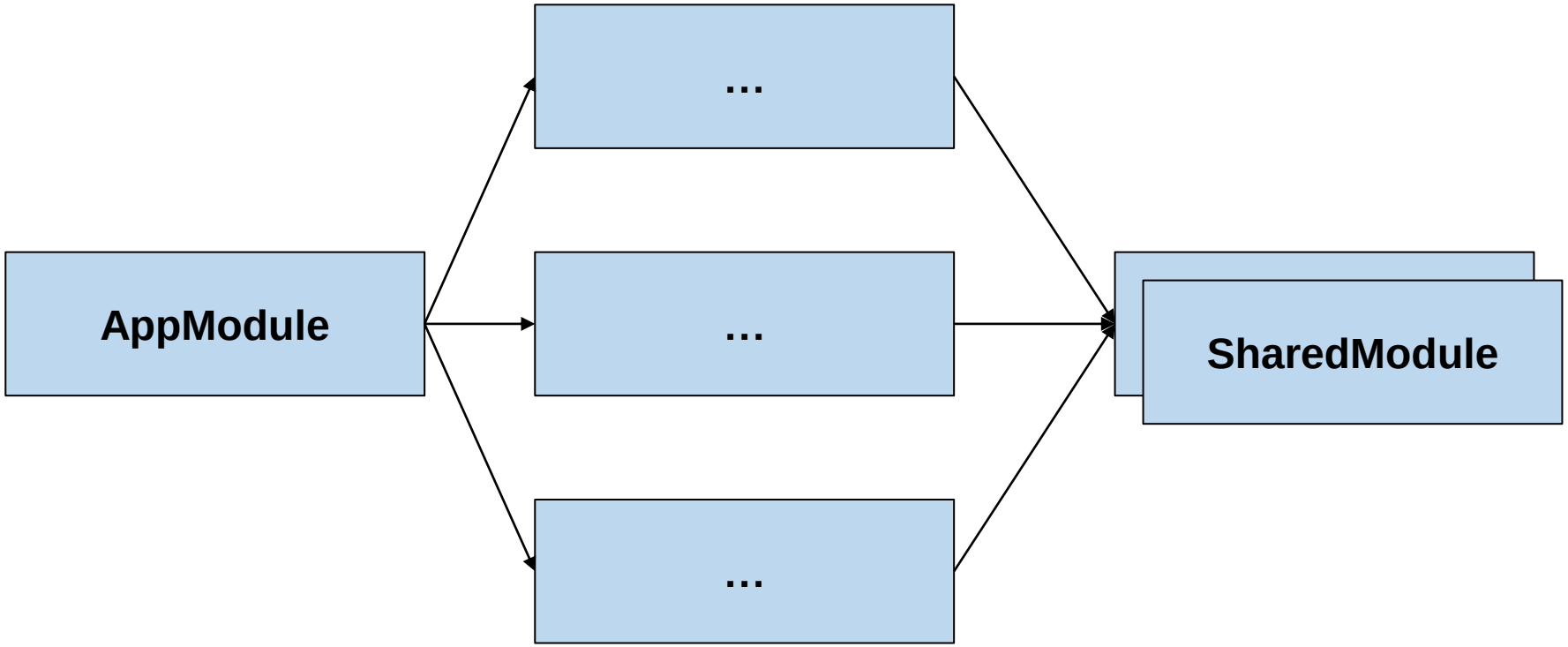
ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Architectures for huge Enterprise Applications with Angular

angular-architects.io

Typical Module Structure



Root Module **Feature Modules** **Shared Modules**

Contents

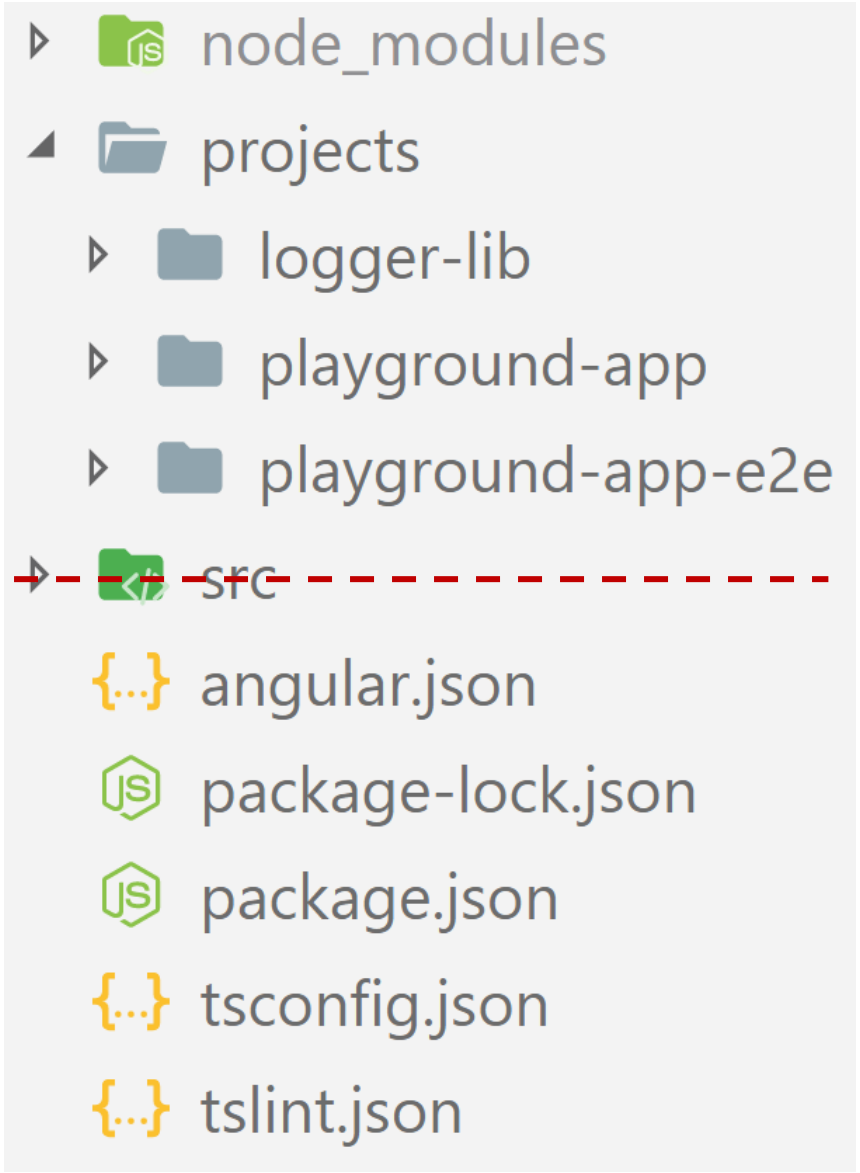
- (npm-)Packages
- Nx Monorepos
- Strategic Design and DDD

npm Packages

Create Library with CLI >= 6

```
npm install -g @angular/cli

ng new lib-project

cd lib-project

ng generate library logger-lib
ng generate application playground-app

ng serve --project playground-app
ng build --project logger-lib
```

Folder Structure



Create Library with CLI >= 6

```
npm install -g @angular/cli

ng new lib-project --create-application false

cd lib-project

ng generate library logger-lib
ng generate application playground-app

ng serve --project playground-app
ng build --project logger-lib
```


Leerungszeiten
Deutsche Post
Tagesleerungen
Spätleerung
Nachtleerung
Montag - Freitag
08:00
Samstag
08:00
Sonntag
Service-Telefon der Briefkastenleerung:
0228 4333112
Briefkästen mit späteren Leerungen:
letzte Tagesleerung
Spätleerung
Nachtleerung
Montag - Freitag
Samstag
Sonntag

Publishing

Publishing to npm Registry

- Increment version in package.json
- ng build logger-lib --prod
- npm publish *dist/logger-lib* --registry http://localhost:4873
- npm install logger-lib --registry http://localhost:4873

Alternatives for setting the Registry

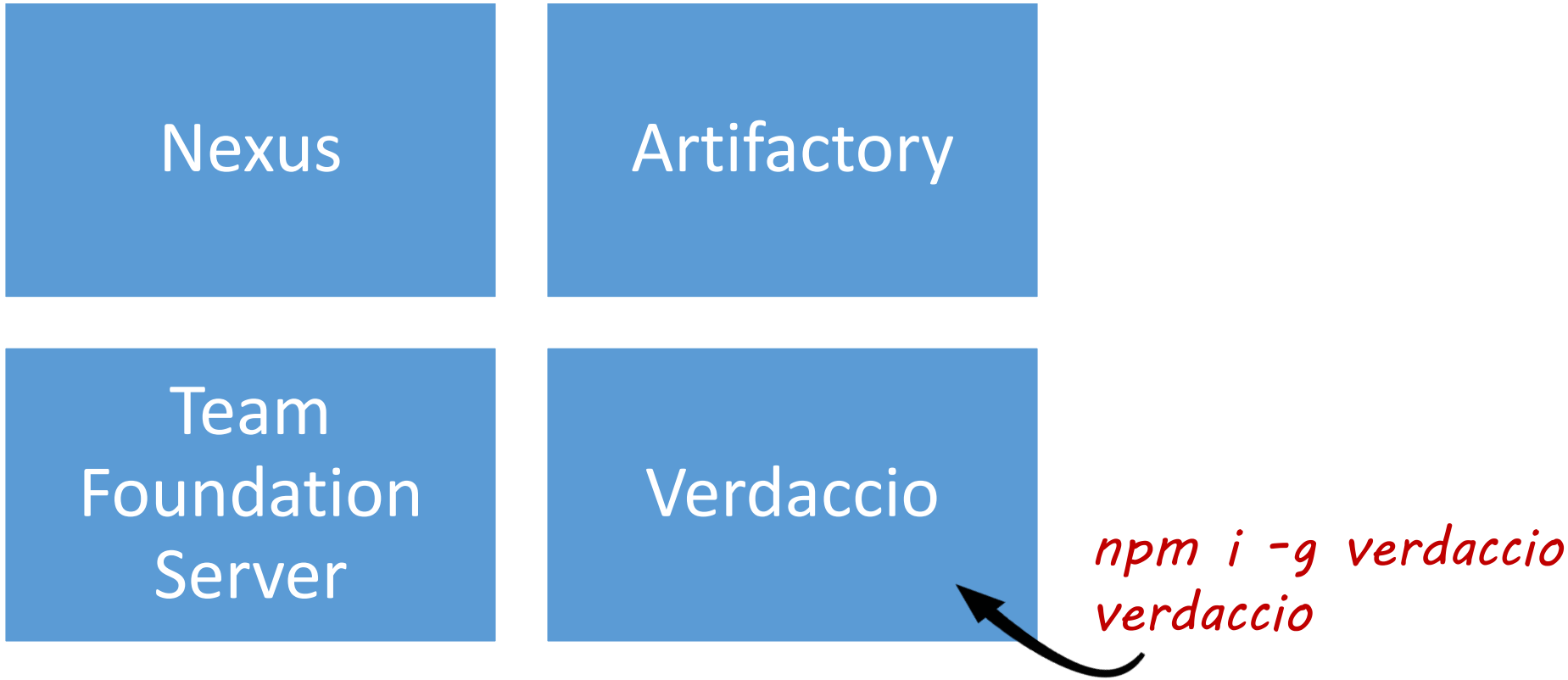
- Global: npm set registry http://localhost:4873
  - Default: registry.npmjs.org
  - npm get registry

- Project: .npmrc in project root

```
registry=http://localhost:4873/
```

```
@my-company:registry=http://my-server:4873/
```

npm Registries



DEMO

Advantages

- Distribution
- Versioning

Disadvantages

- Distribution
- Versioning

;-)

Disadvantages

Distribution

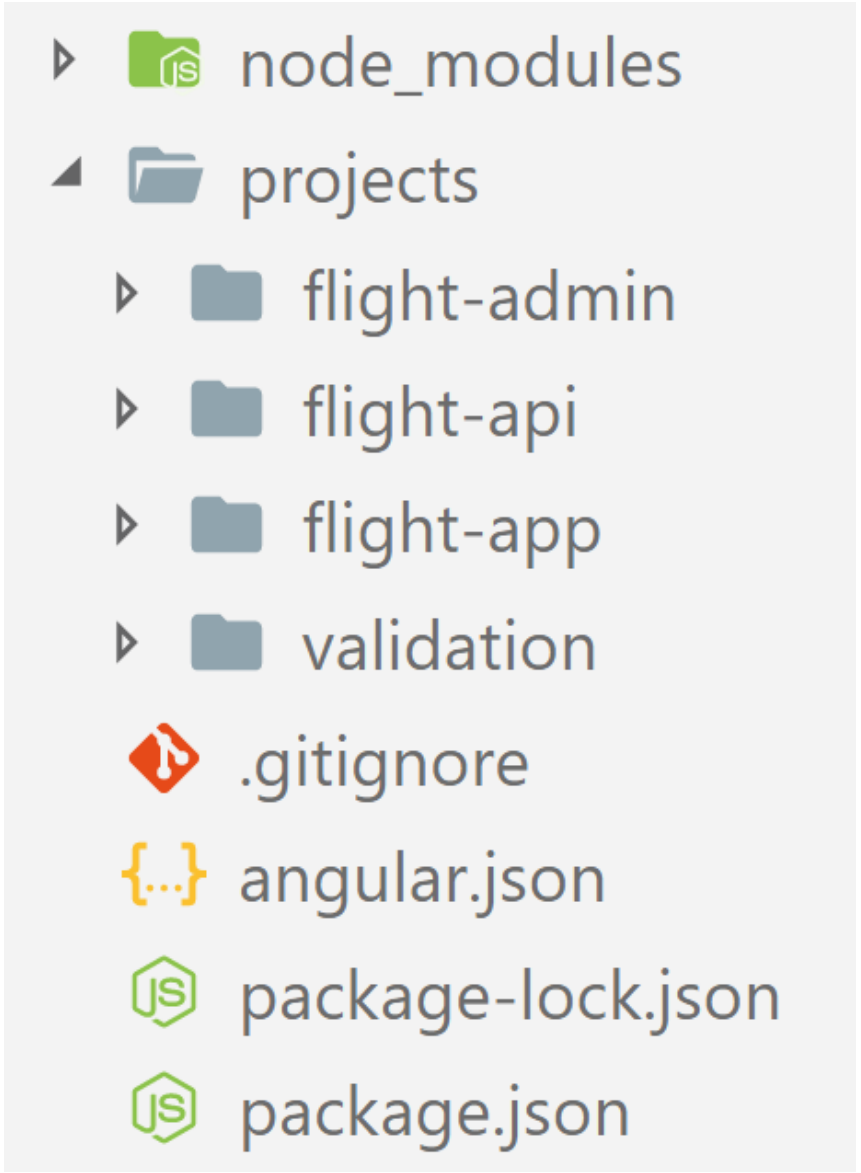
- Annoying within project
- Prevents gritting further libs

Versioning

- Old versions
- Conflicts
- How to force devs to use latest version?

Monorepos

Monorepo Structure



Advantages

- Everyone uses the latest versions
- No version conflicts
- No burden with distributing libs
- Creating new libs: Adding folder
- Experience: Successfully used at Google, Facebook, …

Two Flavors

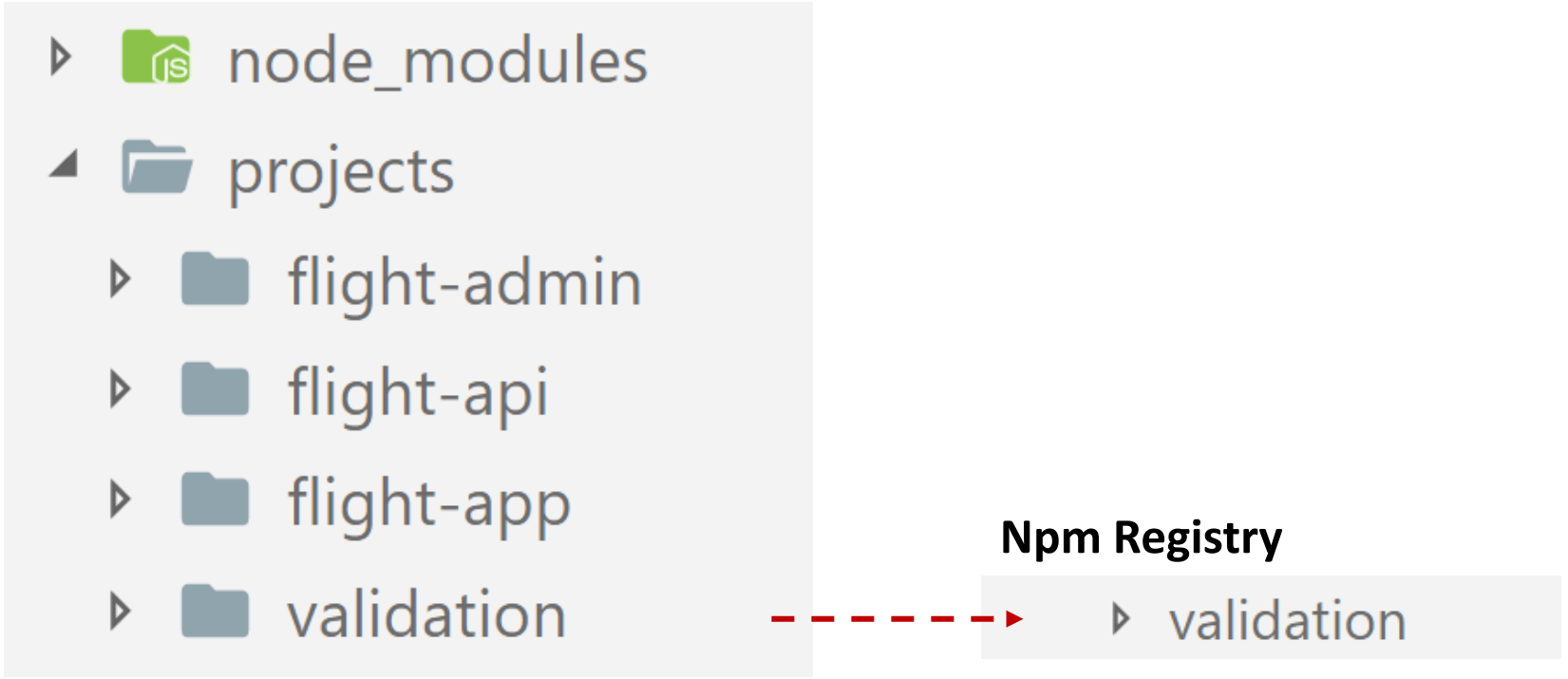
Project Monorepo

- Like Workspaces/Solutions in different IDEs

Company-wide Monorepo

- E. g. used at Google or Facebook

Moving back and forth



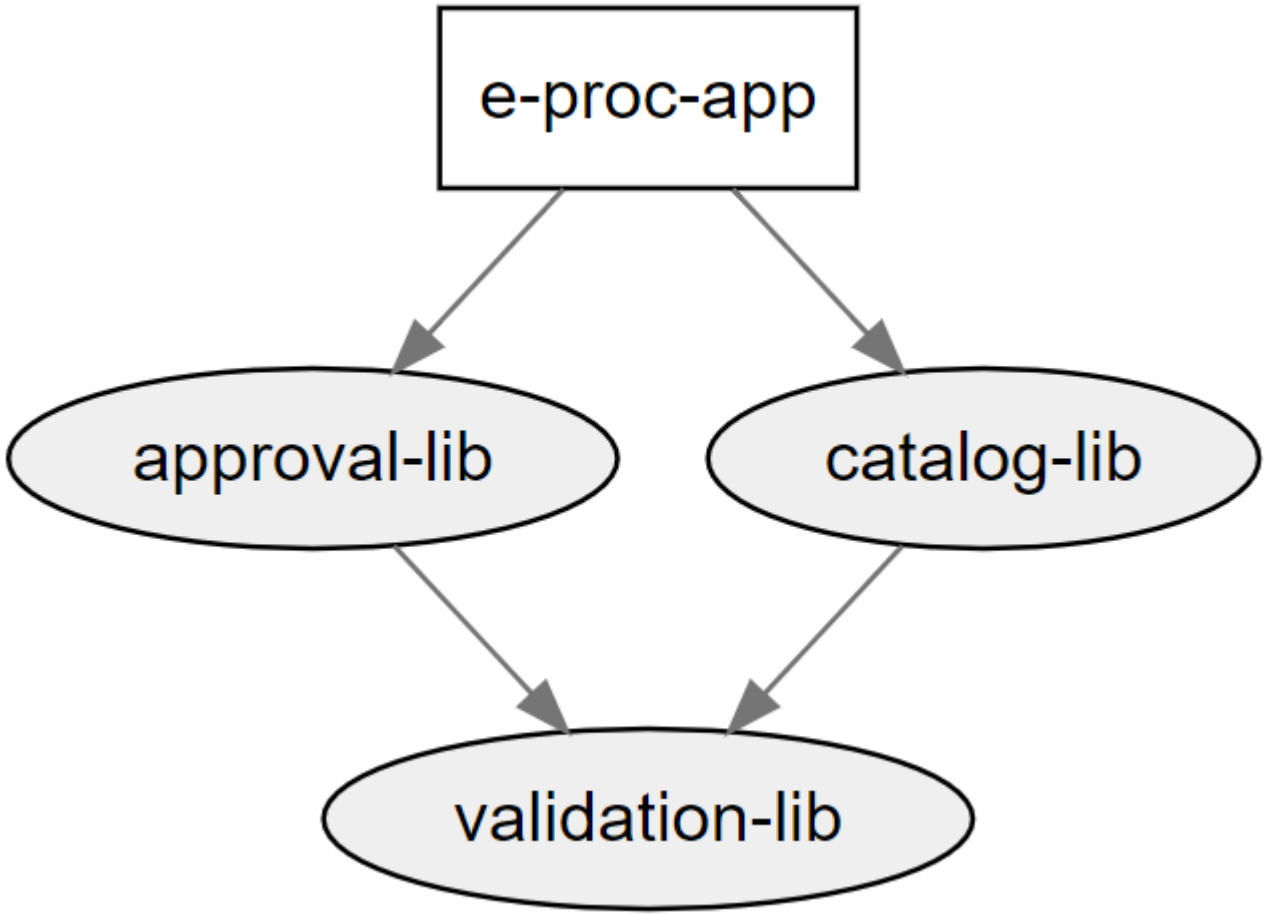
Tooling & Generator

https://nrwl.io/nx

Nrwl Extensions for Angular

An open source toolkit for enterprise Angular applications.

Visualize Module Structure



Creating a Workspace

```
npm install -g @angular/cli

ng new workspace

cd workspace

ng generate app my-app
ng generate lib my-lib

ng serve --project my-app
ng build --project my-app
```

Creating a Workspace

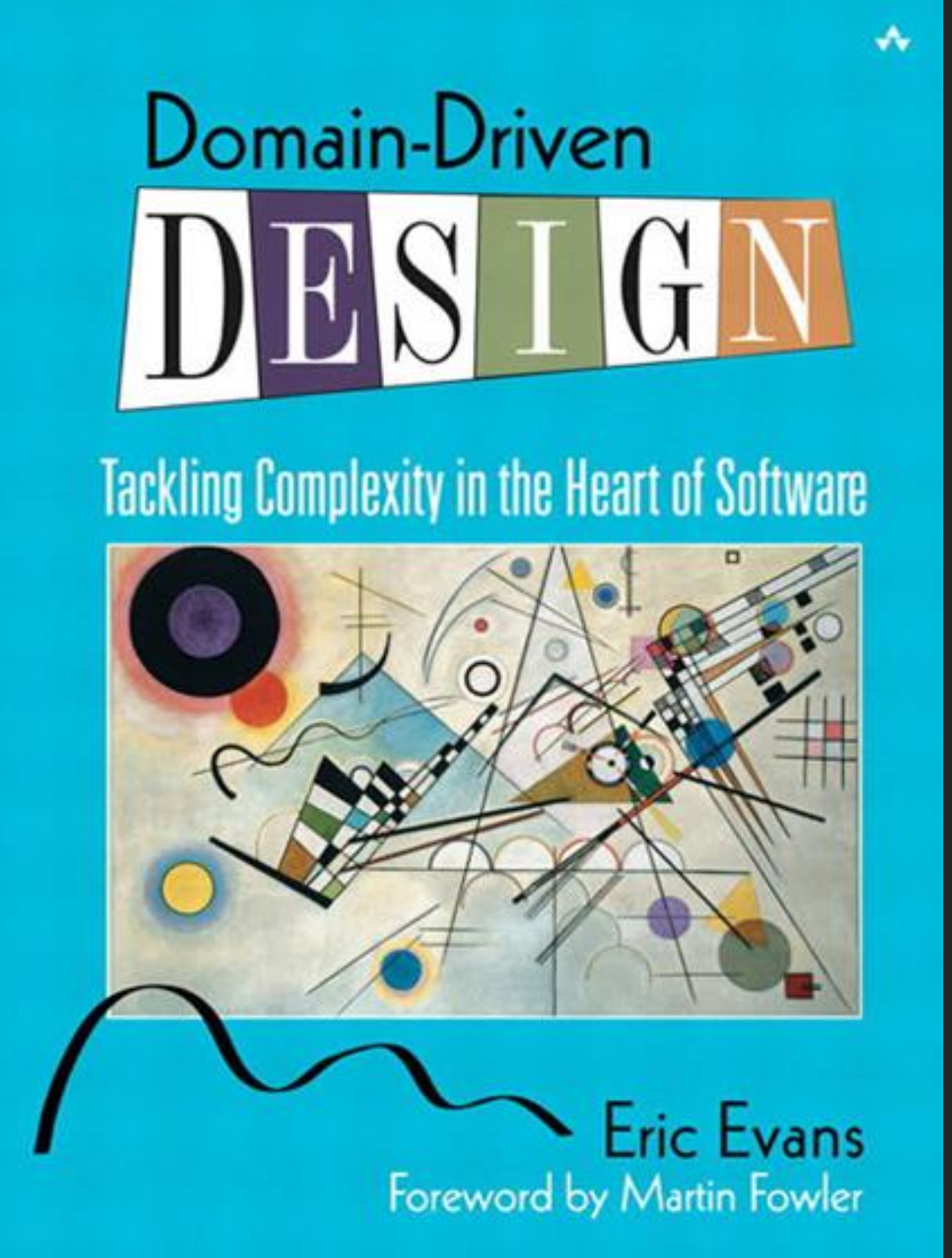
```
npm install -g @angular/cli

npm init nx-workspace workspace

cd workspace

ng generate app my-app
ng generate lib my-lib --buildable

ng serve --project my-app
ng build --project my-app
```

DEMO

LAB

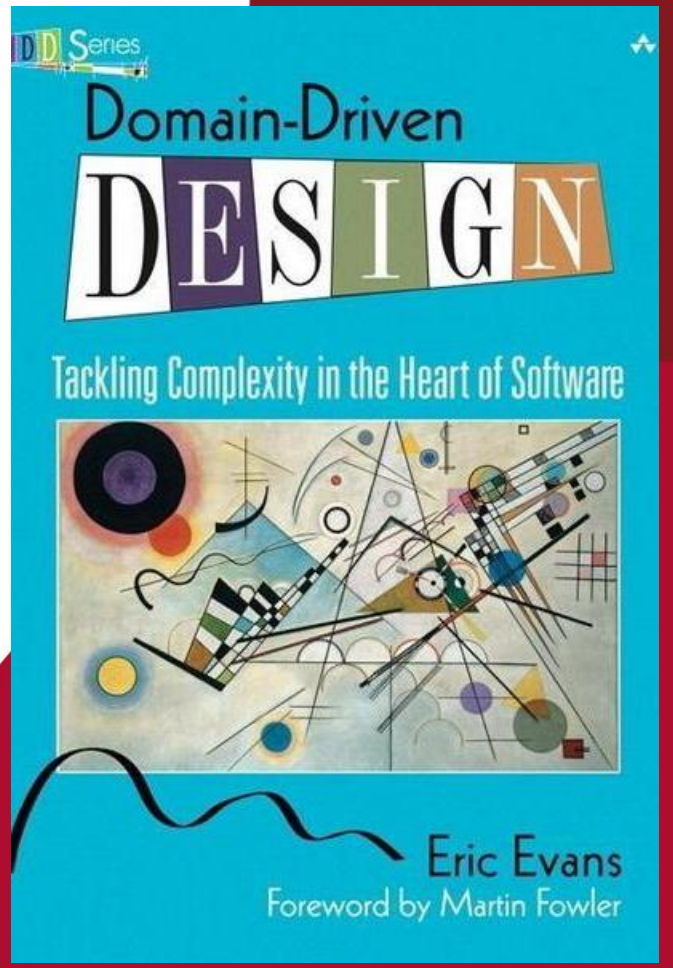
DDD

in a nutshell



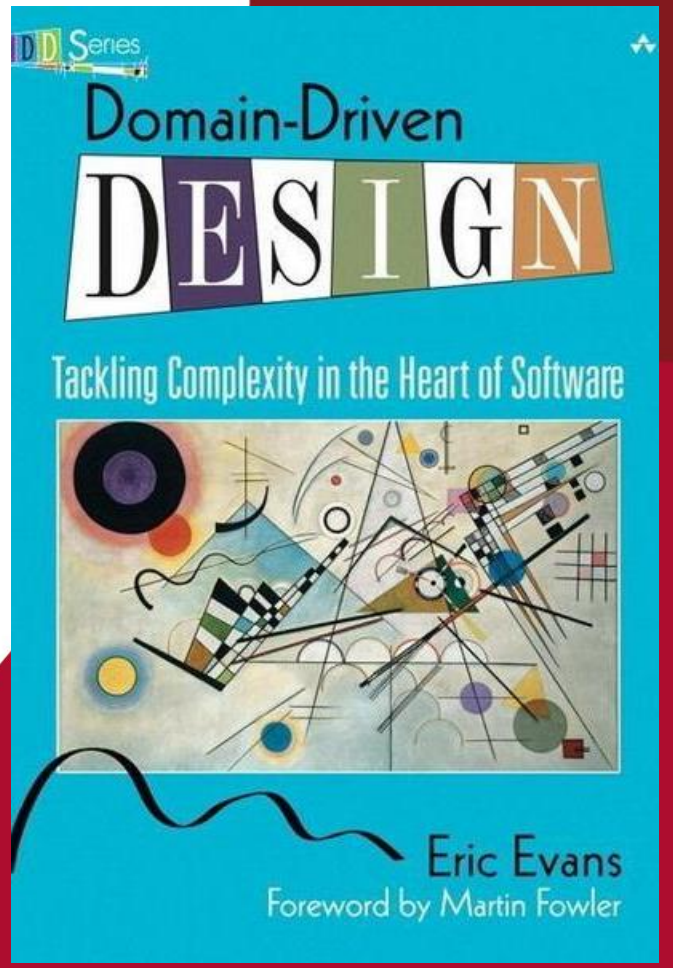
Methodology for bridging the gap b/w requirements and architecture/ design

How to create sustainable frontend architectures with ideas from DDD?



ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

How to create **sustainable** frontend architectures with **ideas from DDD**?

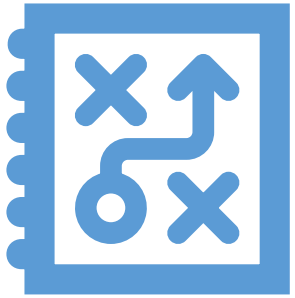


ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

Domain Driven Design

Decomposing a System

Design Patterns
& Practices



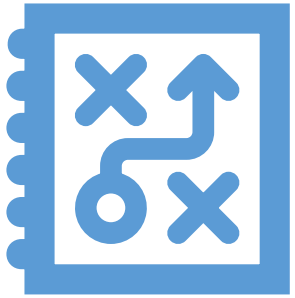
Strategic Design

Tactical Design

Domain Driven Design

Decomposing a System

Design Patterns
& Practices



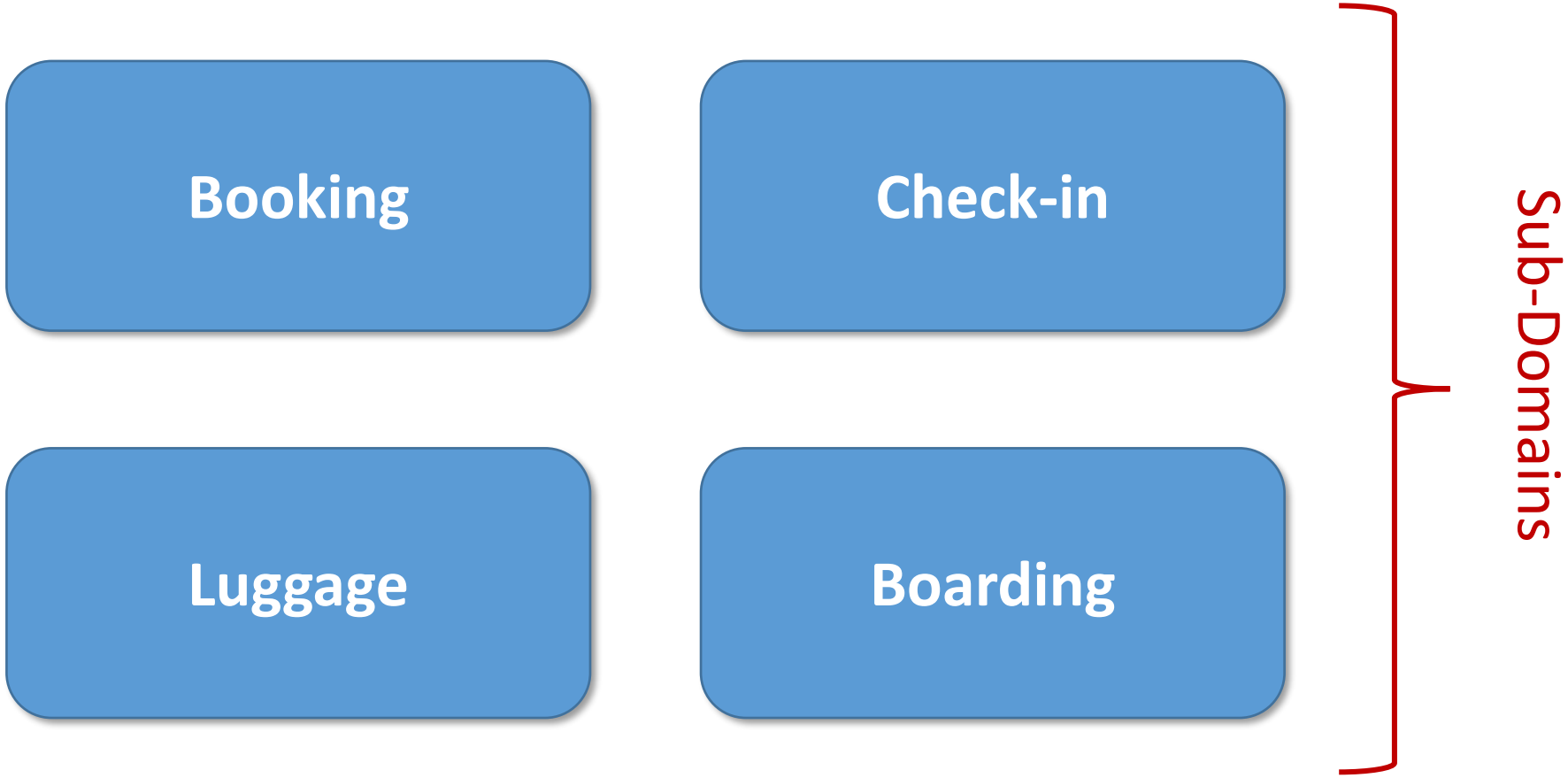
Strategic Design

Tactical Design

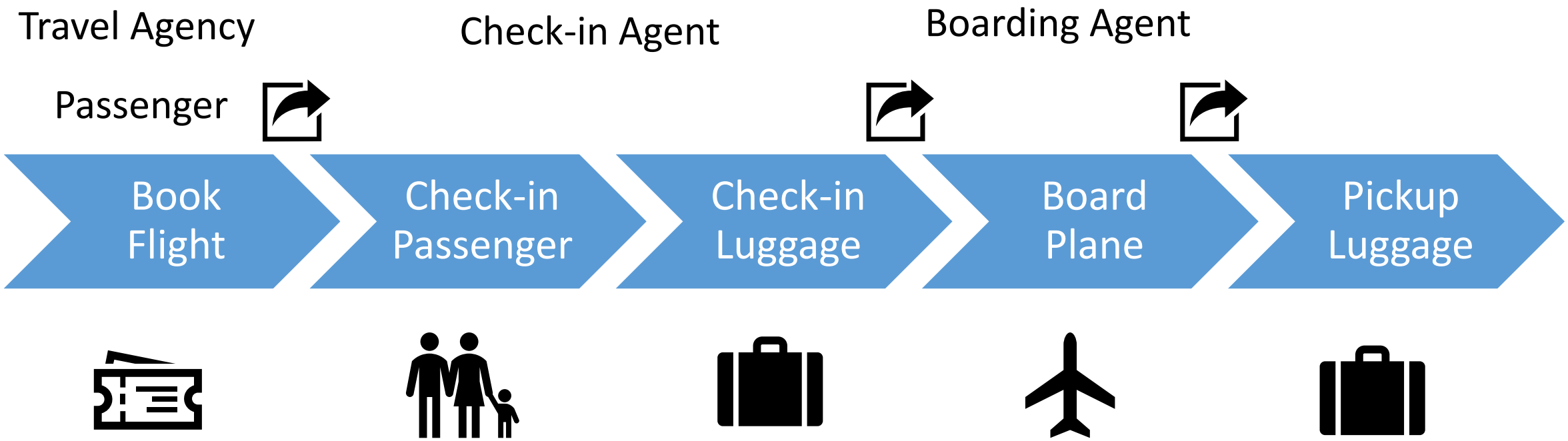
This is what Strategic DDD prevents

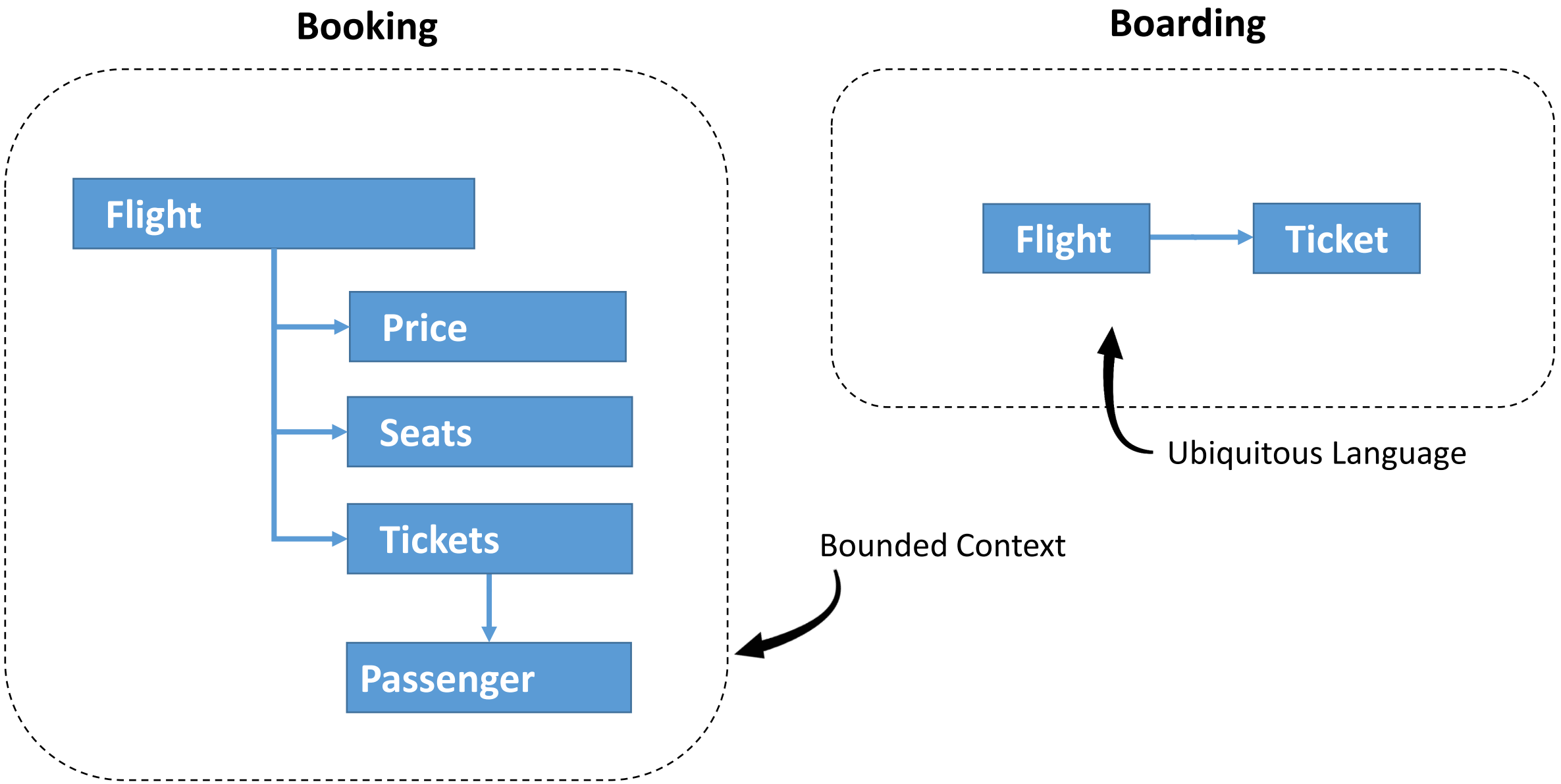
Example

Example



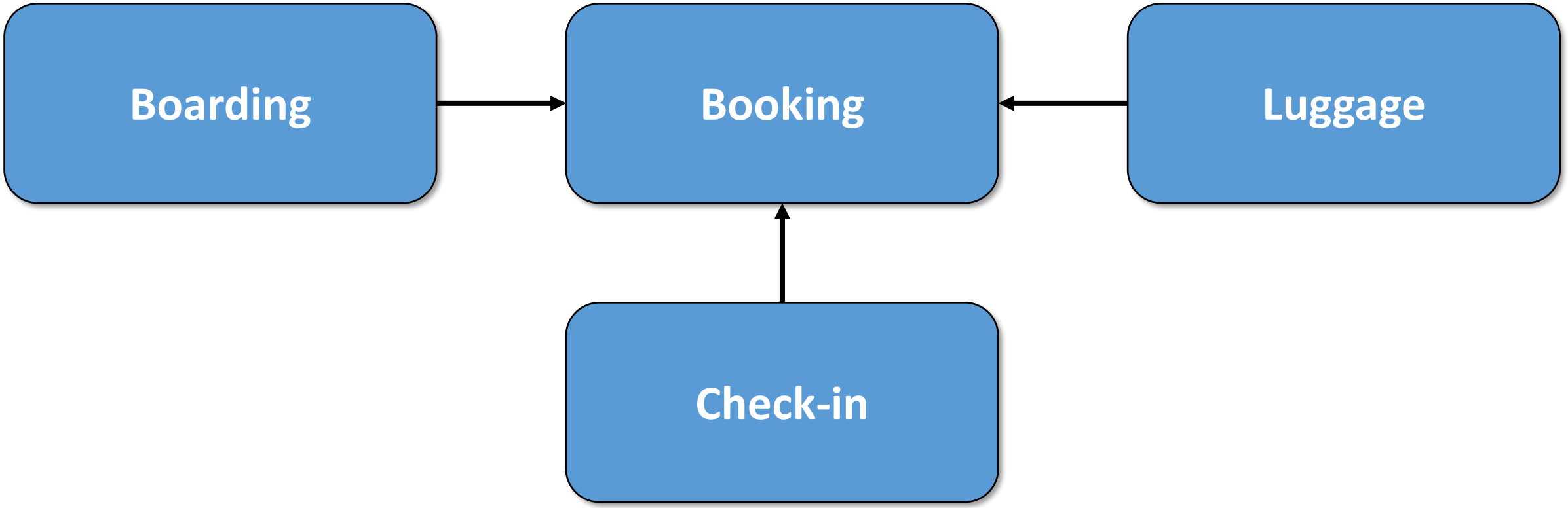
Finding Sub-Domains





@ManfredSteyer

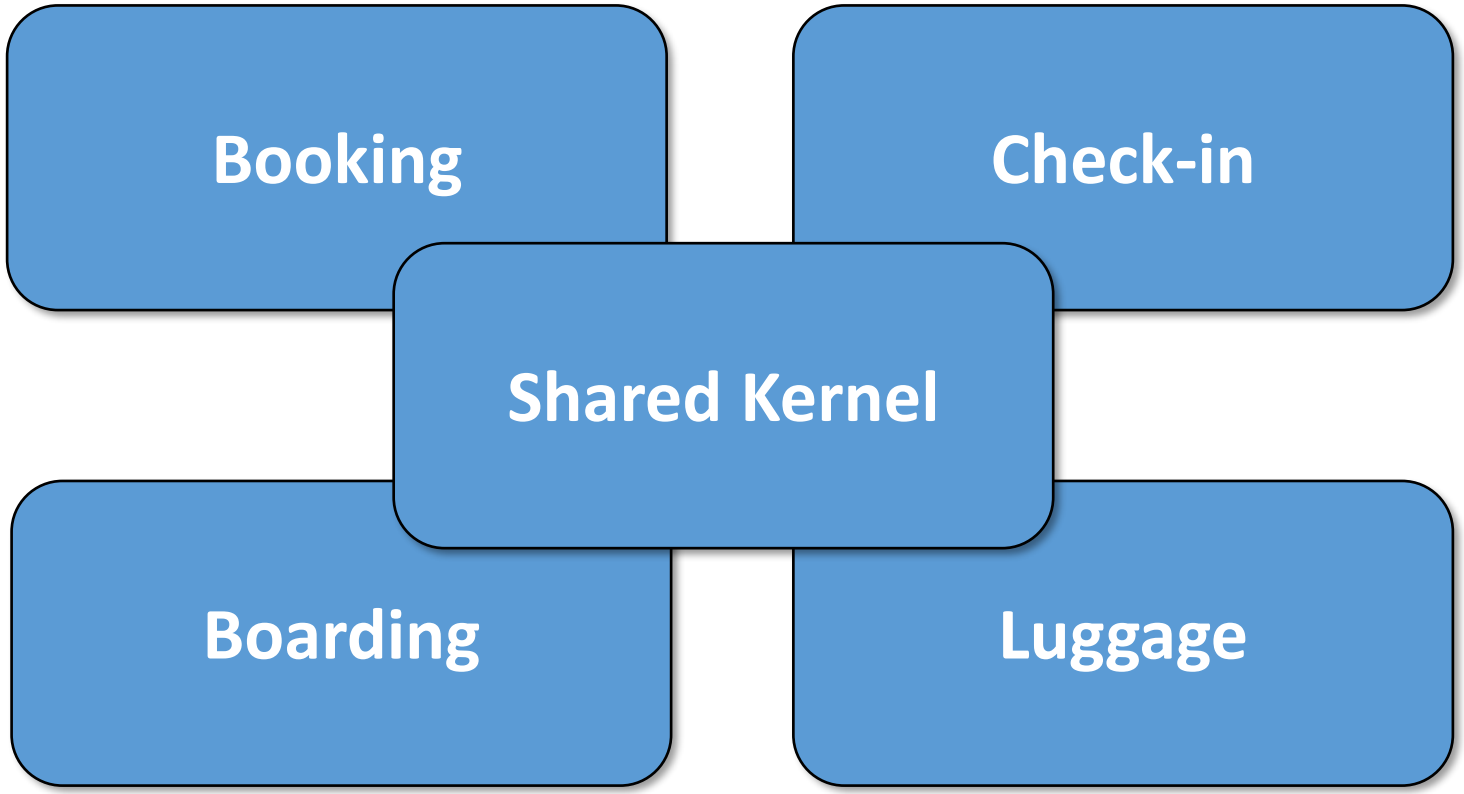
Context Map

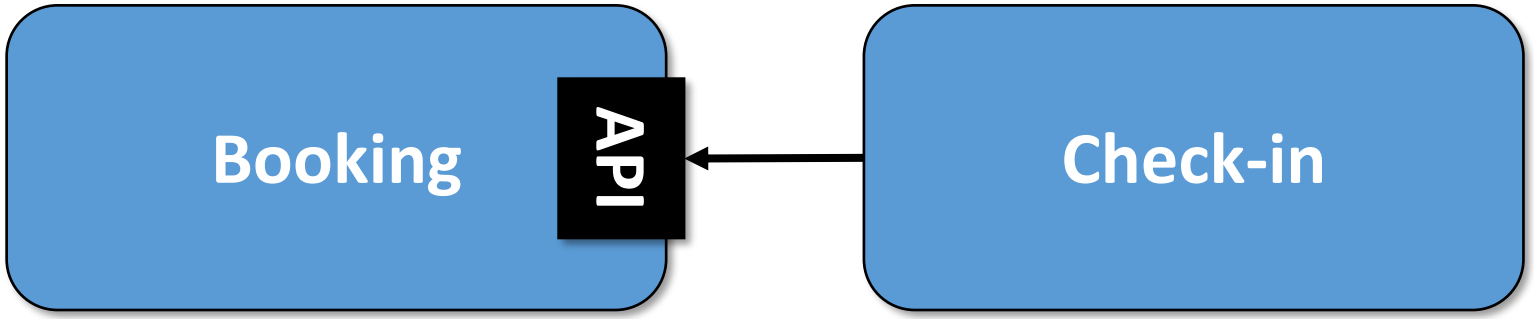


Context Map

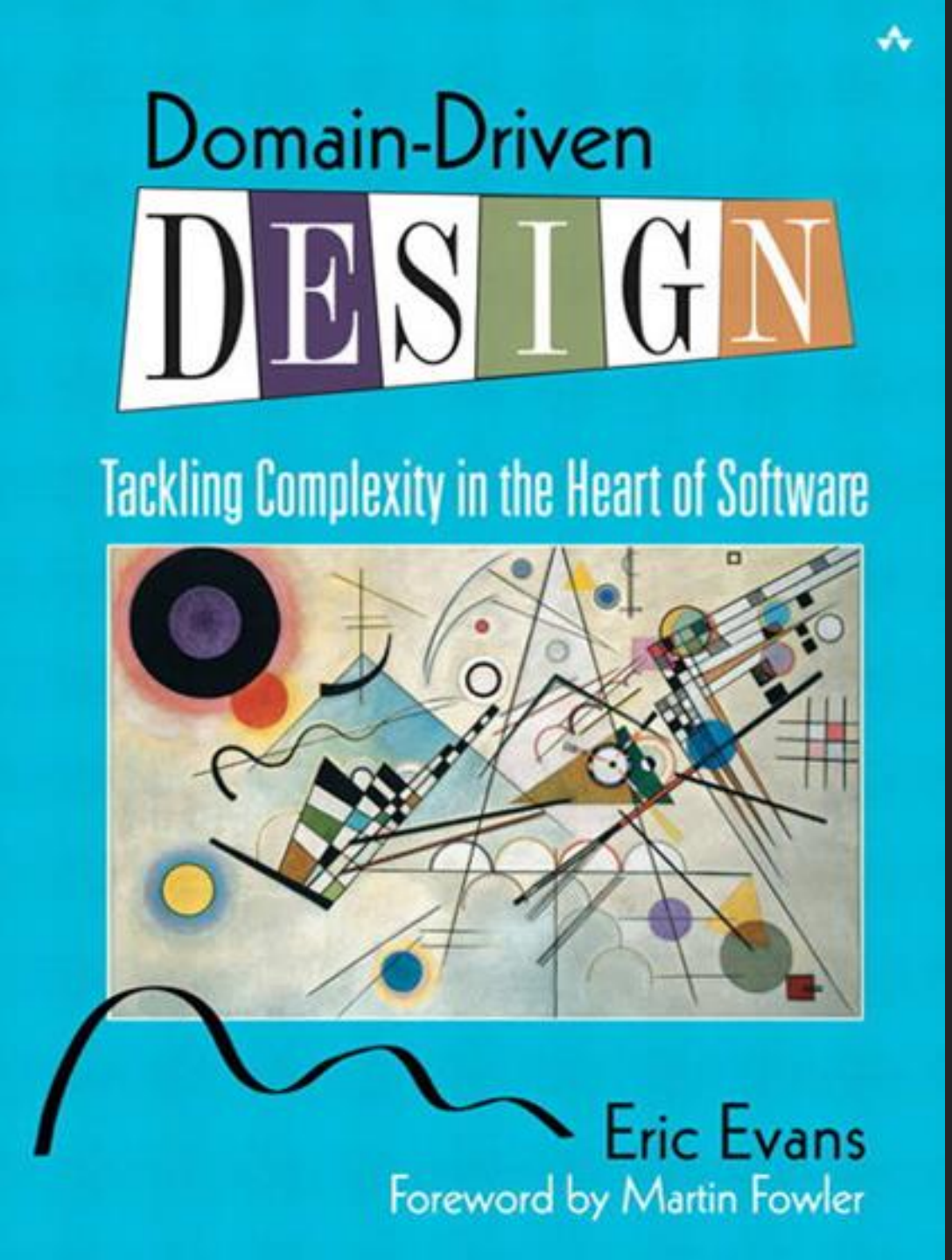
Responsibilities?

Breaking Changes?

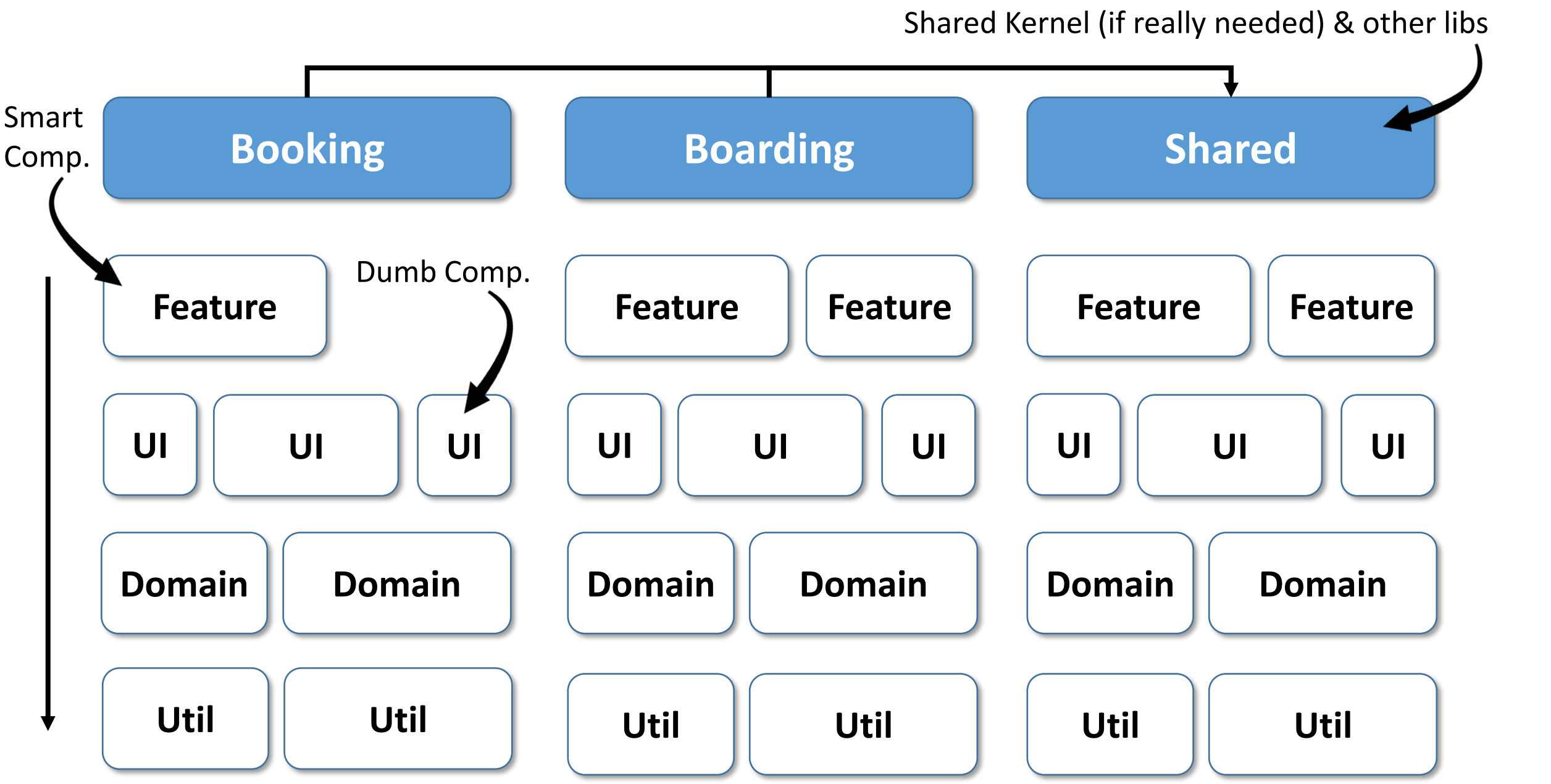


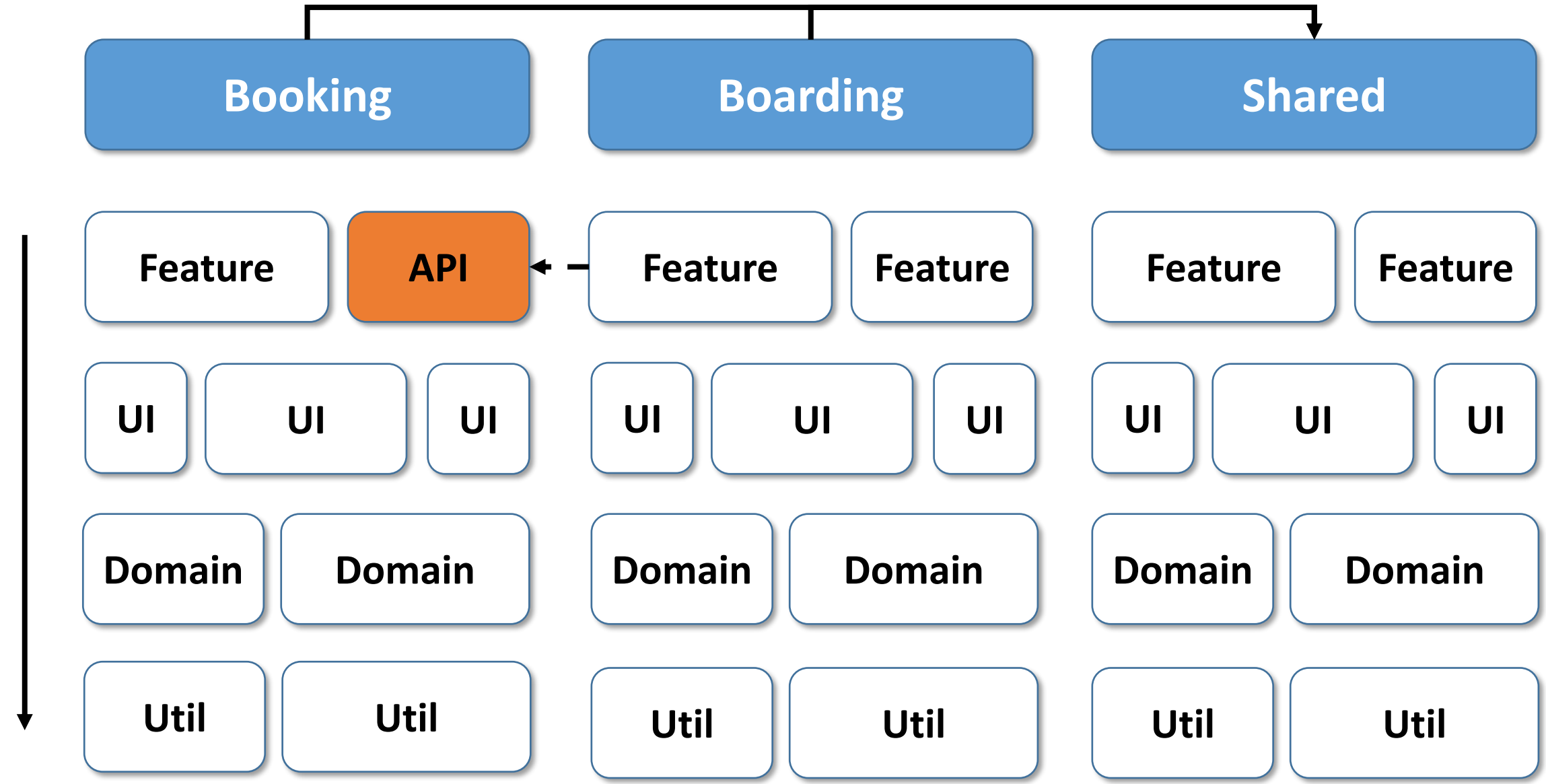


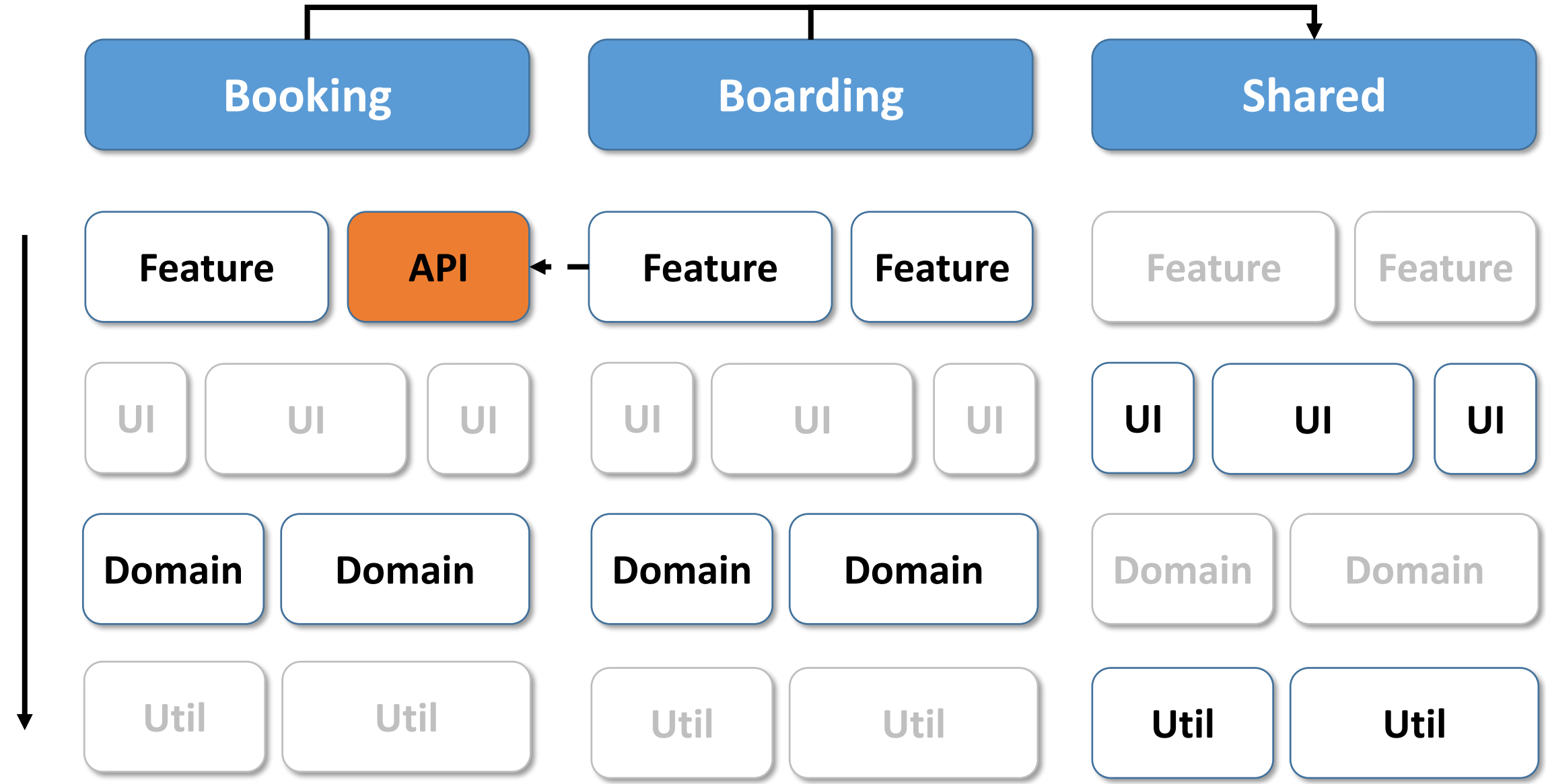
Open-/Host-Service



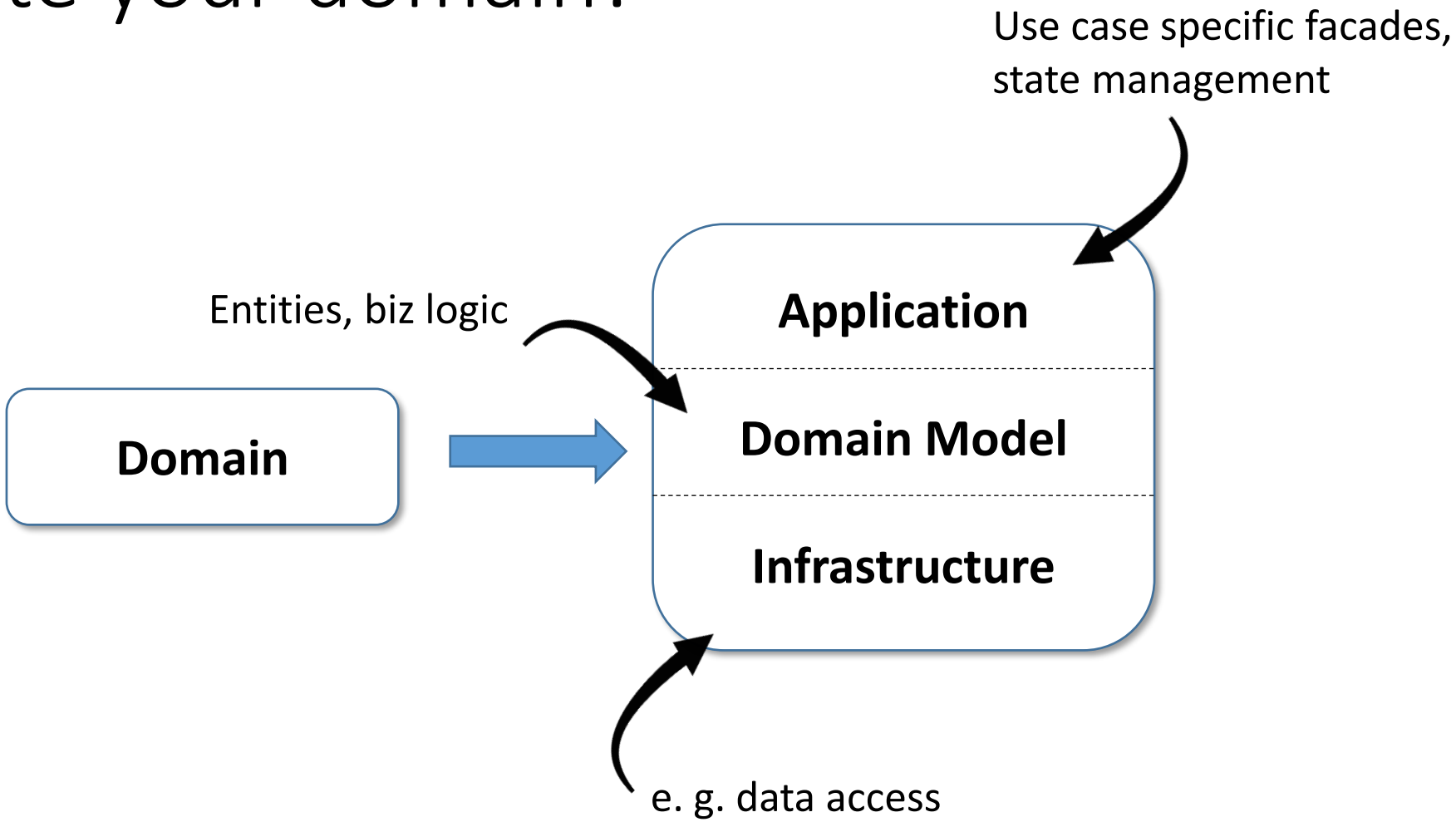
Lots of approaches for cross-domain communication and more …


Shared Kernel (if really needed) & other libs
Smart Comp.
Booking
Boarding
Shared
Dumb Comp.
Feature
Feature
Feature
Feature
Feature
UI
UI
UI
UI
UI
UI
UI
UI
UI
Domain
Domain
Domain
Domain
Domain
Domain
Util
Util
Util
Util
Util
Util


Booking
Boarding
Shared
Feature
API
Feature
Feature
Feature
Feature
UI
UI
UI
UI
UI
UI
UI
UI
UI
Domain
Domain
Domain
Domain
Domain
Domain
Util
Util
Util
Util
Util
Util


Booking
Boarding
Shared
Feature
API
Feature
Feature
Feature
Feature
UI
UI
UI
UI
UI
UI
UI
UI
UI
Domain
Domain
Domain
Domain
Domain
Domain
Util
Util
Util
Util
Util
Util

Isolate your domain!



Alternatives to layering

- e. g. Hexagonal Architecture, Clean Architecture
- Anyway: We need to **restrict access** b/w libraries

DEMO

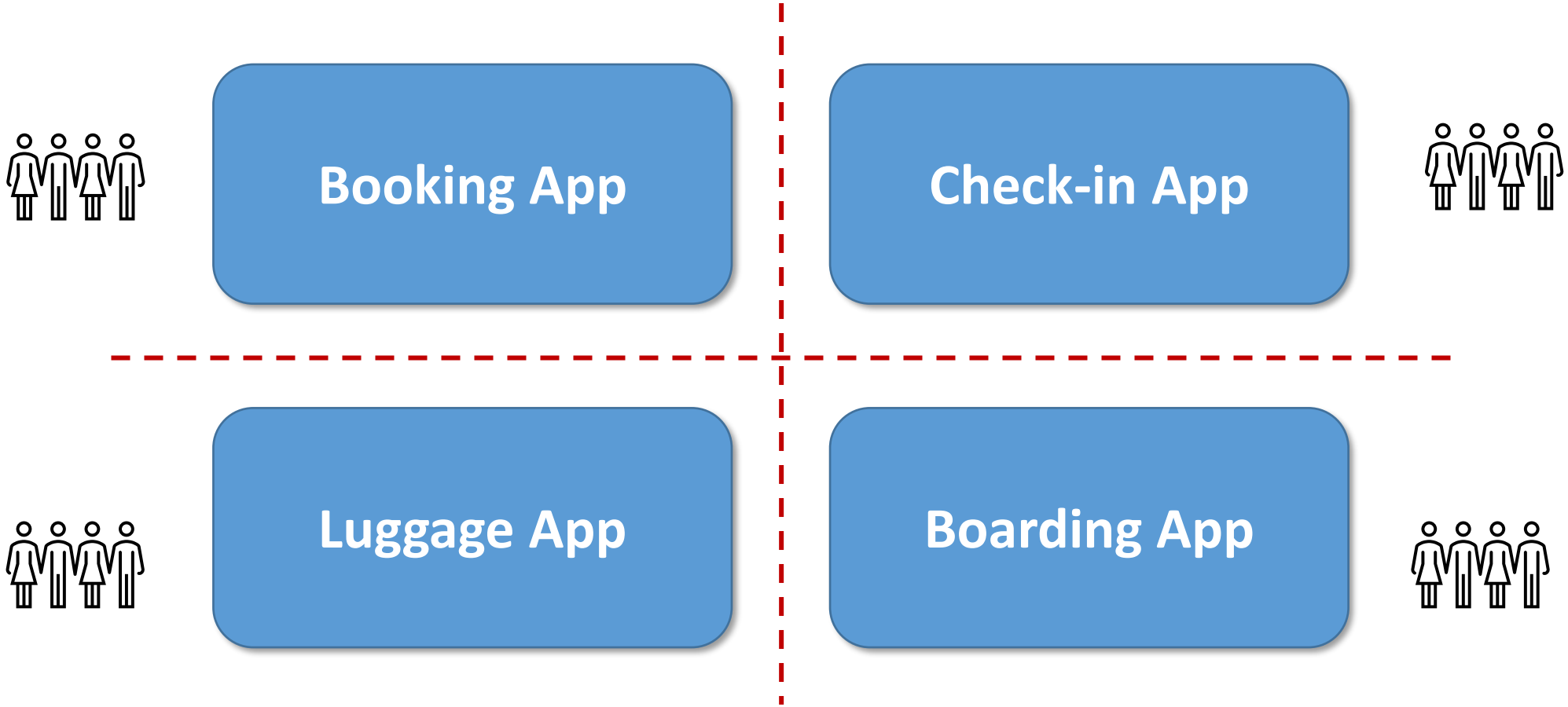
Finegrained Libraries

- Unit of recompilation
- Unit of retesting
- Access restrictions
- Information Hiding
- Easy: Just *ng g lib …*
- Future replacement for NgModules?

Micro Frontends?

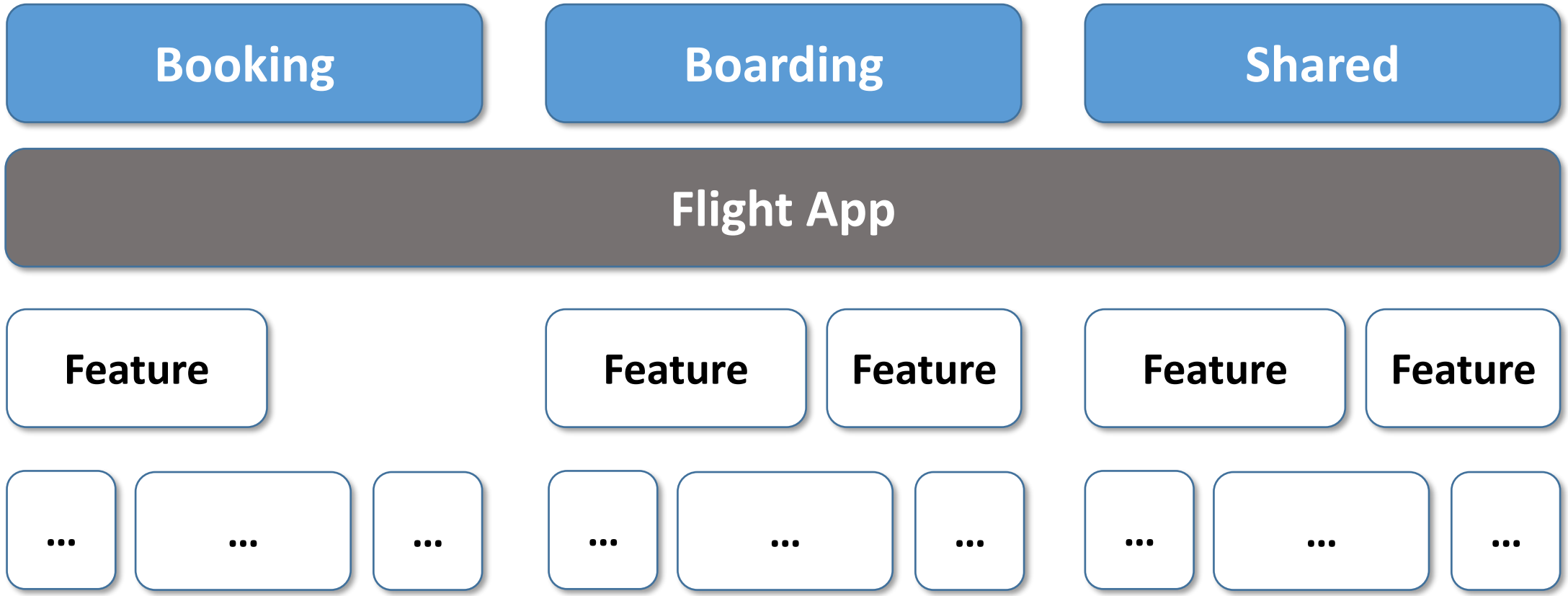
Short outlook

Microfrontends

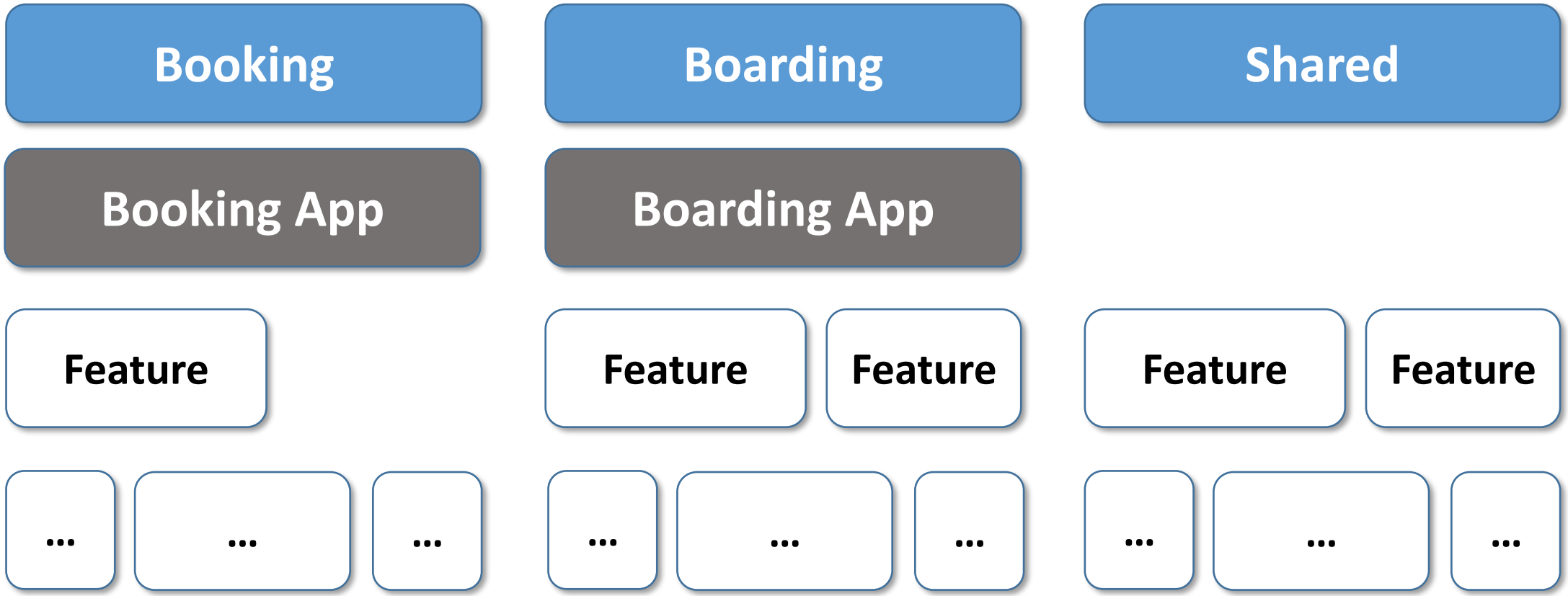


Microfrontends
are first and foremost
about **scaling teams!**

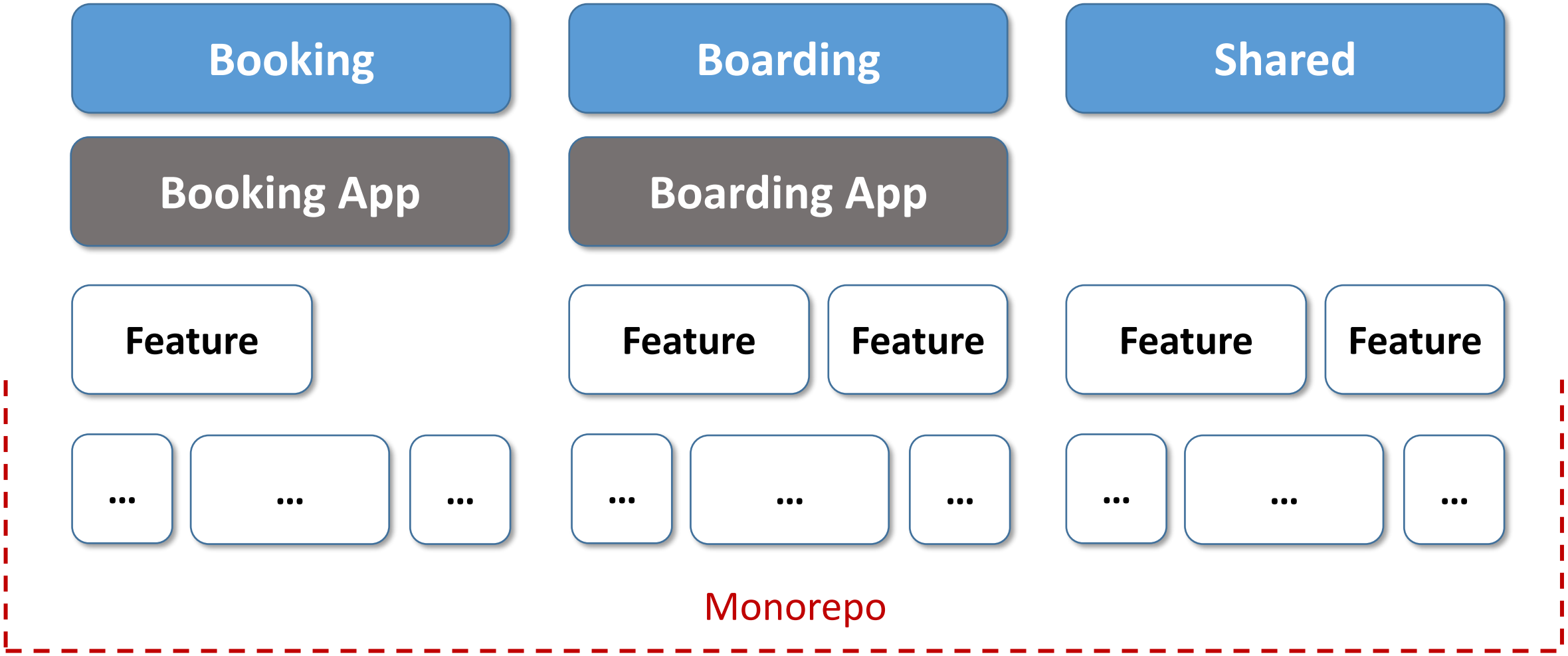
Deployment Monolith



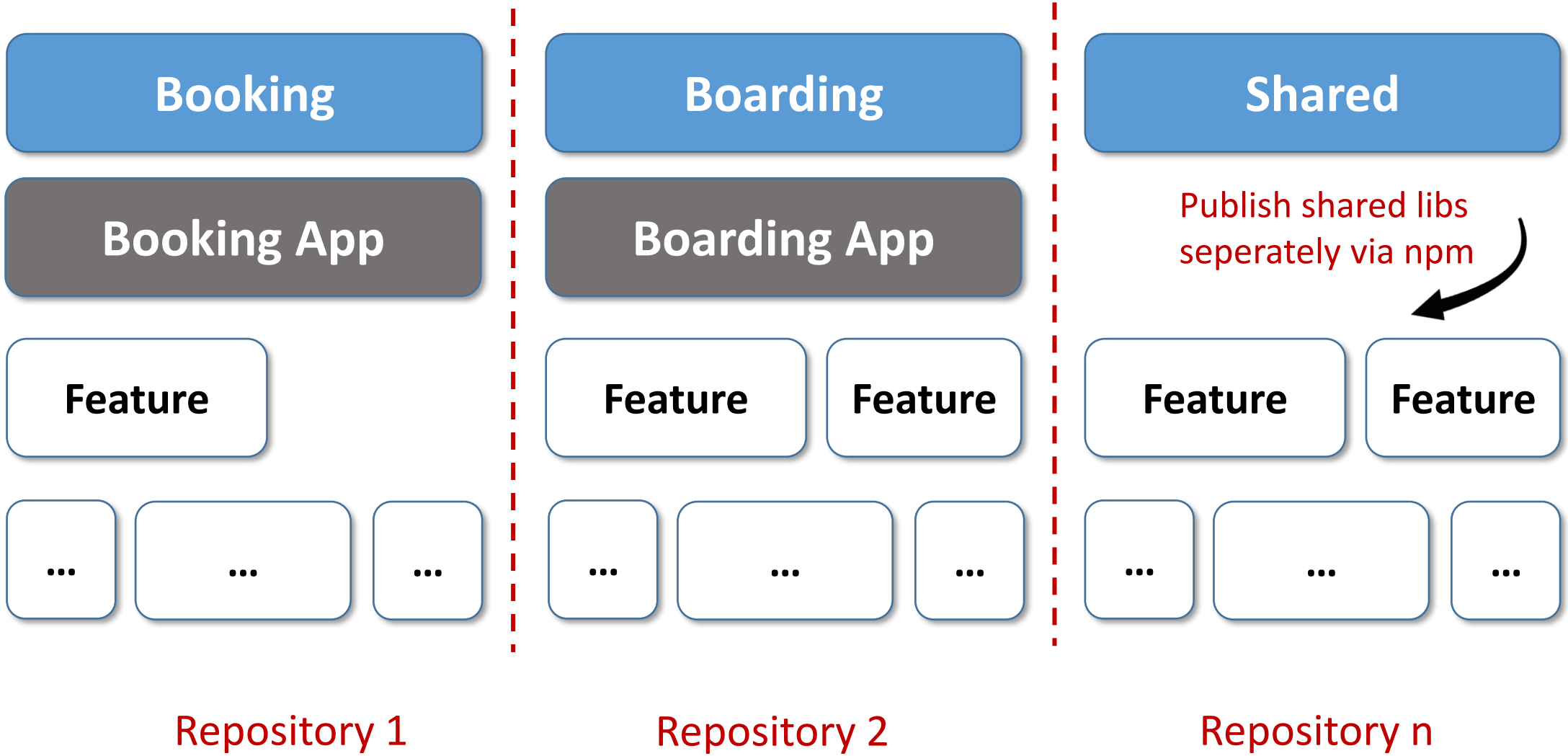
Microfrontends



Option 1: One App per Domain



Option 2: One Monorepo per Domain

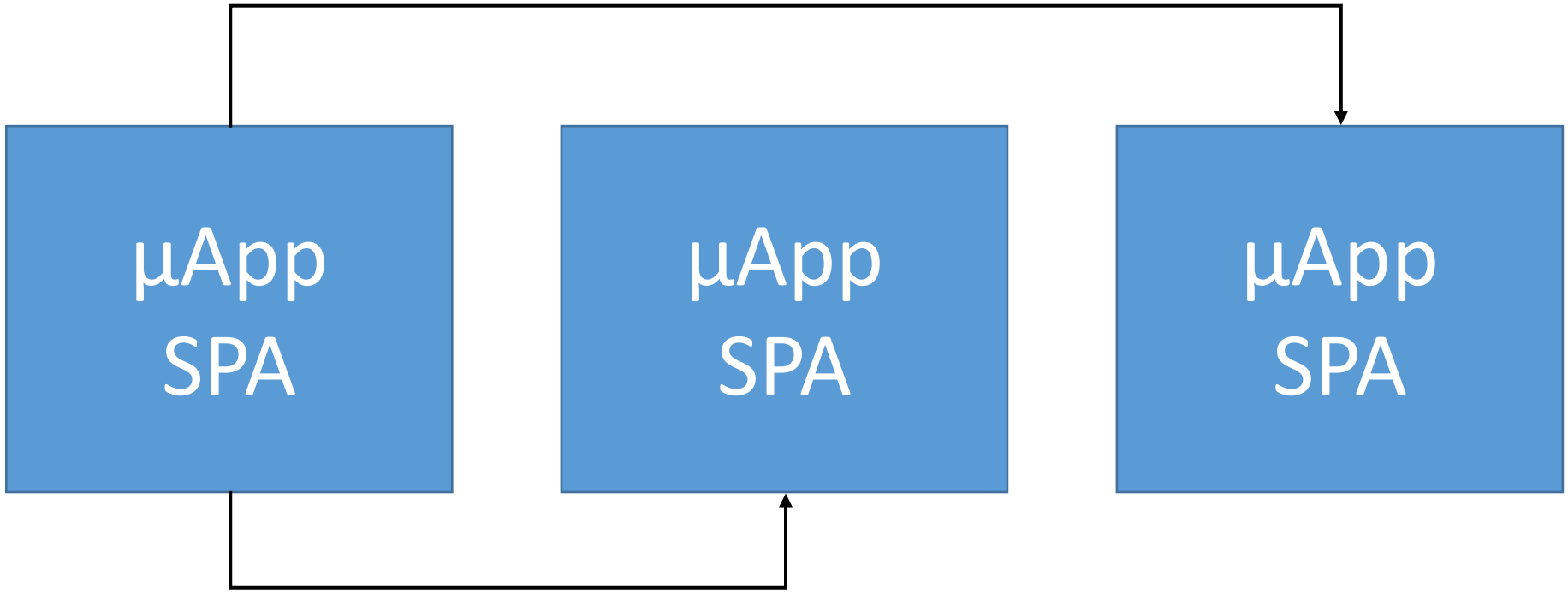


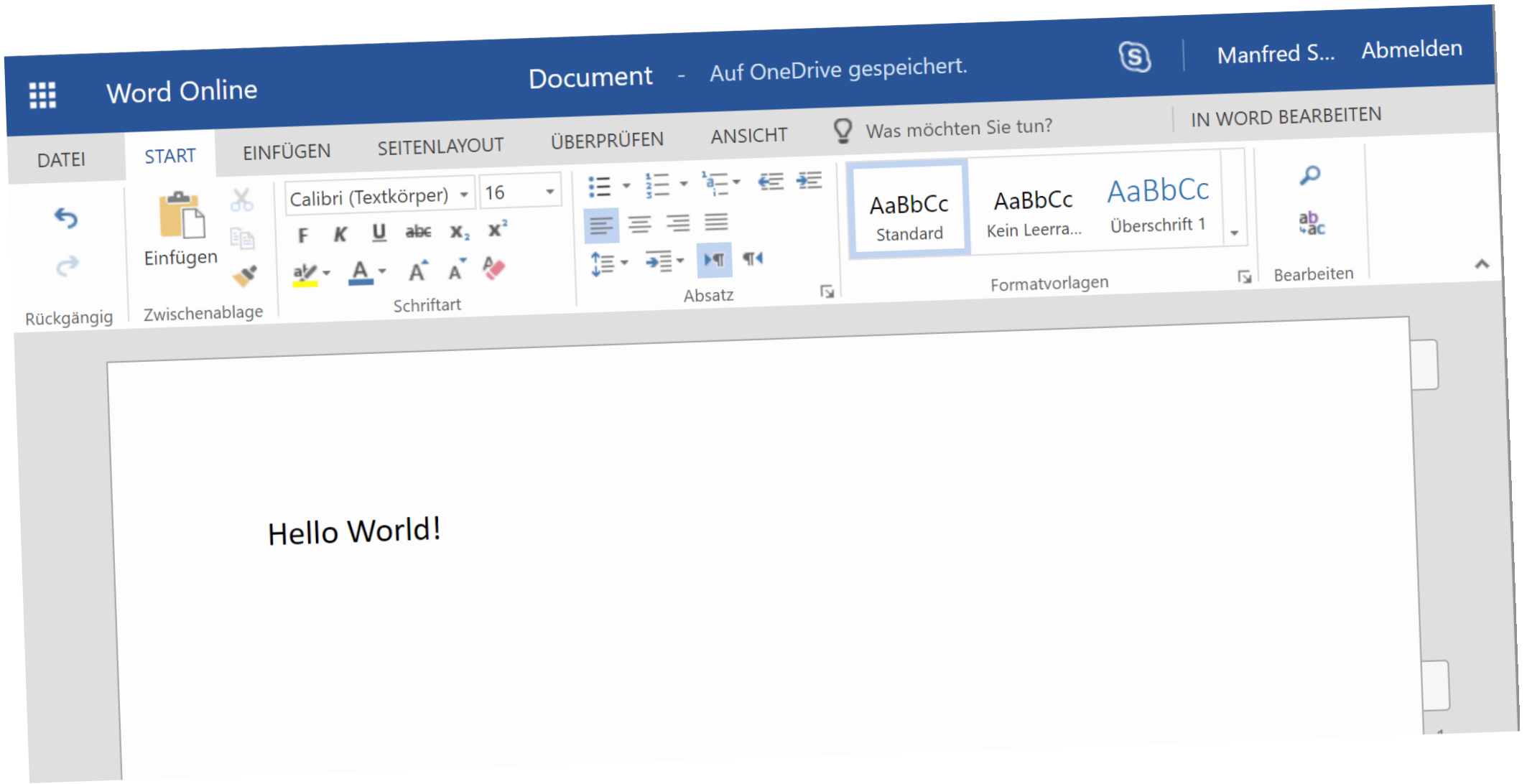
Benefits

- **Autonomous Teams**
- Separate Development
- Separate Deployment
- Own architecture decisions
- Own technology descisions

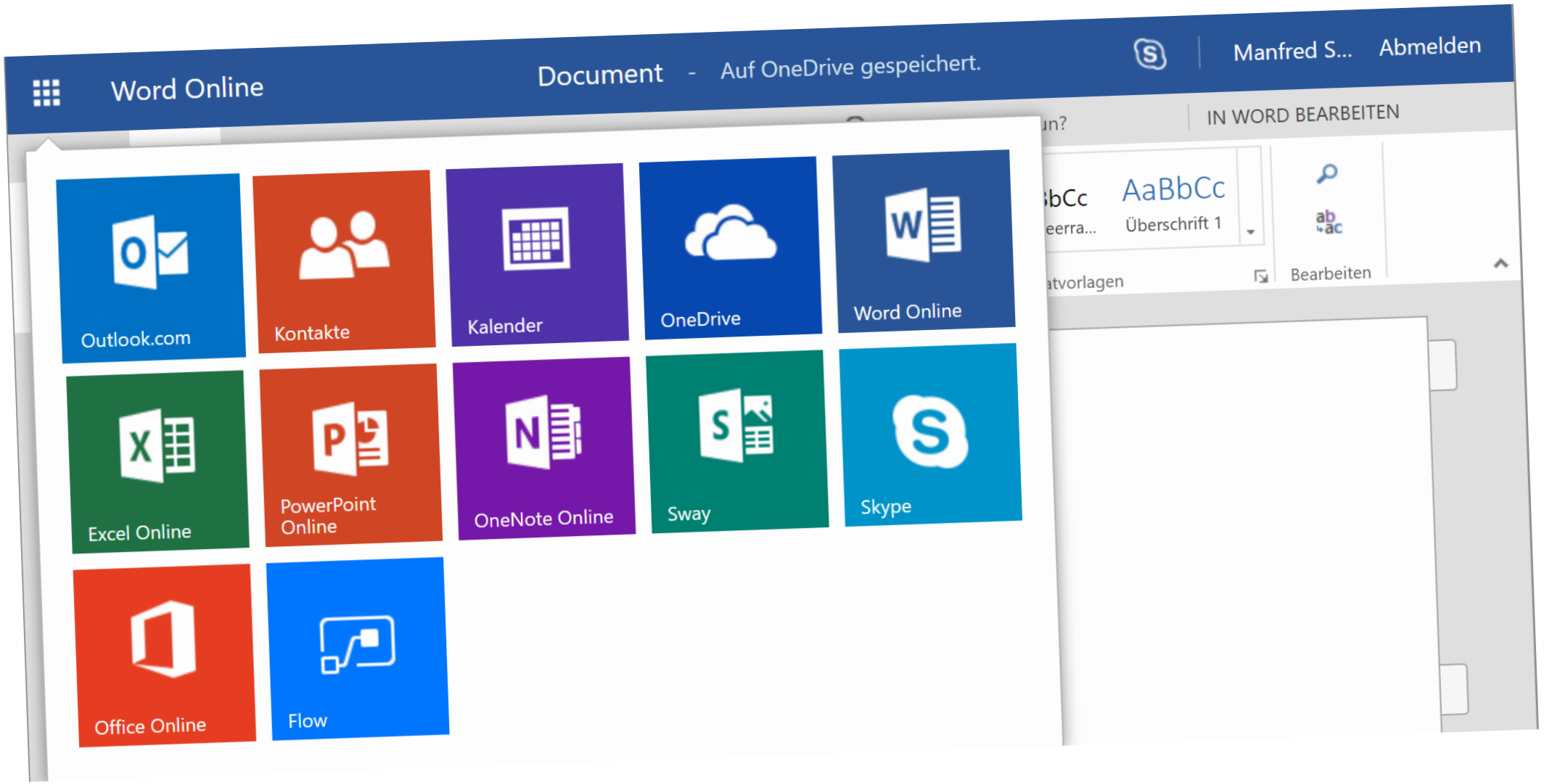
Integration via Hyperlinks

UI Composition
w/ Hyperlinks





@ManfredSteyer

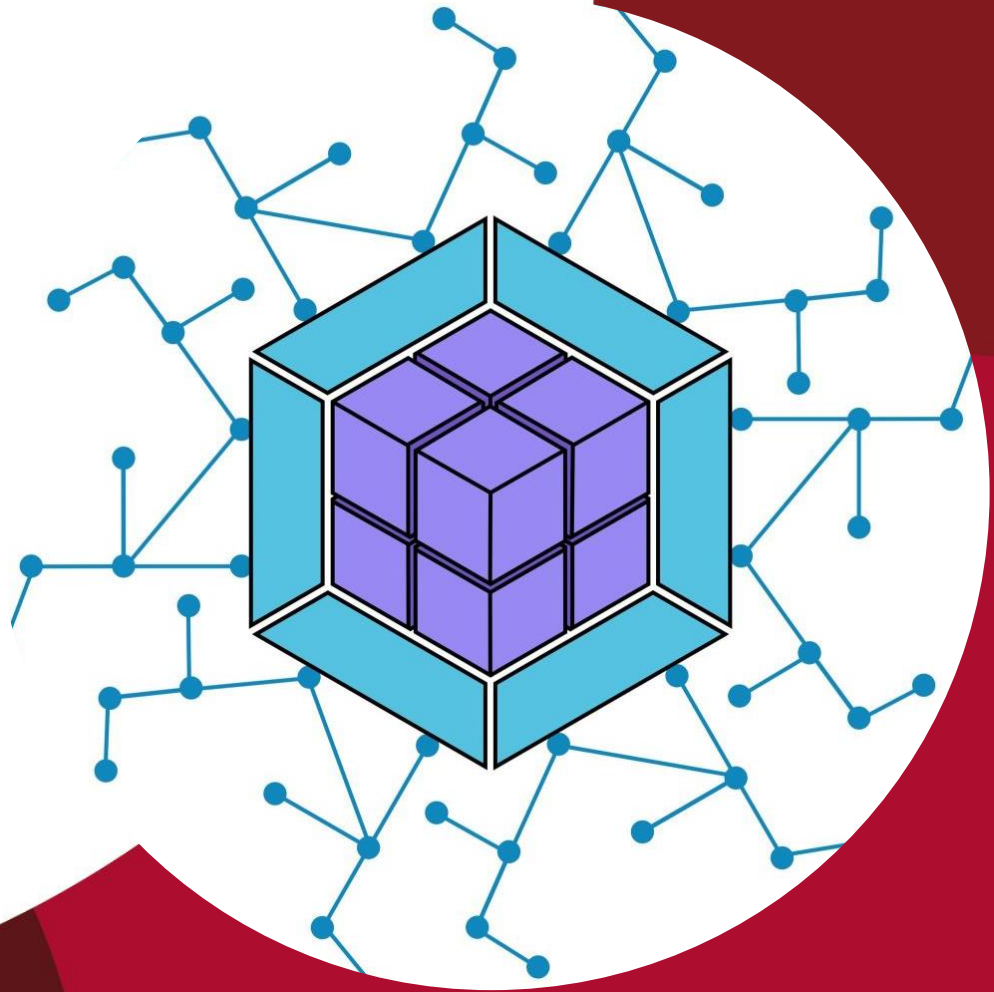


@ManfredSteyer

Integration via Shell

Providing a (SPA based) Shell

Webpack 5
Module Federation



ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

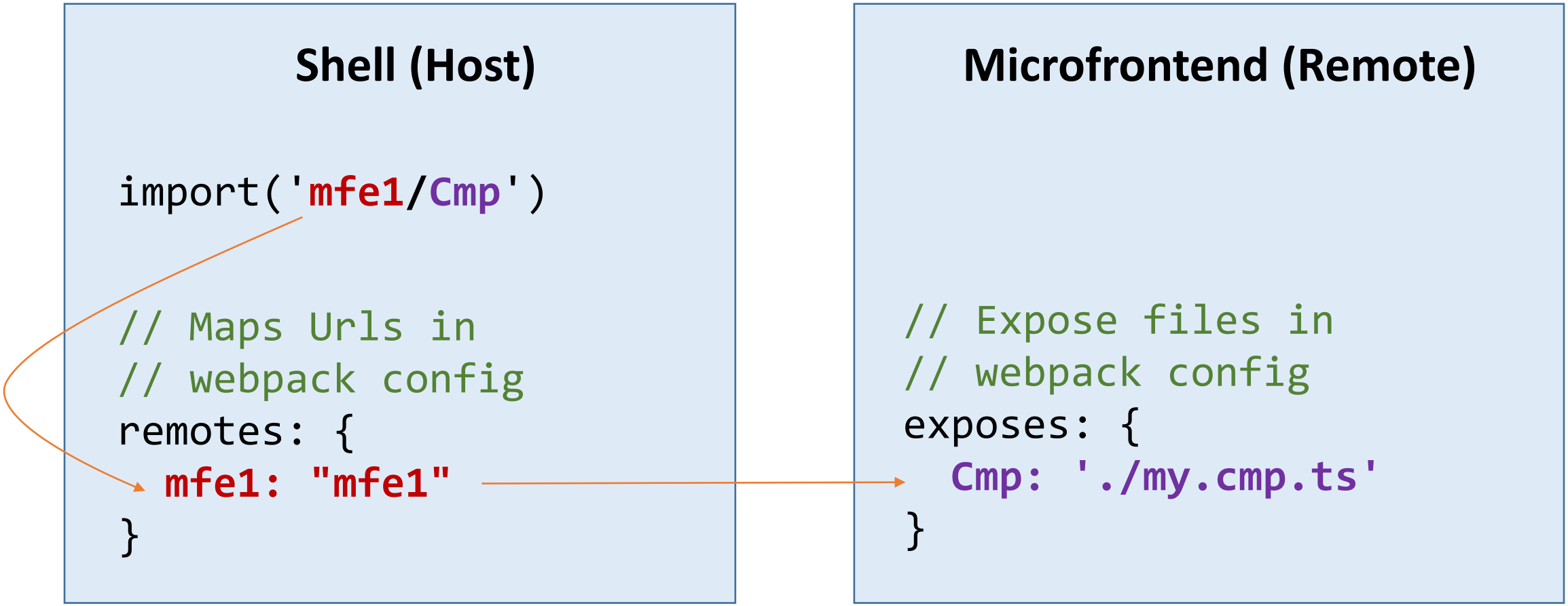
Idea

Does not work with
webpack/ Angular CLI

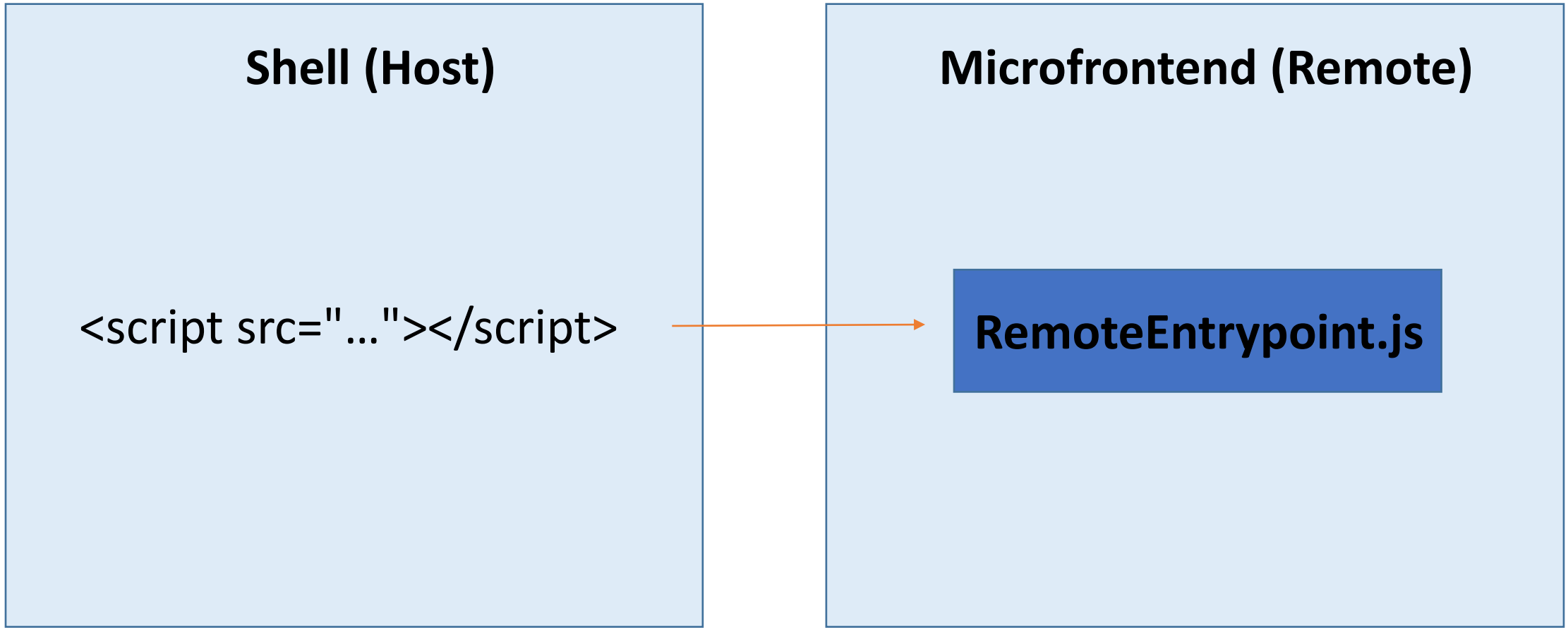
```
const Component = import('http://other-app/xyz')
```

Even lazy parts must be
known at compile time!

Webpack 5 Module Federation



How to Get the Microfrontend's URL?



How to Share Libs?

Shell (Host)

```
shared: [
  "@angular/core", "..."
]
```

Microfrontend (Remote)

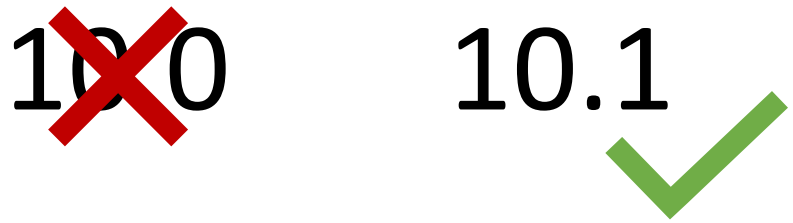
```
shared: [
  "@angular/core", "..."
]
```

Dealing with Version Mismatches

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

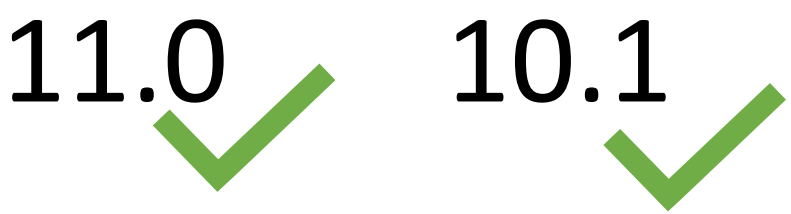
Default Behavior

Selecting the highest compatible version



Default Behavior

Conflict: No highest compatible version



Example

- Shell: my-lib: ^10.0
- MFE1: my-lib: ^10.1
- MFE2: my-lib: ^9.0
- MFE3: my-lib: ^9.1

Result:

- Shell and MFE1 share ^10.1
- MFE2 and MFE3 share ^9.1

Configuring Singletons

```
shared: {
  "my-lib": {
    singleton: true
  }
}
```

11.0 ✓ 10.1 ✗

Configuring Singletons

```
shared: {
  "my-lib": {
    singleton: true,
    strictVersion: true // Error instead of warning!
  }
}
```

11.0 ✓ 1~~0~~.1 ✗

Relaxing Version Requirements

```
shared: {
  "my-lib": {
     requiredVersion: ">=1.0.1 <11.1.1"
  }
}
```

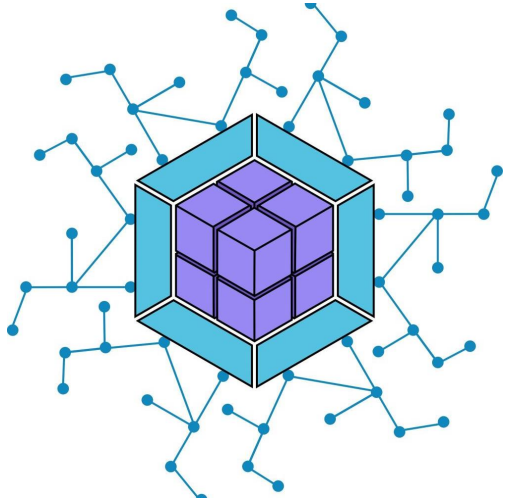
Federated Angular:
Angular, CLI, & Module Federation

ANGULAR ARCHITECTS
INSIDE KNOWLEDGE

webpack

Angular CLI

?



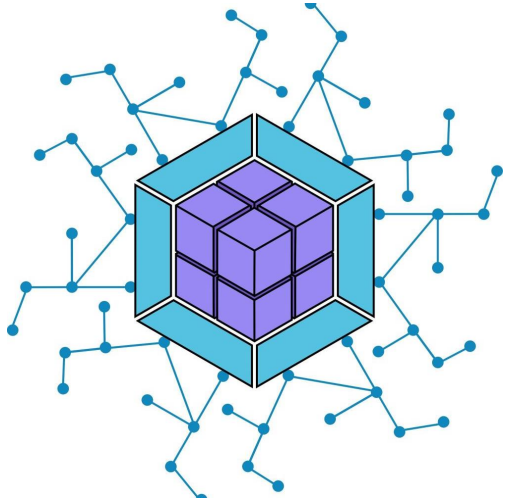
Module Federation
Configuration

@ManfredSteyer

Custom Builder

Angular CLI

webpack



Module Federation
Configuration

@angular-architects/module-federation

1.0.2 • Public • Published 18 hours ago

Readme | Explore BETA | 3 Dependencies

Features 

✅ Generates the skeleton for a Module Federation config.

✅ Installs a custom builder to enable Module Federation.

✅ Assigning a new port to serve (`ng serve`) several projects at once.

Usage

1) ng add @angular-architects/module-federation
2) Adjust generated configuration
3) ng serve

DEMO

ANGULAR
ARCHITECTS
INSIDE KNOWLEDGE

Possible Roadmap

Well …

- Webpack 5: final
- @angular-architects/module-federation: final
- CLI 11: Experimental webpack 5 support
- CLI 12: Official webpack 5 support (May 2021)


Choosing a Solution

Some General Advice