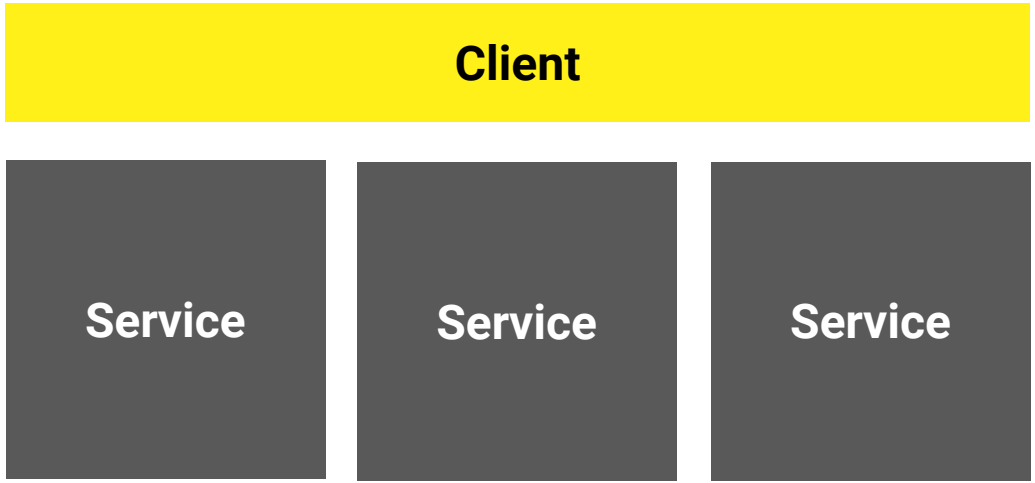
Container Bootcamp

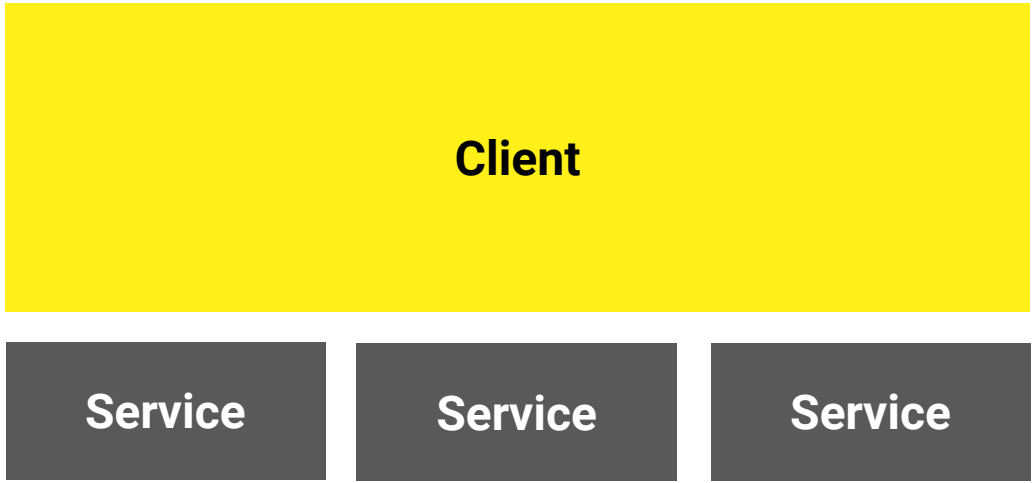
SCS
Self-Contained Systems

Vision



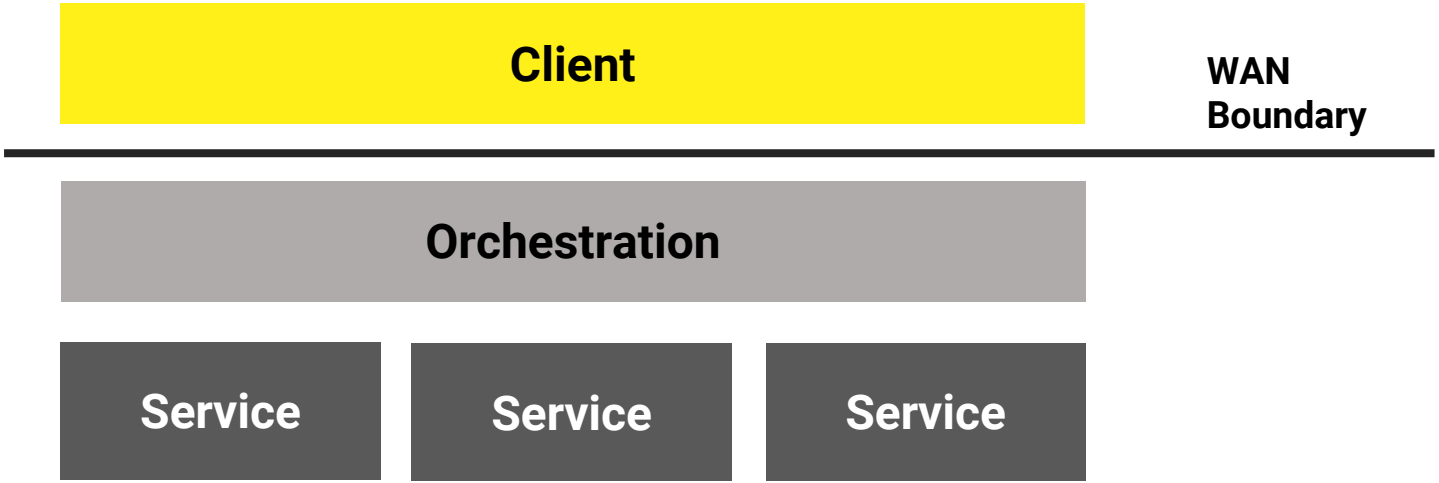
- Services are powerful and reusable
- Building clients is the easy part

Reality



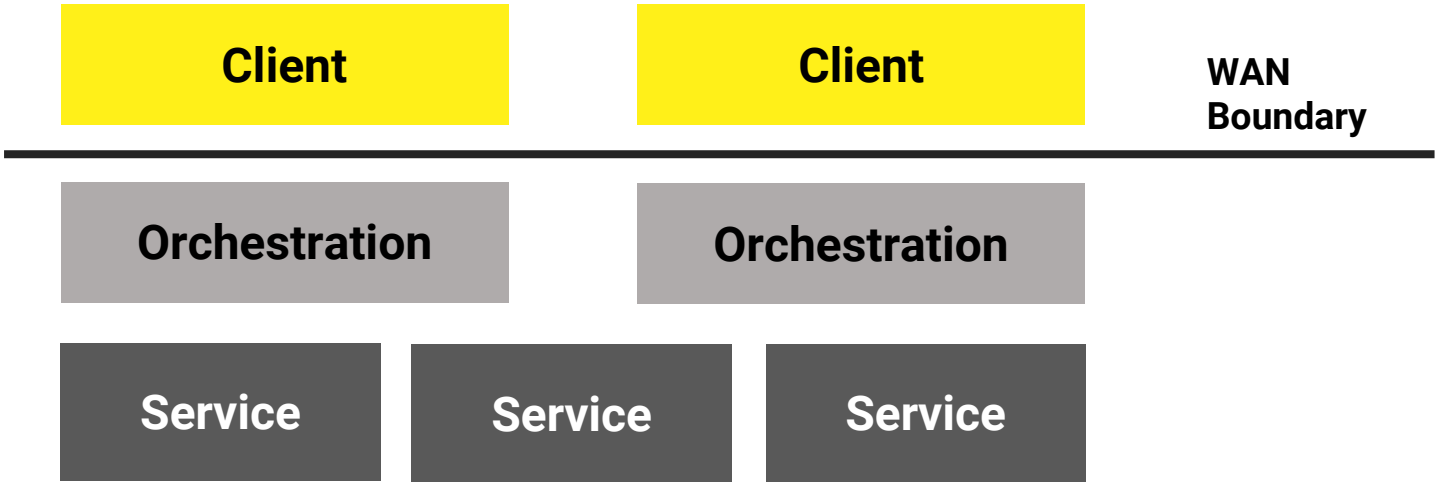
- Services are weak
- Building clients = orchestration: the hard part
- Lot of fine-grained calls

Explicit Orchestration



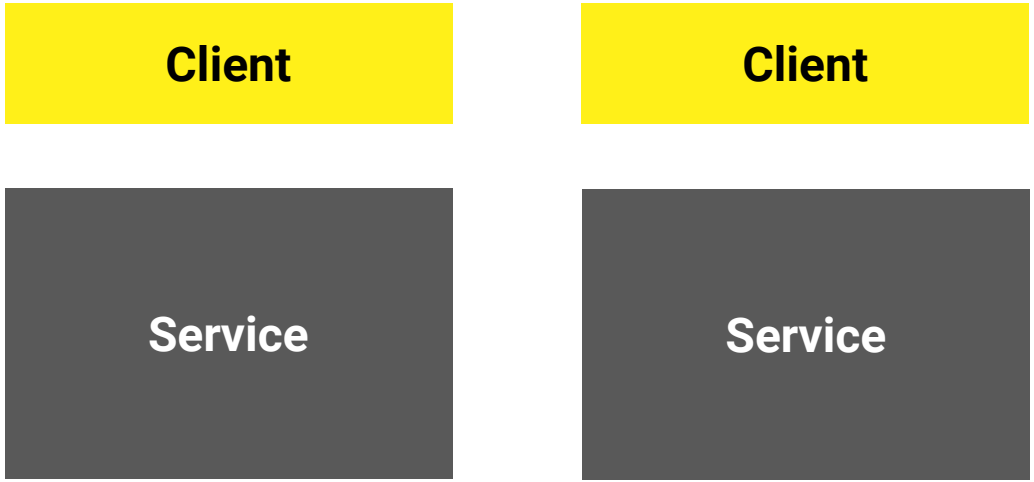
- Higher-level services
- Bundle fine-grained calls

Client-Specific Orchestration



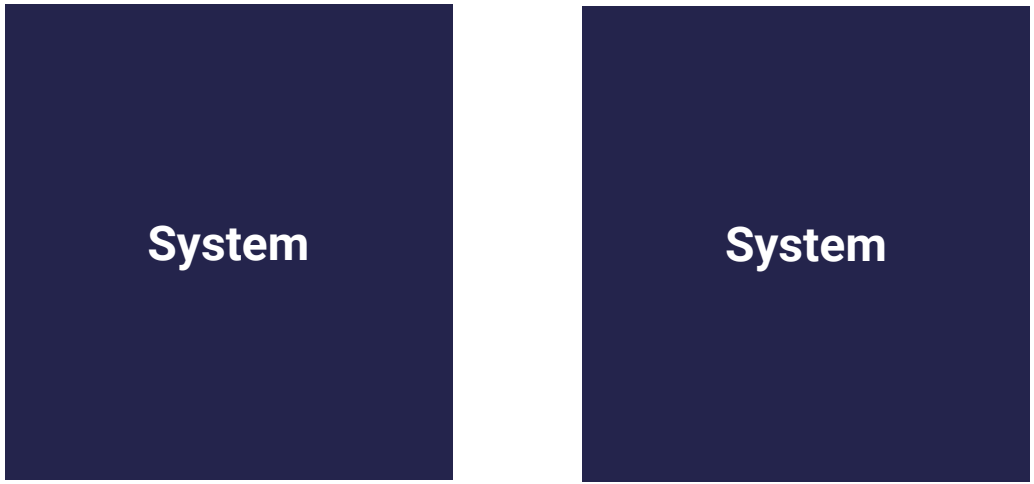
- Fine-tune for client needs
- Explicit & specific
- Partially redundant

Higher-Level Services



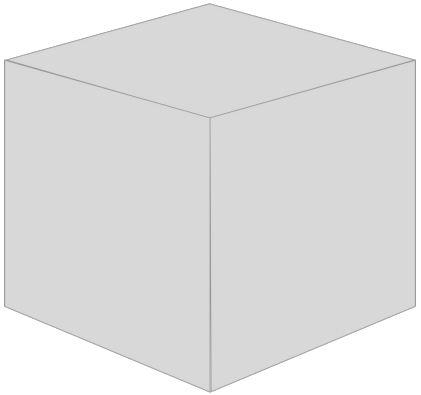
- Designed to include process up front
- Shift view to vertical

System View



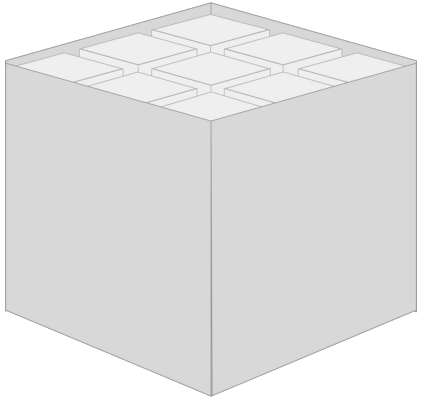
- Systems as complete units of DB, Logic, UI
- Isolated, independent, autonomous

Self-Contained System (SCS)

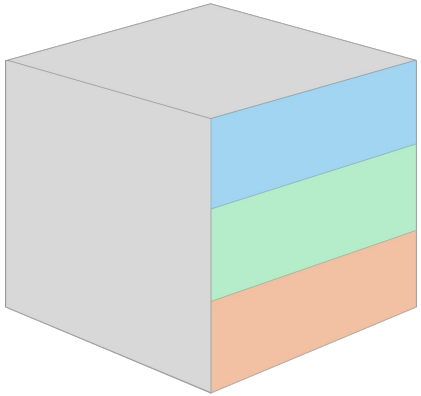


Deployment monolith

Graphics by Roman Stranghöhner, INNOQ
http://scs-architecture.org



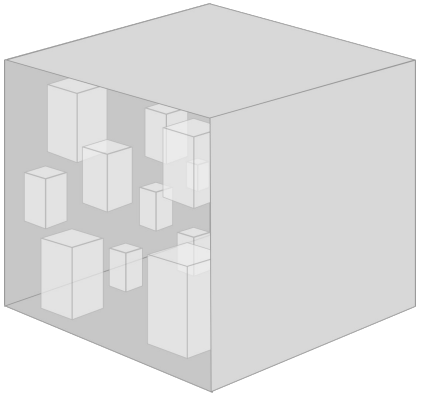
Various Domains



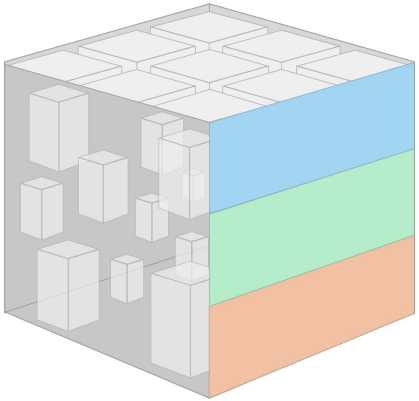
User interface

Business logic

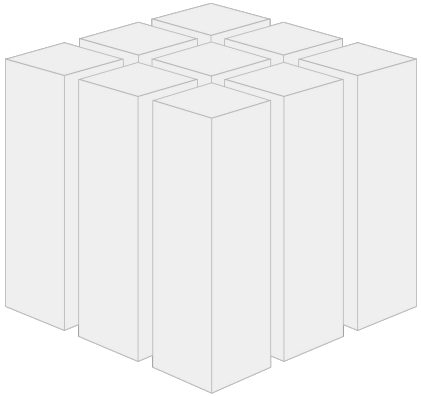
Persistence



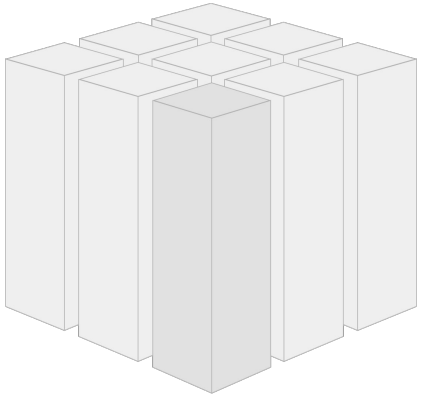
… a lot of modules, components, frameworks and libraries



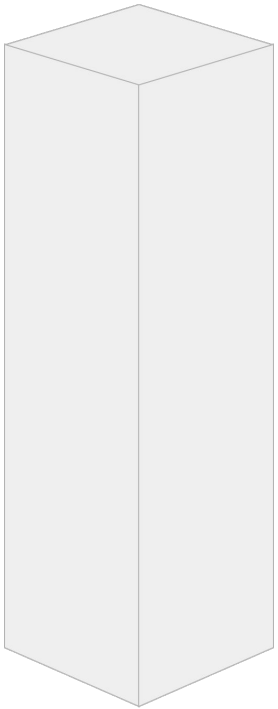
With all these layers in one place, a monolith tends to grow.



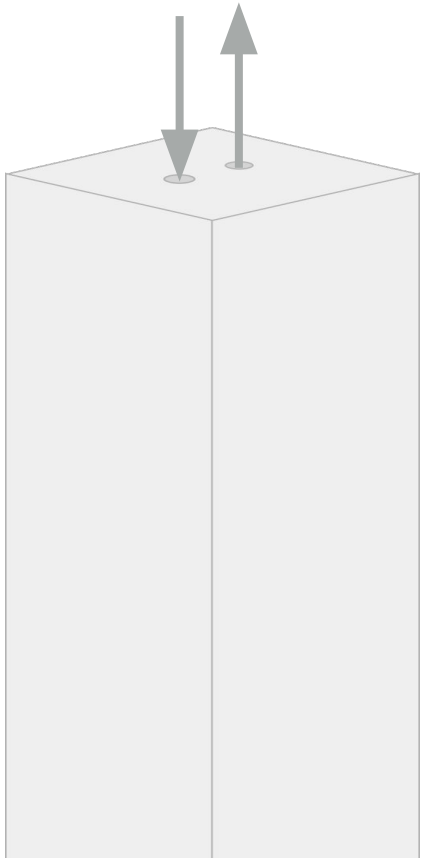
Cut Deployment monolith along domains …



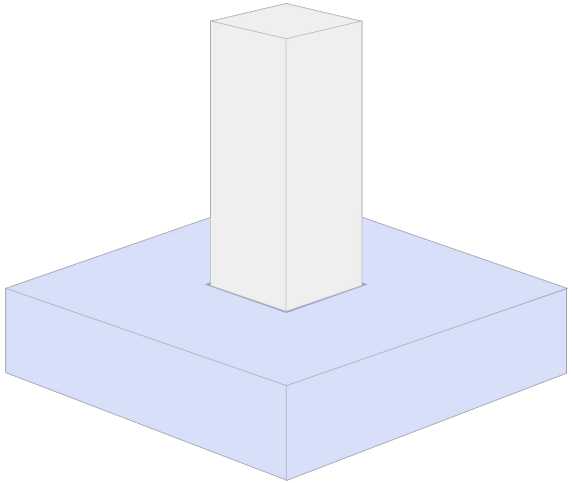
… wrap domain in separate web application …



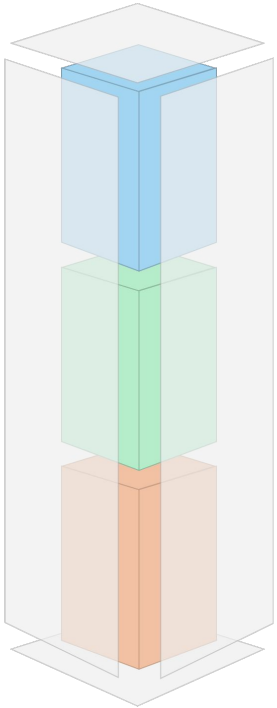
Self-contained System (SCS) – individually deployable



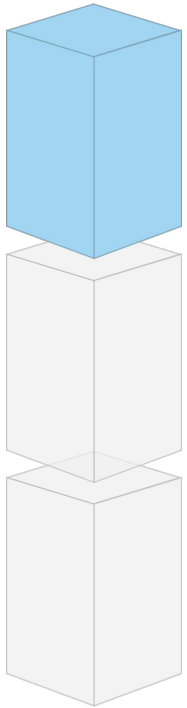
Decentralized unit communicating with other systems via RESTful HTTP or lightweight messaging.



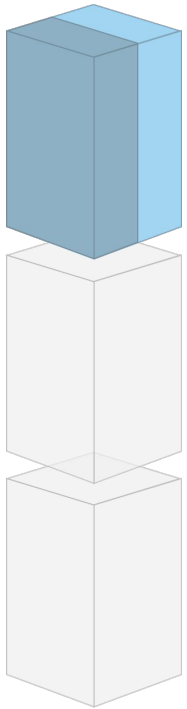
SCS can be individually developed for different platforms.



An SCS contains its own user interface, specific business logic and separate data storage

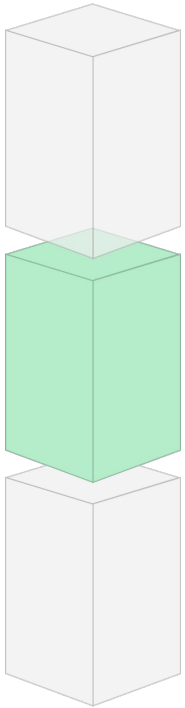


Web user interface composed according to ROCA principles.

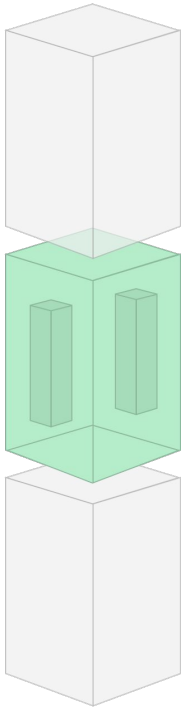


optional API e.g.

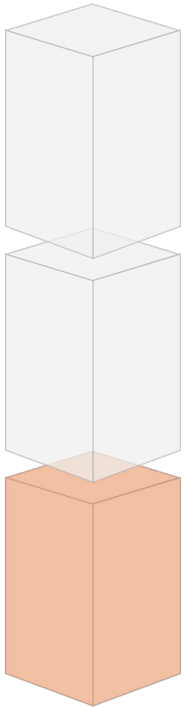
for mobile



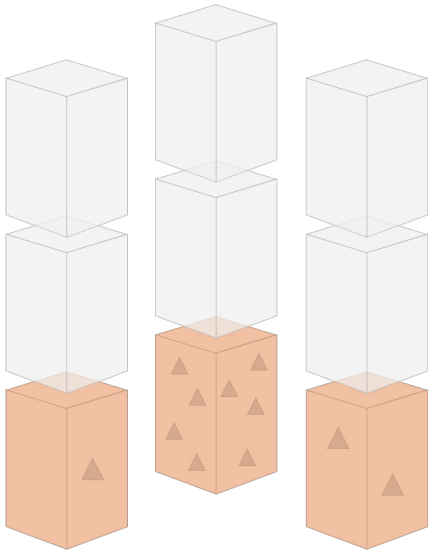
Logic only shared over a well defined interface.



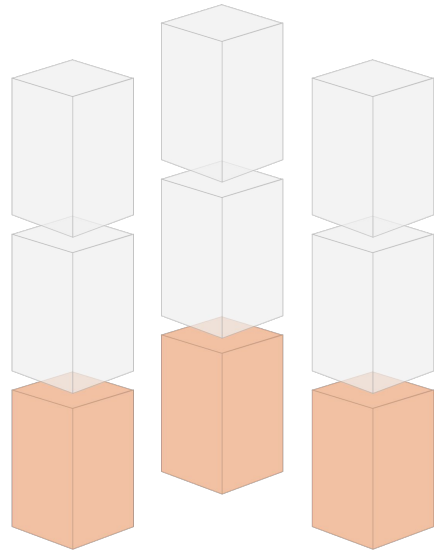
Business logic can consist of microservices



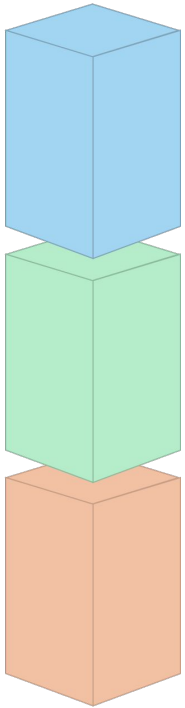
Every SCS brings its own data storage with its own (potentially redundant) data



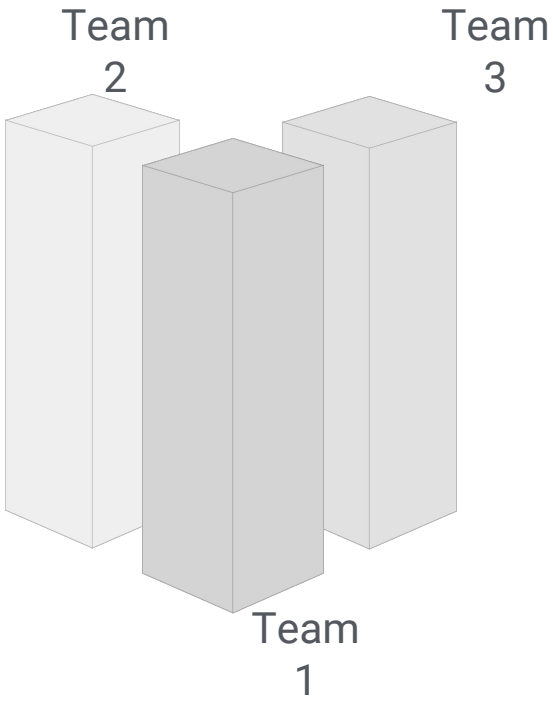
Redundancies are tolerable as long as the sovereignty of data by its owning system is not undermined.



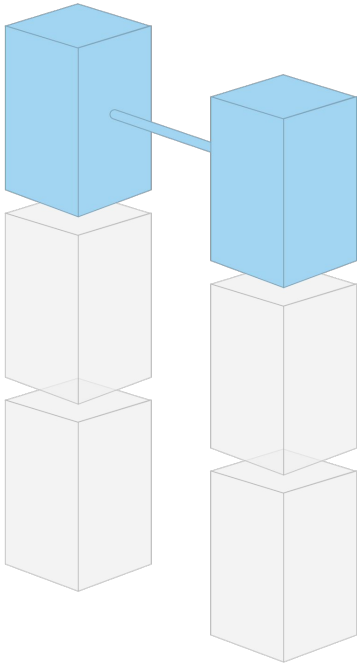
Enables polyglot persistence



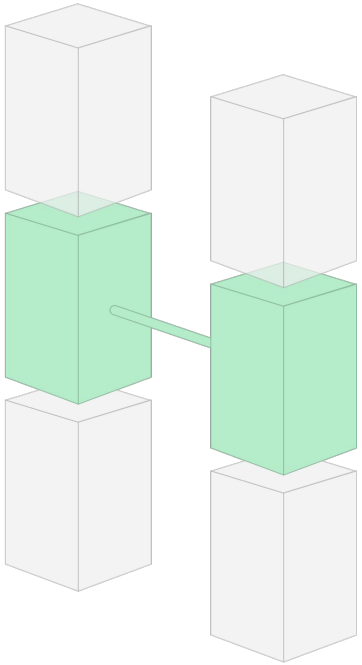
Technical decisions can be made independently from other systems (programming language, frameworks, tooling, platform)



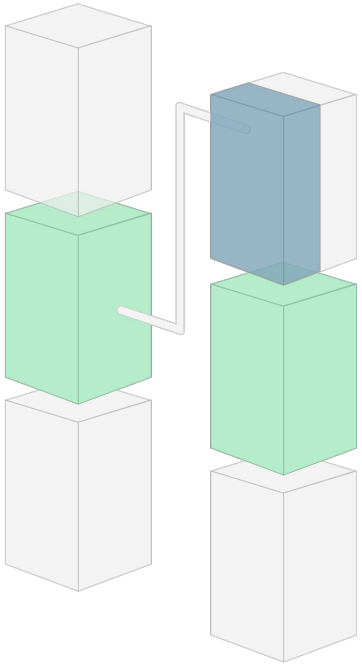
The manageable domain specific scope enables the development, operation and maintenance of an SCS by a single team.



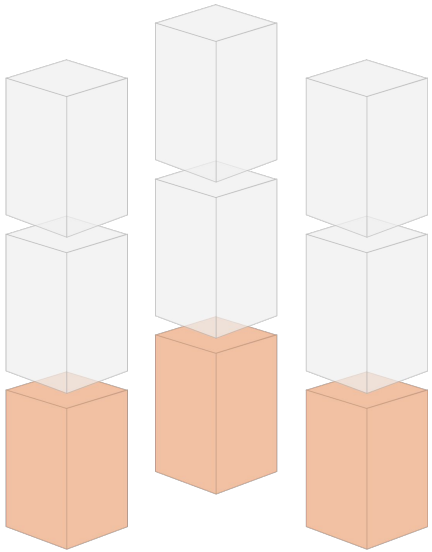
Self-contained Systems should be integrated in the web interface



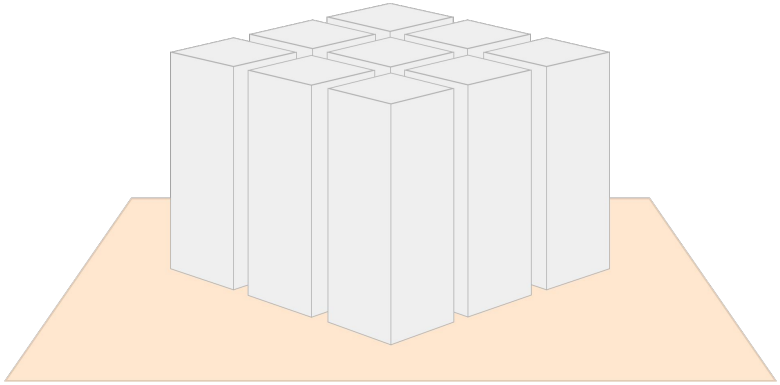
Synchronous remote calls inside the business logic should be avoided.



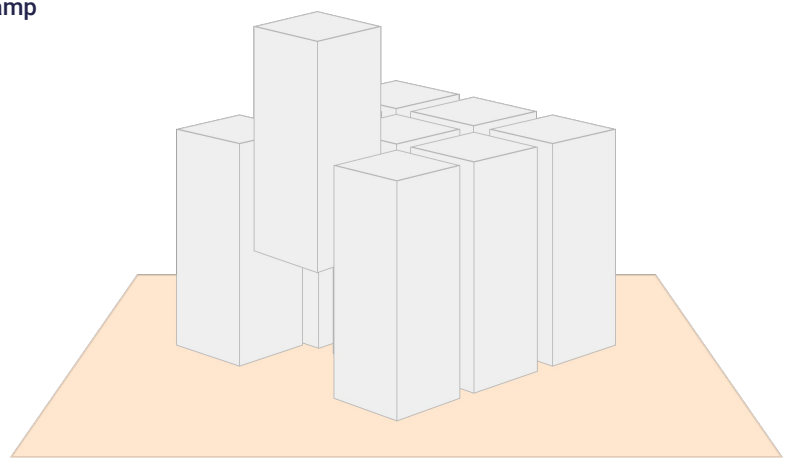
Remote API calls should be handled asynchronously to reduce dependencies and prevent error cascades.



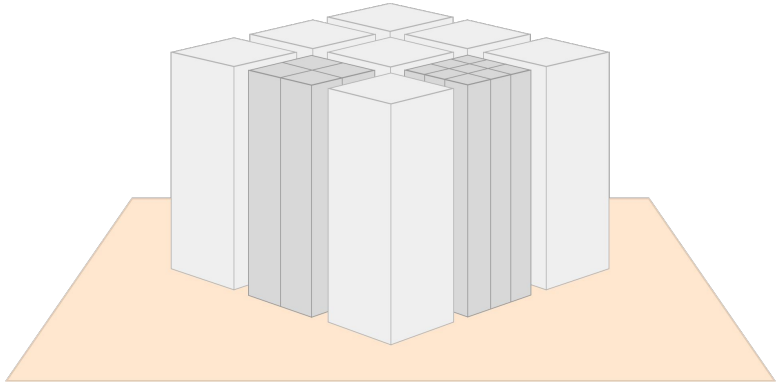
Implies replication & data model's consistency guarantees are relaxed.



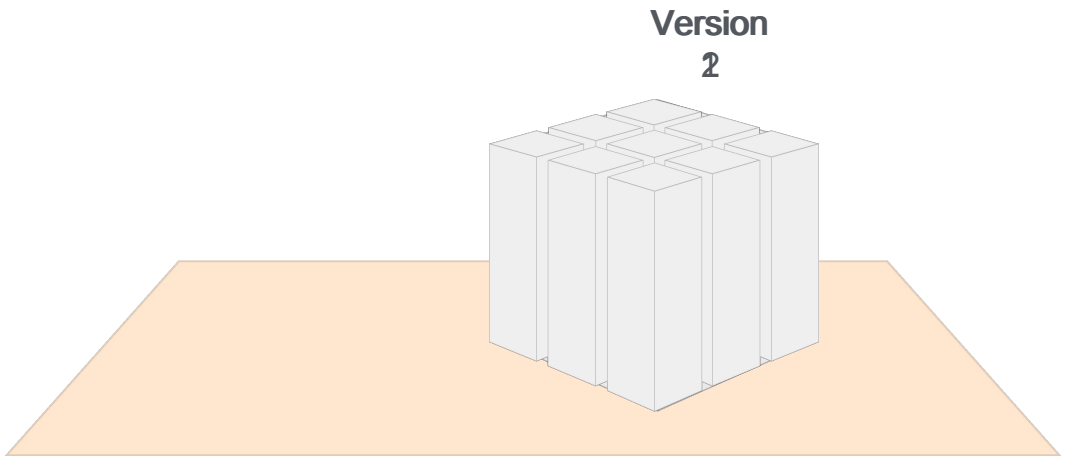
An integrated
system of systems
like this has many benefits.



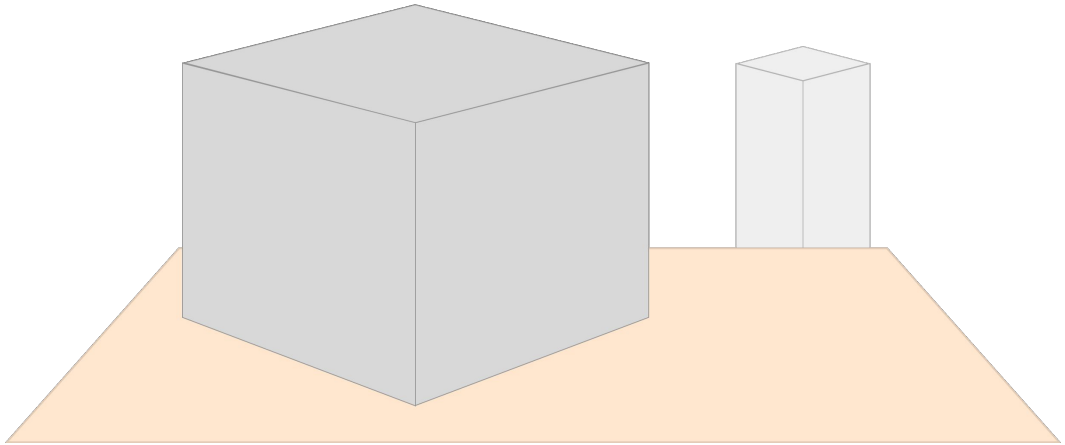
Overall resilience is improved through loosely coupled, replaceable systems.



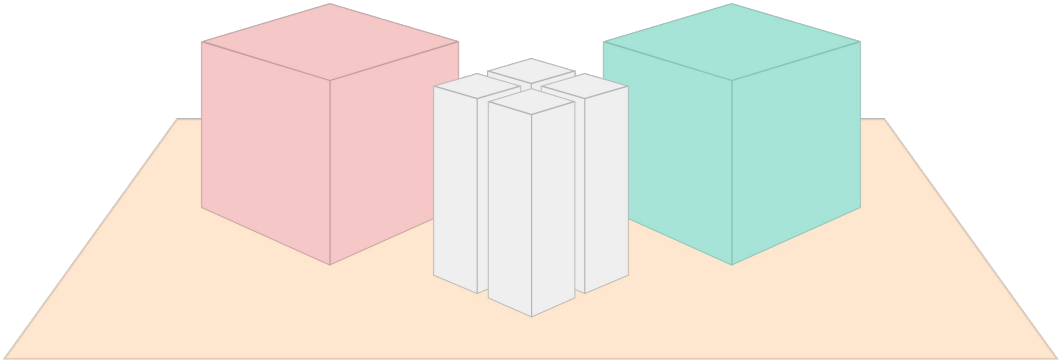
Some systems can be individually scaled to serve varying demands.



It is not necessary to perform a risky big bang release to migrate an outdated, monolithic system into a system of systems.



Migration can happen in small, manageable steps which minimize risk of failure and lead to an evolutionary modernization of big and complex systems.



In reality a system of systems consists of individually developed software and standard products.

Microservice Characteristics

- small
- each running in its own process
- lightweight communication mechanism
- built around business capabilities
- independently deployable
- minimum of centralized management
- may be written in different programming languages
- may use different data storage technologies

http://martinfowler.com/articles/microservices.html

SCS Characteristics

- Autonomous web application
- Owned by one team
- No sync remote calls
- Service API optional
- Includes data and logic
- No shared UI
- No or pull-based code sharing only

SCS ≡ Microservice

SCS —▷ Microservice

SCS 1 . . * Microservice

Conclusion

- **Self-contained systems are microservices …**
  - **that are not “micro”**
  - **and don’t have to be “services”**