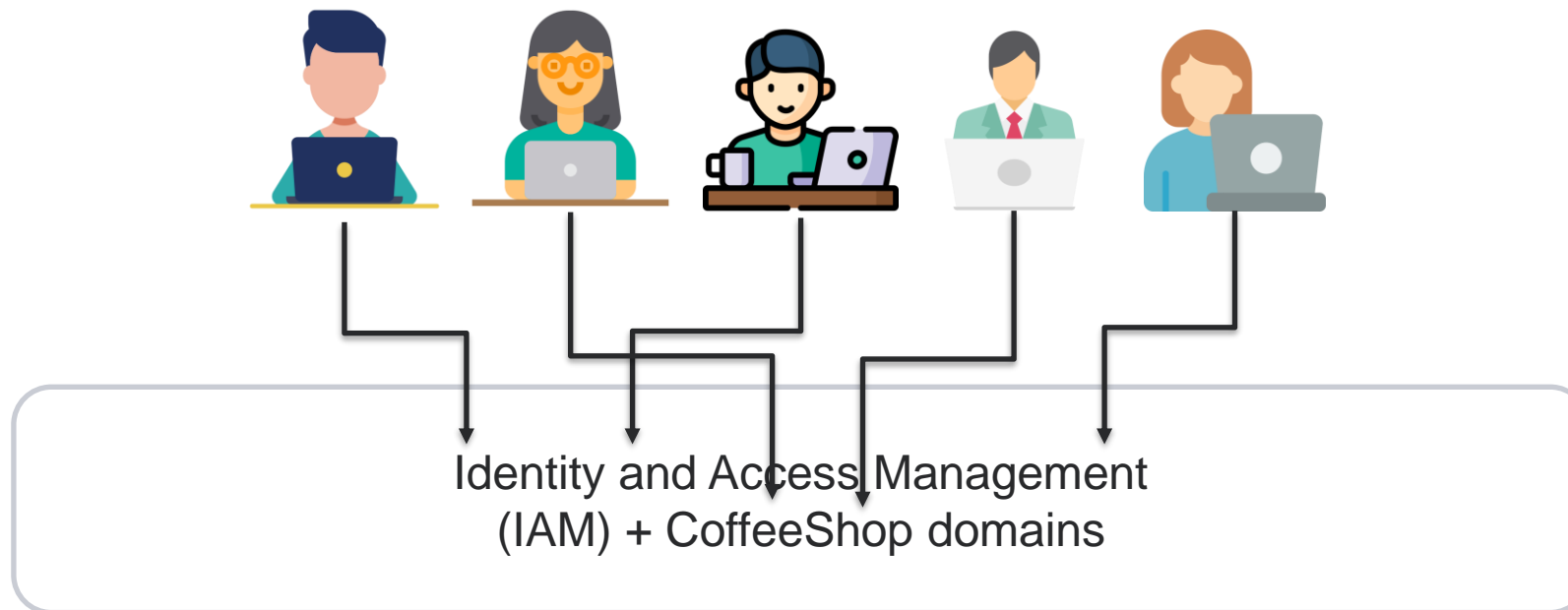
Business's operating costs

Setup the software team (reality)



Setup the software team

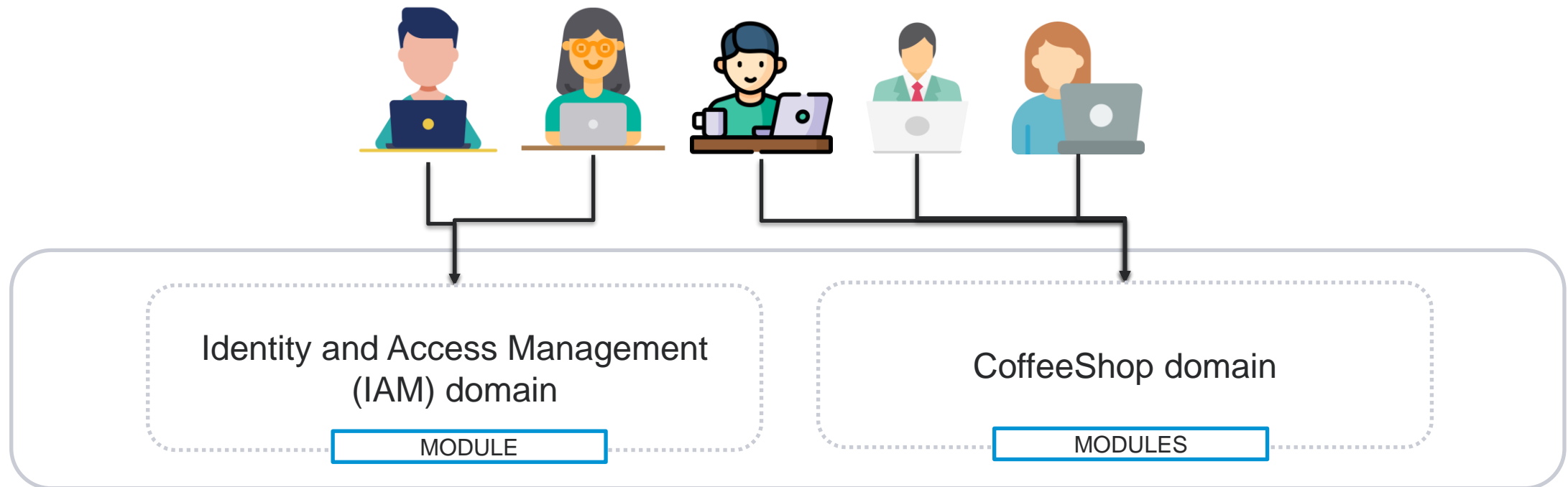
Inverse Conway's law



Big ball of mud

Setup the software team

Inverse Conway's law



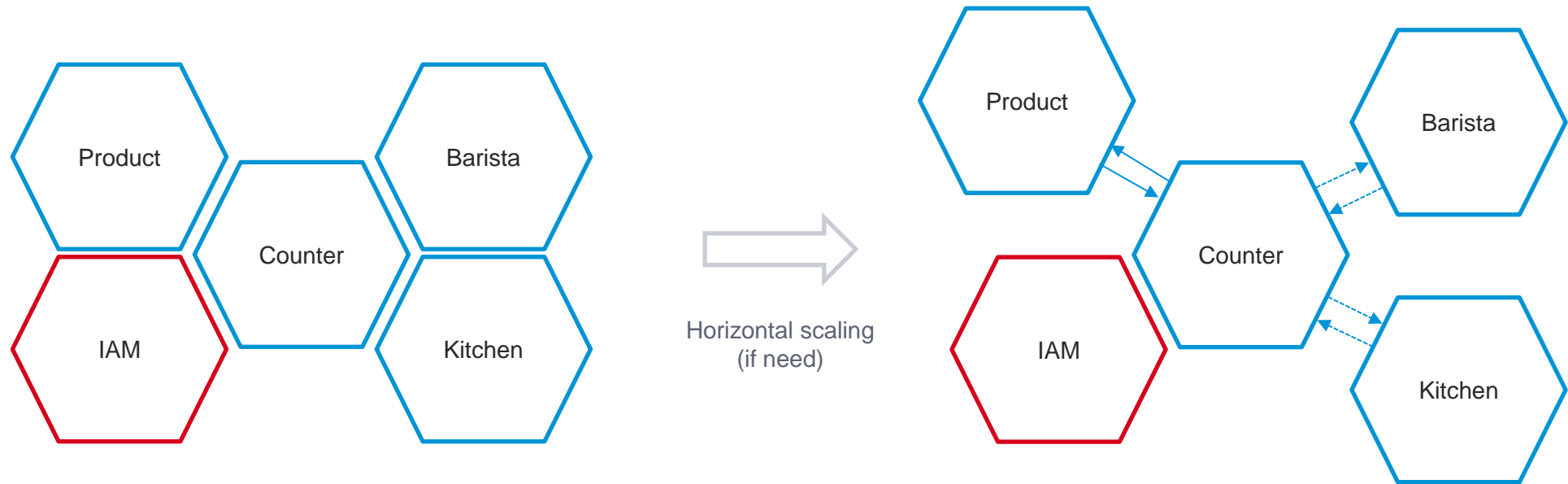
Modular Monolith

Modular Monolith

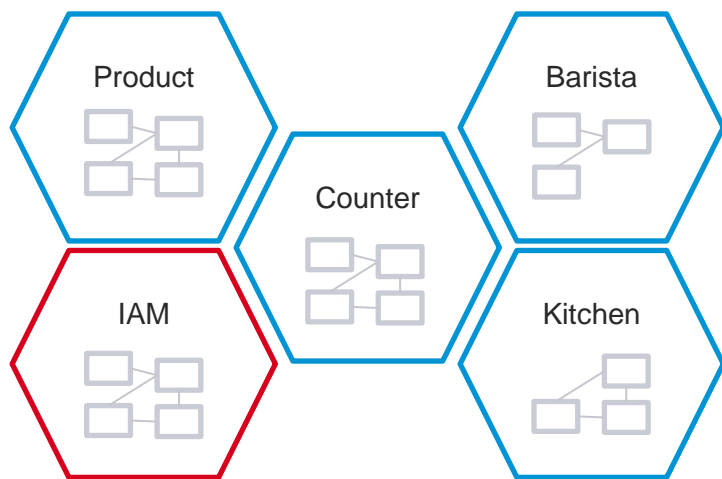


Modular Monolith definition

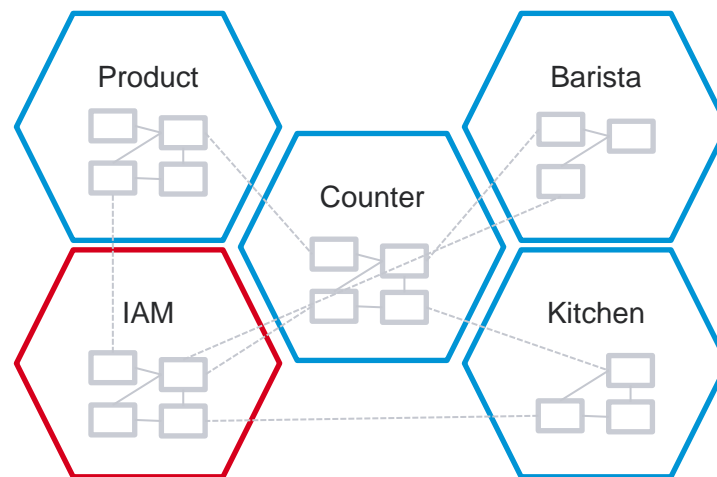
- A Modular Monolith is a software architecture that structures the application as a **single deployment unit** (like a traditional monolith) but **organizes its internal components or modules** in such a way that they are **loosely coupled and highly cohesive**.
- **Each module** within the architecture **focuses on a specific business domain or functionality**, similar to how microservices operate, but **without the distributed system** complexity.



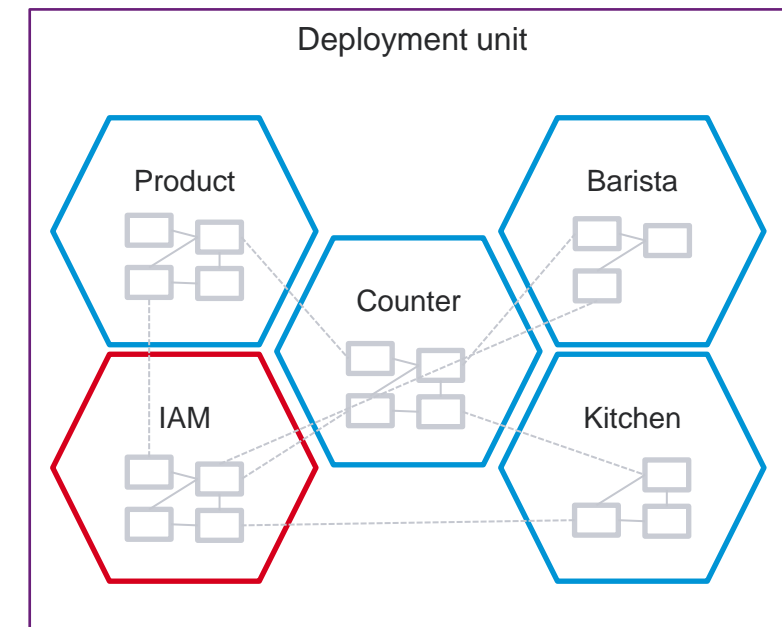
Key Characteristics



Cohesion within Modules

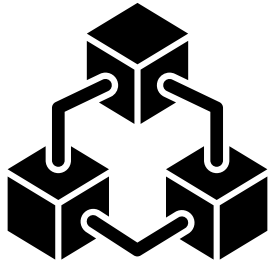


Loose Coupling between Modules

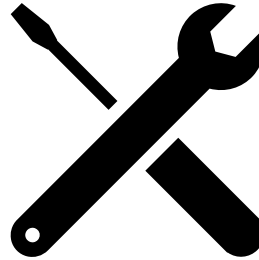


Single Deployment Unit

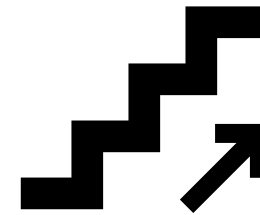
Advantages of Modular Monoliths



Simplified Development and
Deployment



Improved Maintainability



Flexibility for Future Scaling

Modular Monolith

- Some design patterns to support maintainability and extensibility



What we want



Image from freepik

What we have



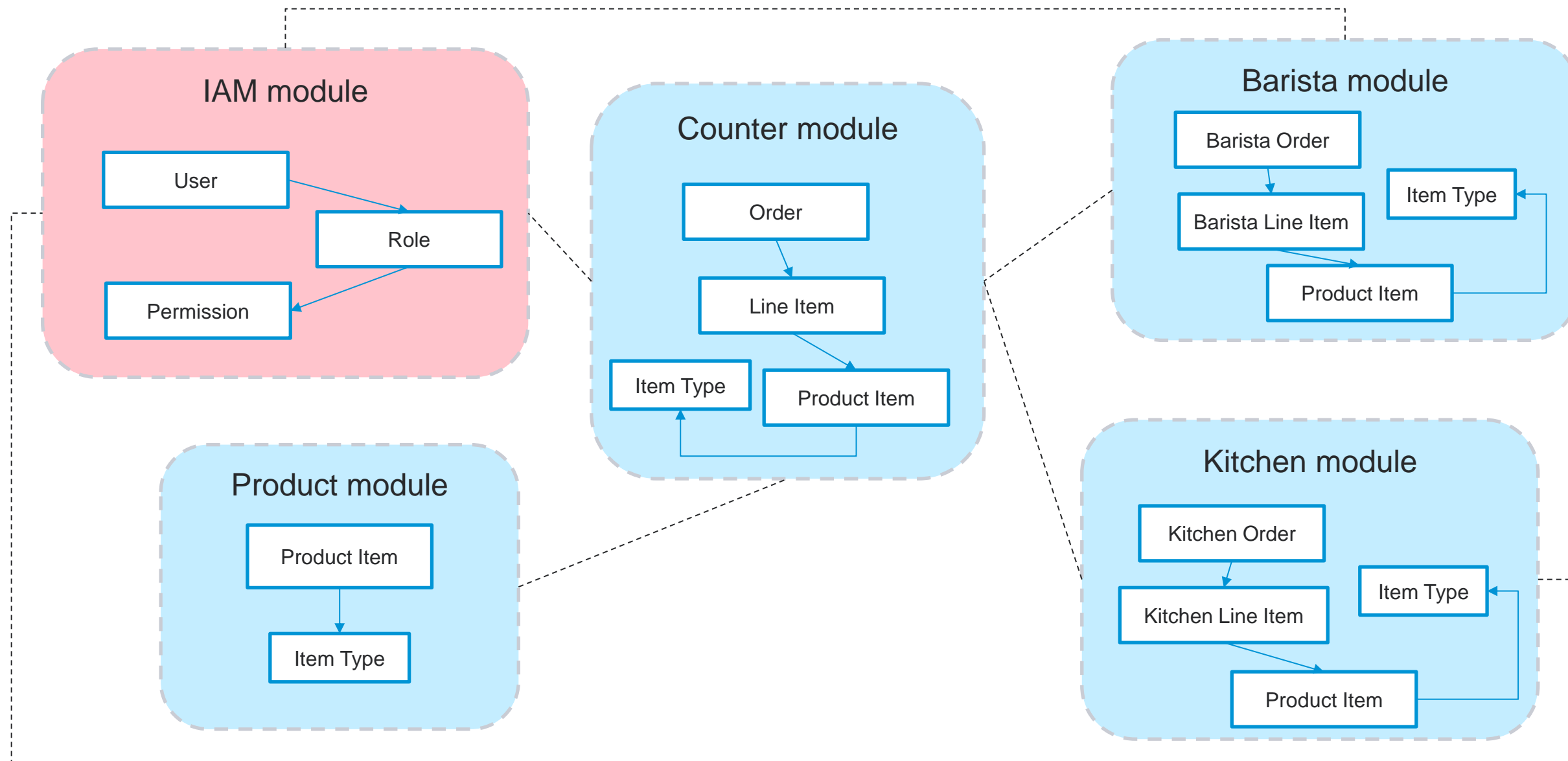
Image from corecursive.com

Then how can we avoid this problem?

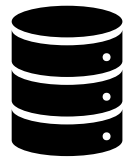
Domain centric



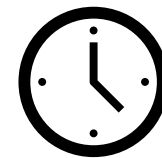
Domain-driven Design – Bounded Context



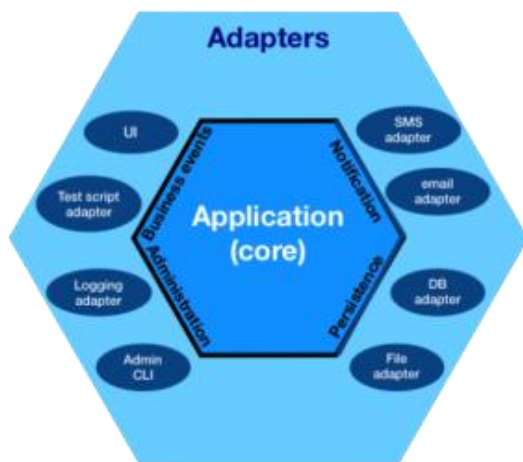
Modelling around business domains, but how about
Infrastructure (database, email provider), services, web server,
message broker, background jobs...



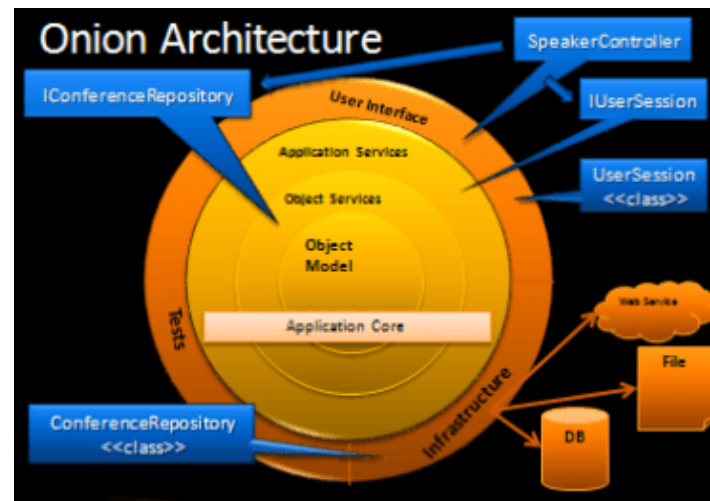
...



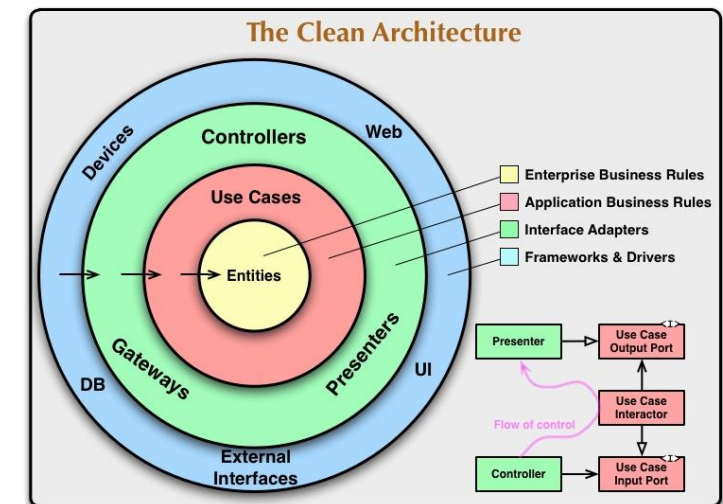
Applying some architectural patterns



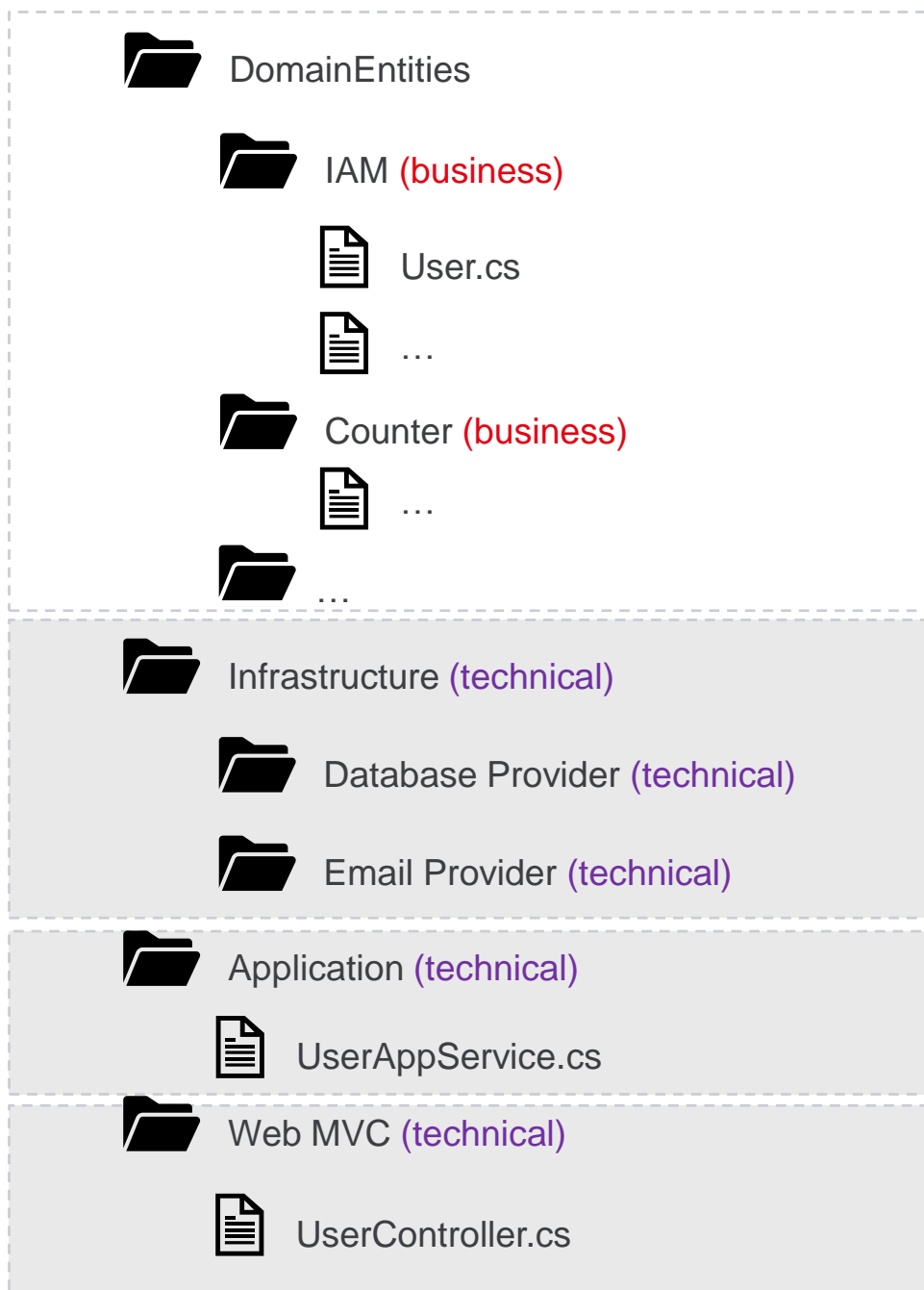
Hexagonal Architecture



Onion Architecture





Clean Architecture

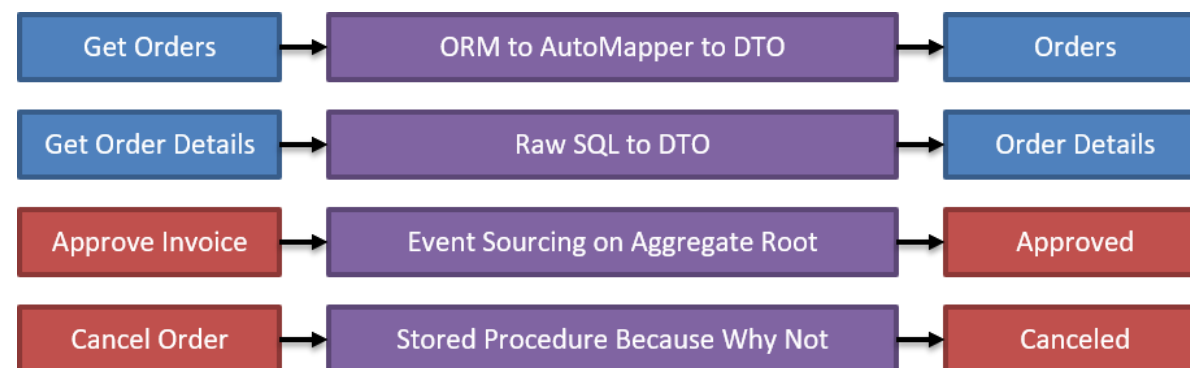
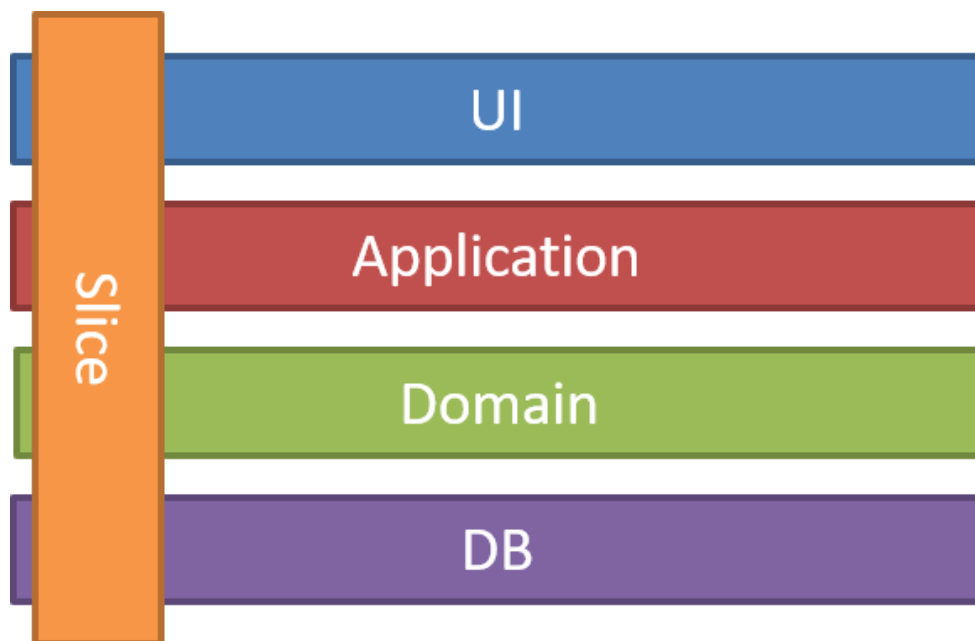


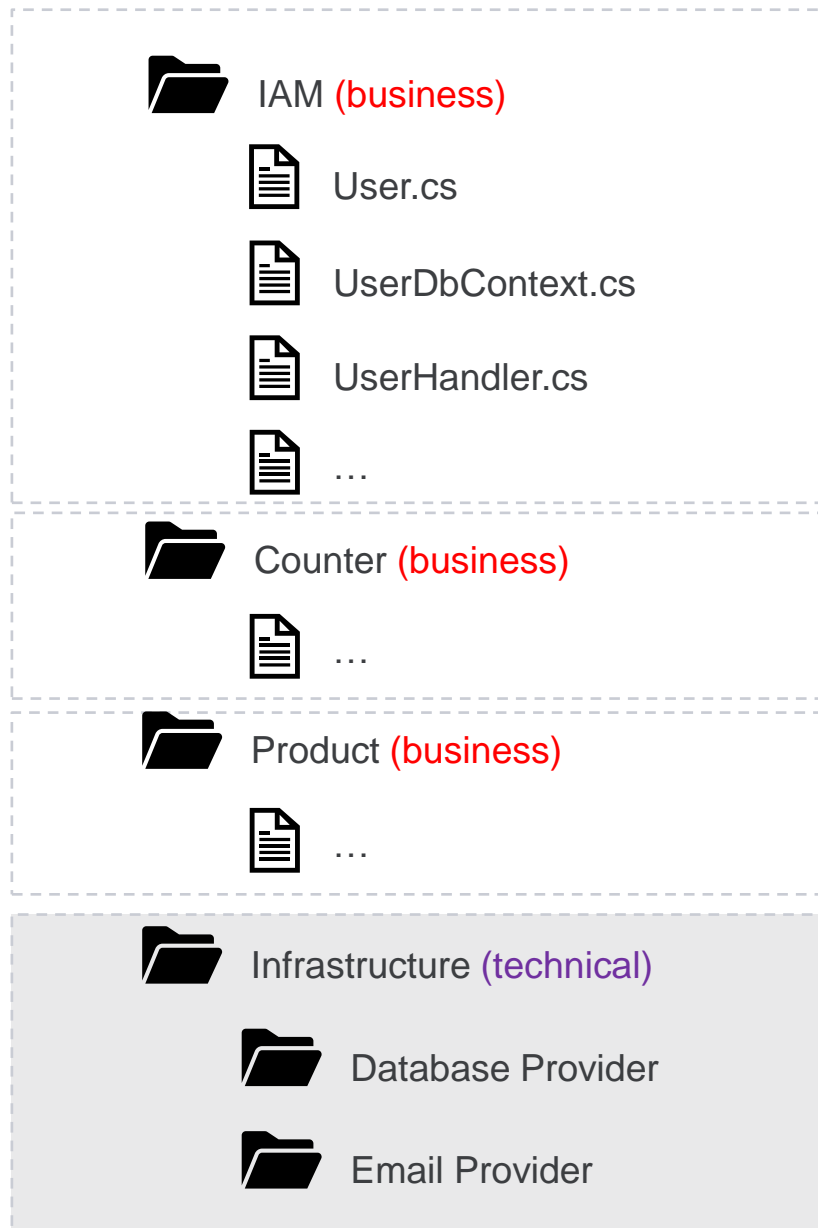
It does not look like what we want to build for **multiple teams scaling...**



 Domain components
 Shared components

Vertical Slide Architecture (VSA)







It's better now

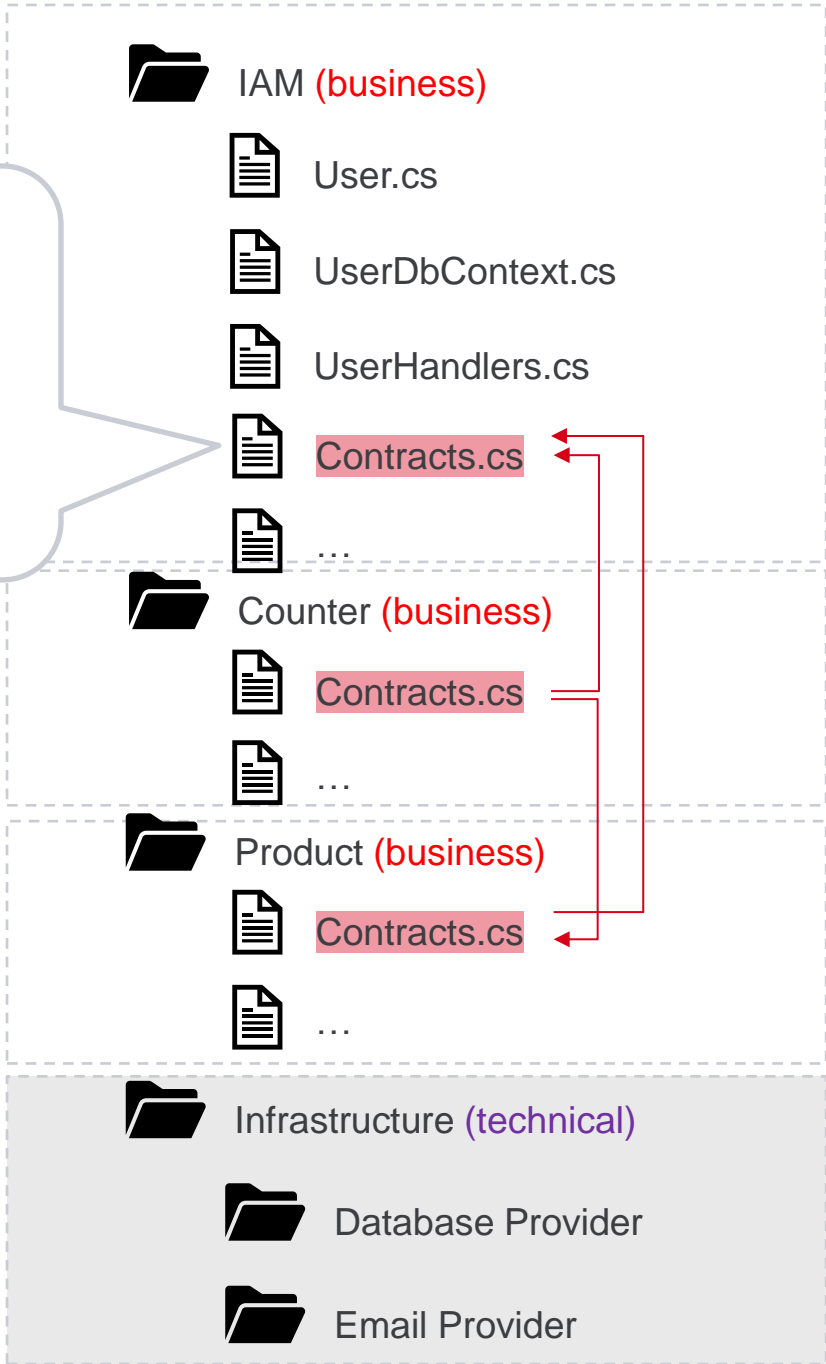


But how about code calling from the Counter module to the Product module...

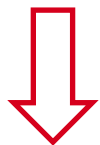


 Domain components
 Shared components

Only contains **interface objects** (SOLID principal)





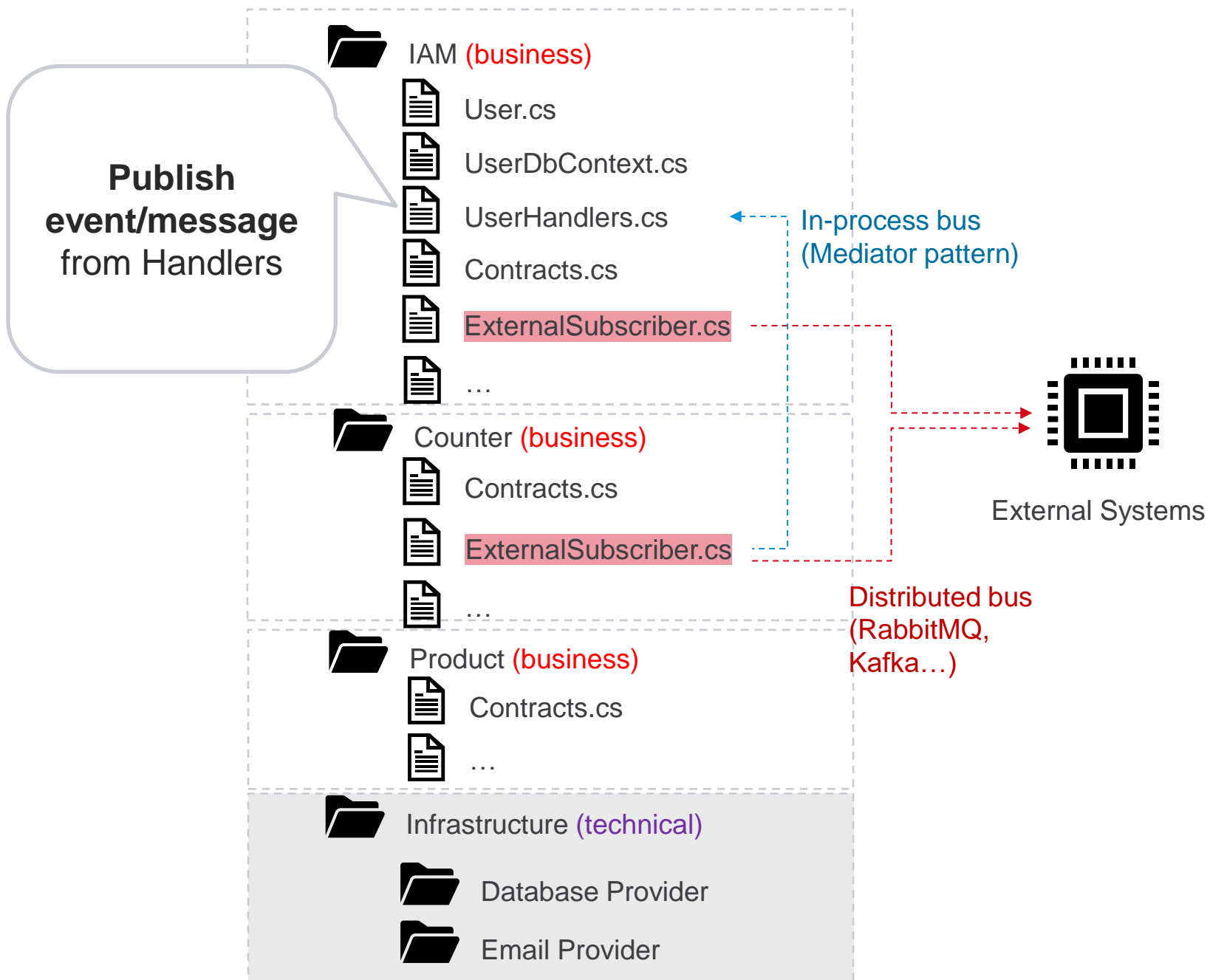
For **synchronous communication**, remember REST API communication, but it is now an **in-process module invocation**.



But how about asynchronous communication with external system...





 Domain components
 Shared components



Asynchronous communication
 (external system, publish event to data platform – batching or streaming)



 Domain components
 Shared components

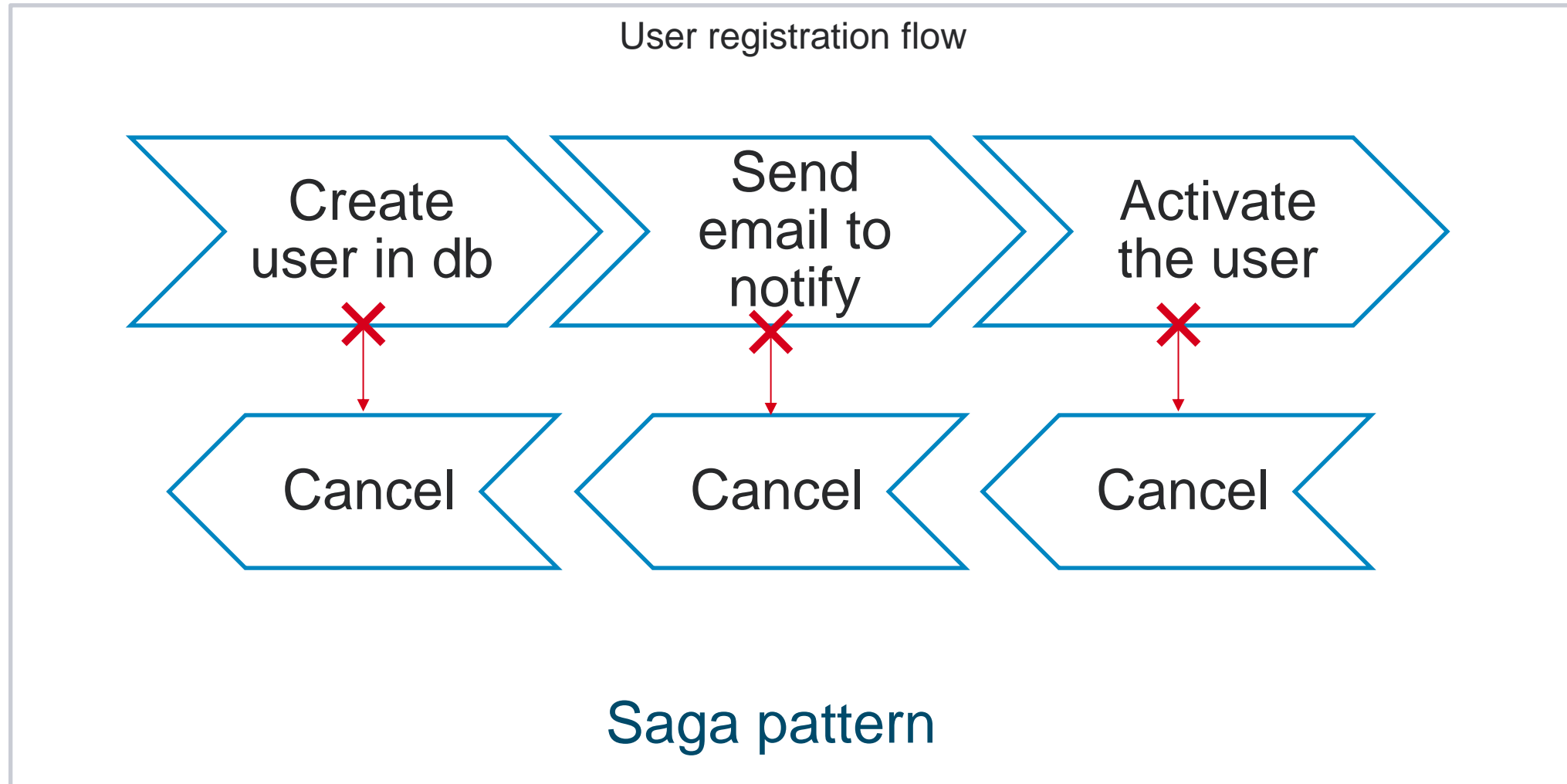
My works on Modular Monolith approach

- <https://github.com/thangchung/modulith-starter-kit>
- <https://github.com/thangchung/coffeeshop-modular>
- <https://github.com/thangchung/coolstore-moduliths>
- <https://github.com/thangchung/modular-starter-kit>
- <https://github.com/thangchung/blog-core>

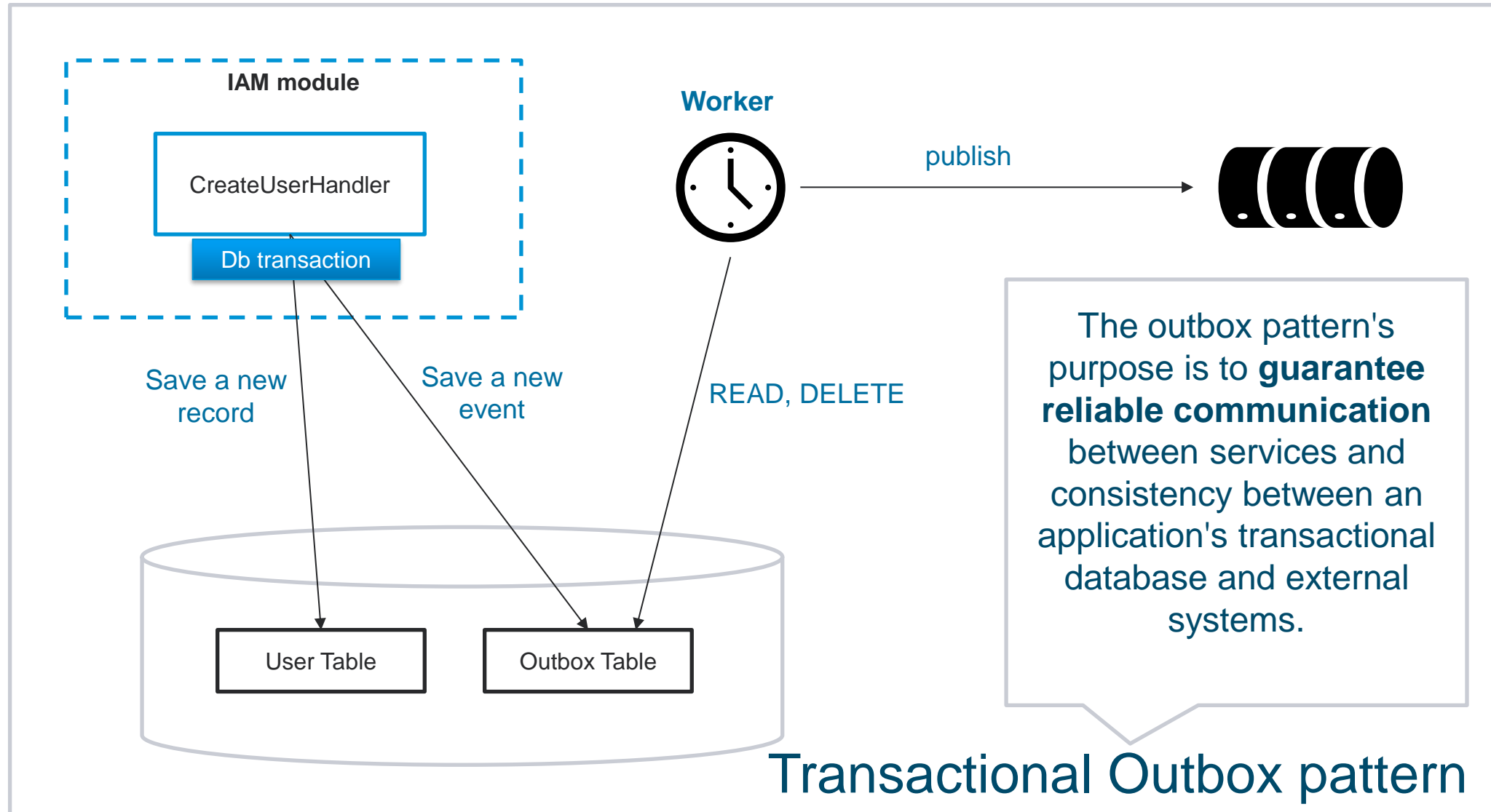
Appendix.



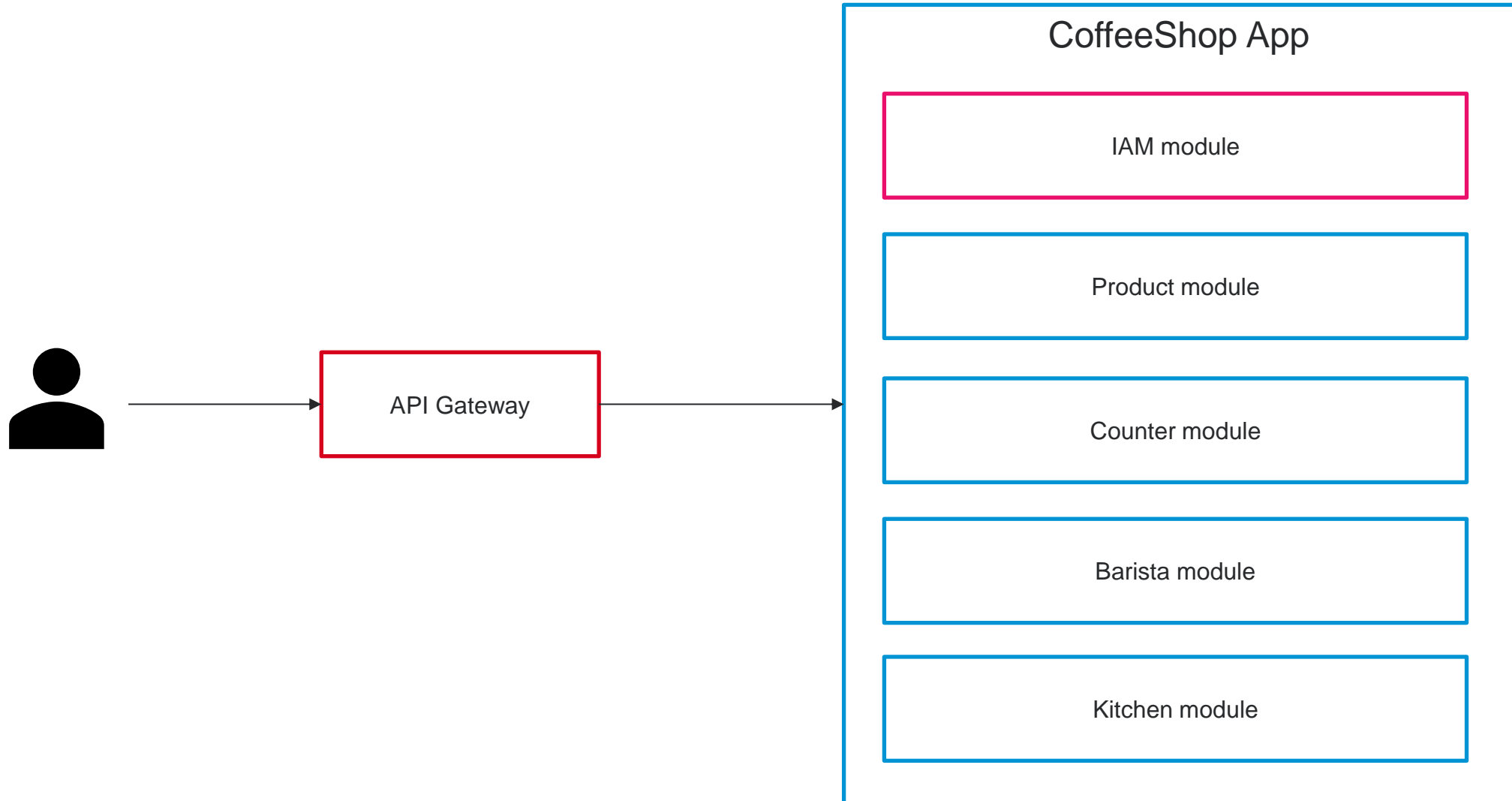
Other architectural patterns (1/3)



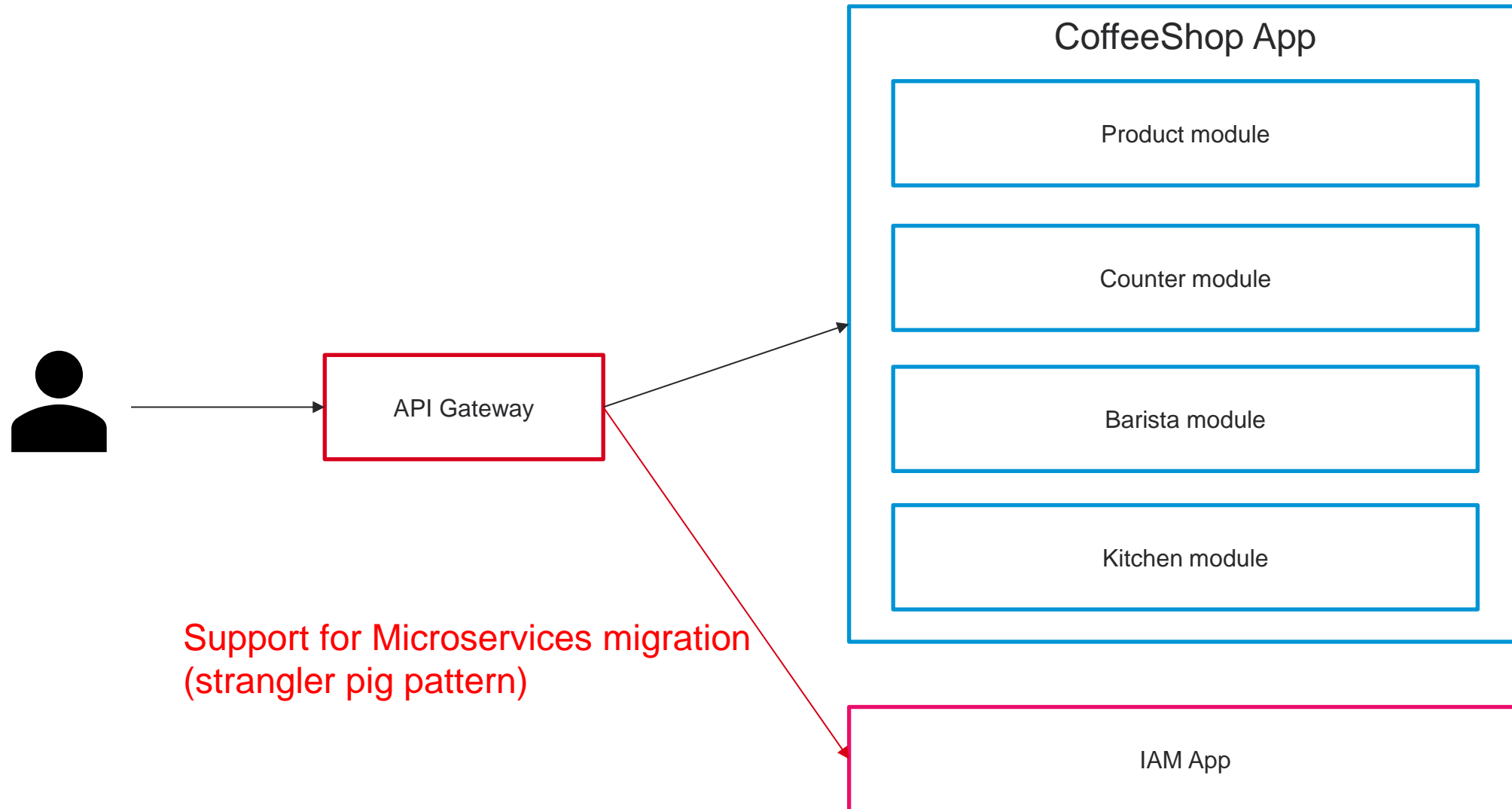
Other architectural patterns (2/3)



Other architectural patterns (3/3)



Other architectural patterns (3/3)



References

- https://www.melconway.com/Home/Conways_Law.html
- <http://www.laputan.org/mud/>
- <https://martinfowler.com/bliki/MonolithFirst.html>
- <https://shopify.engineering/deconstructing-monolith-designing-software-maximizes-developer-productivity>
- <https://ardalis.com/introducing-modular-monoliths-goldilocks-architecture/>
- <https://www.thoughtworks.com/insights/blog/microservices/modular-monolith-better-way-build-software>
- https://files.gotocon.com/uploads/slides/conference_12/515/original/gotoberlin2018-modular-monoliths.pdf
- <https://www.kamilgrzybek.com/blog/posts/modular-monolith-primer>
- <https://speakerdeck.com/ewolff/the-evolution-of-architecture-through-team-topologies>
- <https://alistair.cockburn.us/hexagonal-architecture/>
- <https://jeffreypalermo.com/2008/07/the-onion-architecture-part-1/>
- <https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html>
- <https://www.jimmybogard.com/vertical-slice-architecture/>

Thank you