# Bounded Contexts und Event-Contracts

MEA, Workshop 2, 22.10.2026

# Der Tag

| | |
|---|---|
| 10:00 | Eine Karte pro Person |
| 10:05 | Rückblick mit dem Lern-Tutor |
| 10:45 | Der Weg zum Code (und was wir dabei abkürzen)<br>Modulgrenzen finden · Werkzeuge für Subdomains · Core Domain Chart · Bounded Context · Teams |
| 12:30 | Mittagspause |
| 13:15 | Context Map am Whiteboard |
| 14:30 | Repository-Struktur |
| 15:15 | Ausblick auf Implementierung mit Daniel |
| 16:15 | Ausblick und Hausaufgabe |

# Der Weg zum Code

15.10.

### EventStorming

*Was passiert in diesem Restaurant?*

Stickies auf einer Zeitleiste

HEUTE VORMITTAG

### Bounded Contexts

*Wer ist wofür verantwortlich?*

Core Domain Chart, vier Services, Context Map

HEUTE NACHMITTAG

### Context Map und Repositories

*Welche Events fließen wohin, und wo liegt was?*

Context Map am Whiteboard, Repository-Struktur

29.10.

### Einstieg in die Implementierung

*Was teilt mein Service der Welt mit?*

Contracts, Event-Katalog, Umfang des Walking Skeletons, Clients

05.11. BIS 26.11.

### Walking Skeleton

*Läuft es wirklich?*

Jeder Service tauscht Events in der gemeinsamen Umgebung aus

informell, billig zu ändern

formal, teuer zu ändern

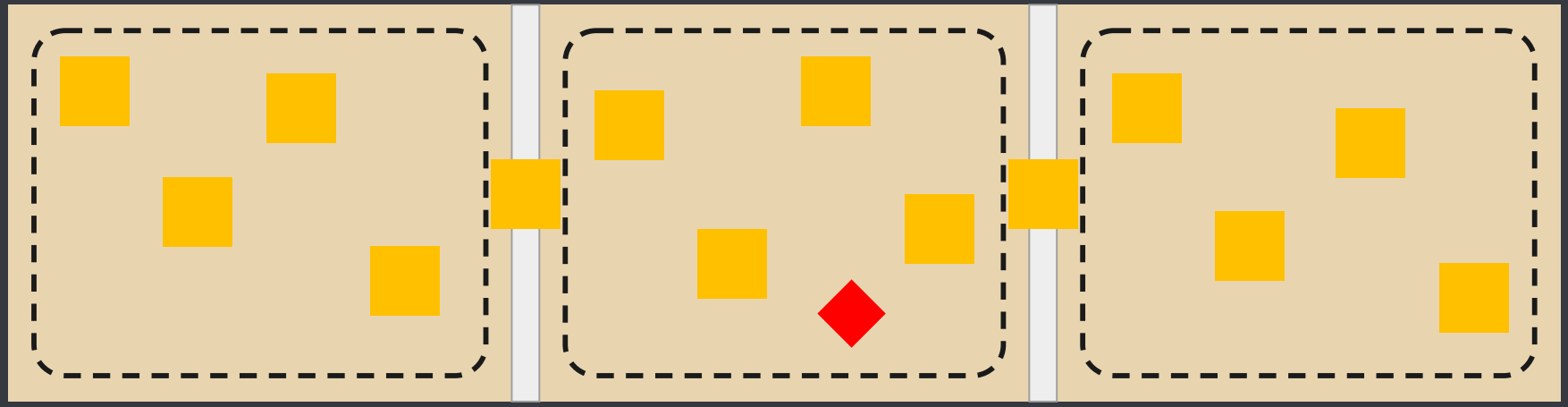
> Wir haben nur vier Workshops im Grundlagenteil. Deshalb gehen wir den Weg schneller als üblich und schreiben schon nächste Woche Code.

# Von der Domain Exploration zum Beginn der Implementation

Wichtigstes Ziel: eine tragfähige Modulstruktur. Die Module ergeben sich aus den (Sub)domains.



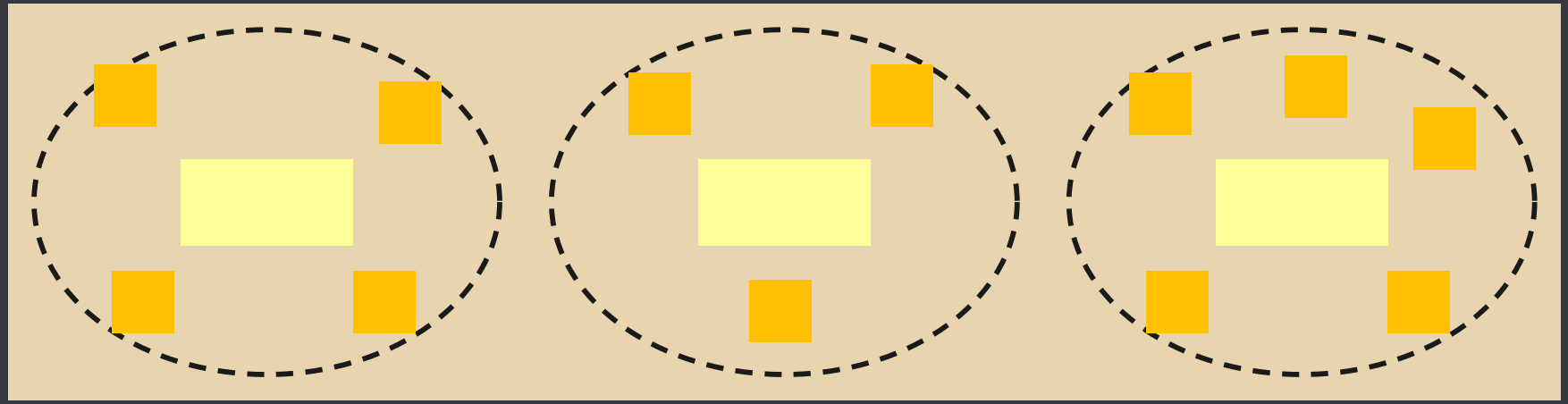
# Subdomains finden: Cluster zwischen Pivotal Events




> Die Fläche zwischen zwei Pivotal Events ist ein Kandidat für eine Subdomain.

Bei uns: Gemacht am 15.10., aber unsere Pivotal Events sind noch "unscharf". Das wird uns später noch Zeit kosten :-(

# Subdomains finden: Aggregates



Aggregate A

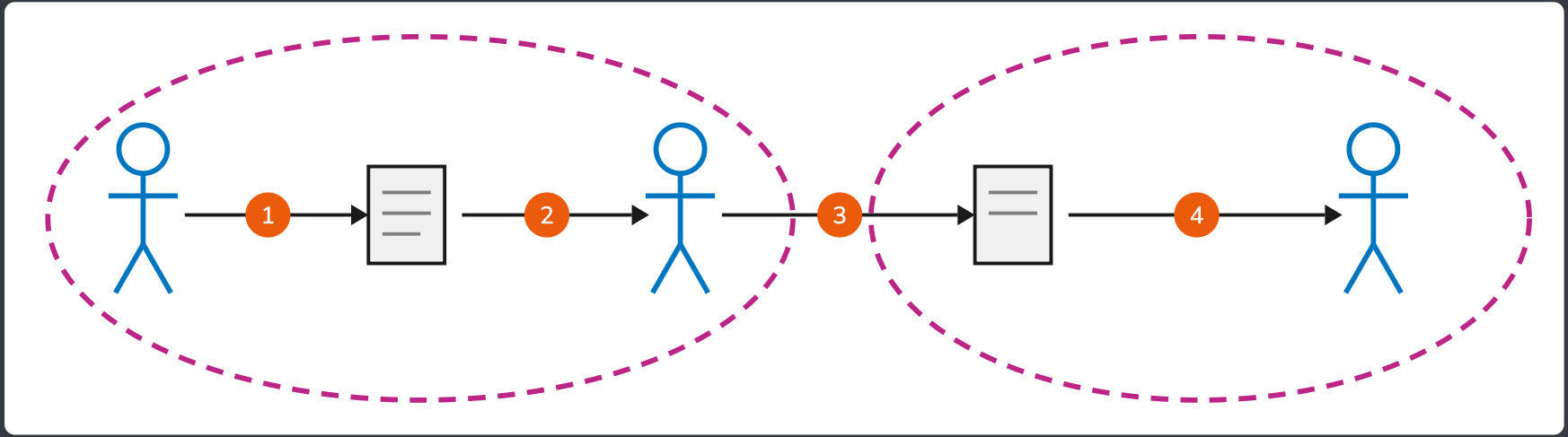
Aggregate B

Aggregate C

> Die Aggregate-Stickies aus dem EventStorming sind die wichtigen Business-Objekte. Events, die dasselbe Business-Objekt verändern, gehören zusammen.

Bei uns: Übersprungen, denn die Aggregates der Restaurant-Domäne kennen Sie aus ST2.

# Subdomains finden: Domain Storytelling

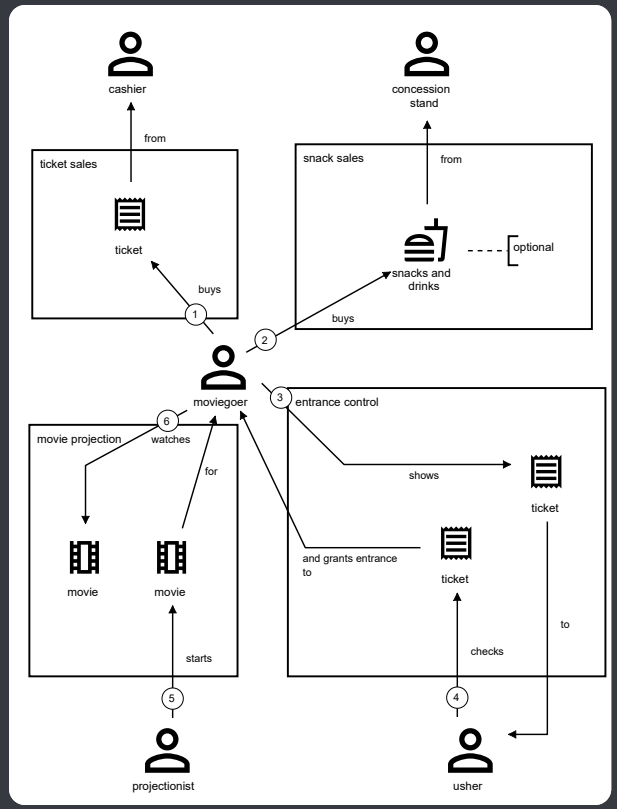



> Eine Domain Story erzählt einen Ablauf als Bildergeschichte: wer was mit welchem Arbeitsgegenstand tut, in welcher Reihenfolge. Im Gesamtbild aller Abläufe kreist man die Bereiche ein, die eng zusammenhängen.

Domain Storytelling: Stefan Hofer, Henning Schwentner, domainstorytelling.org.

Bei uns: Aus Zeitgründen ganz weggelassen.

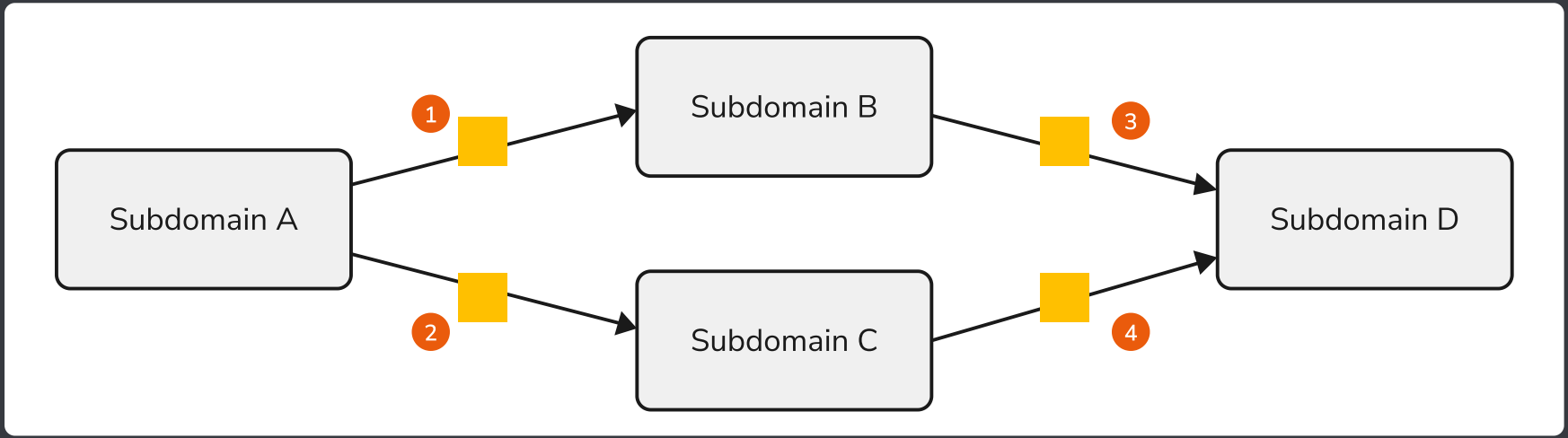
# Beispiel: ein Kinobesuch als Domain Story




- Akteure (Kinobesucher, Kasse, Snacktheke, Einlass, Vorführer) und Arbeitsgegenstände (Ticket, Snacks, Film)
- Nummerierte Sätze: Wer tut was womit, in welcher Reihenfolge?
- Die Kästen sind die eingekreisten Bereiche: Ticketverkauf, Snackverkauf, Einlass, Filmvorführung. Jeder ist ein Kandidat für eine Subdomain.

Domain Story von domainstorytelling.org, CC BY 4.0. Im PDF als statisches Bild ohne die Animation des Originals.

# Subdomains verbinden: Domain Message Flow

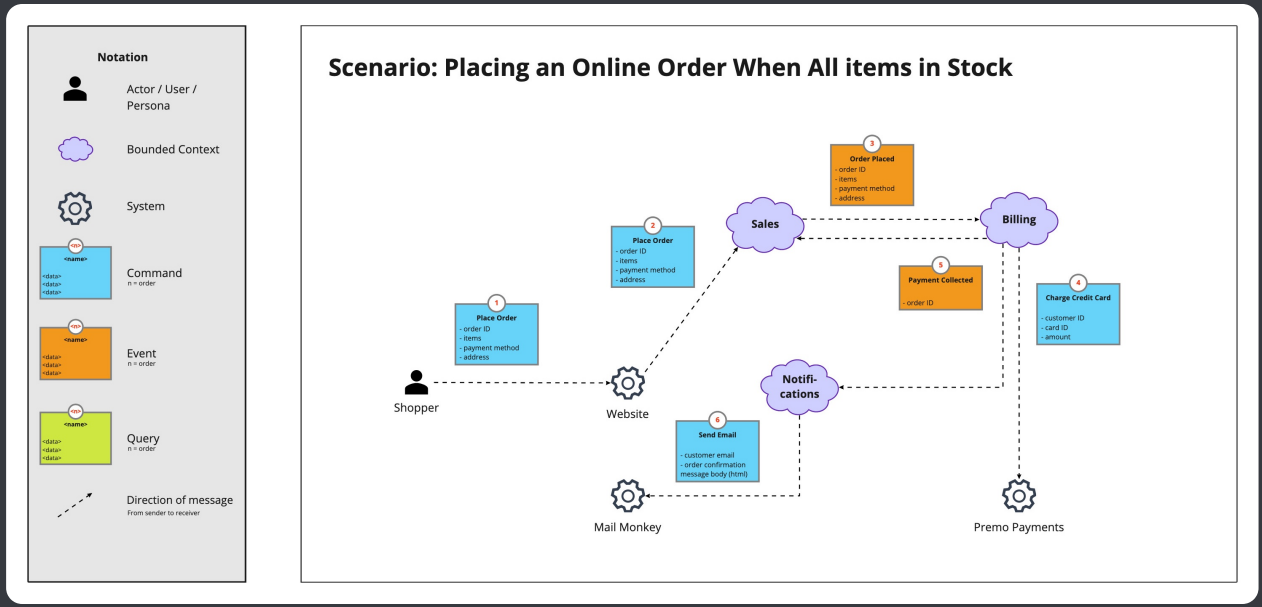



> Für einen Ablauf feingranular modellieren, welche Events in welcher Reihenfolge zwischen den Subdomains fließen. Von hier ist es nur ein kleiner Schritt zu den Event-Contracts.

Domain Message Flow Modelling: DDD Crew, github.com/ddd-crew/domain-message-flow-modelling.

Bei uns: Weggelassen, weil dieses Modul den Fokus auf die Umsetzung legt. Die Context Map heute Nachmittag ist eine grobe Variante davon.

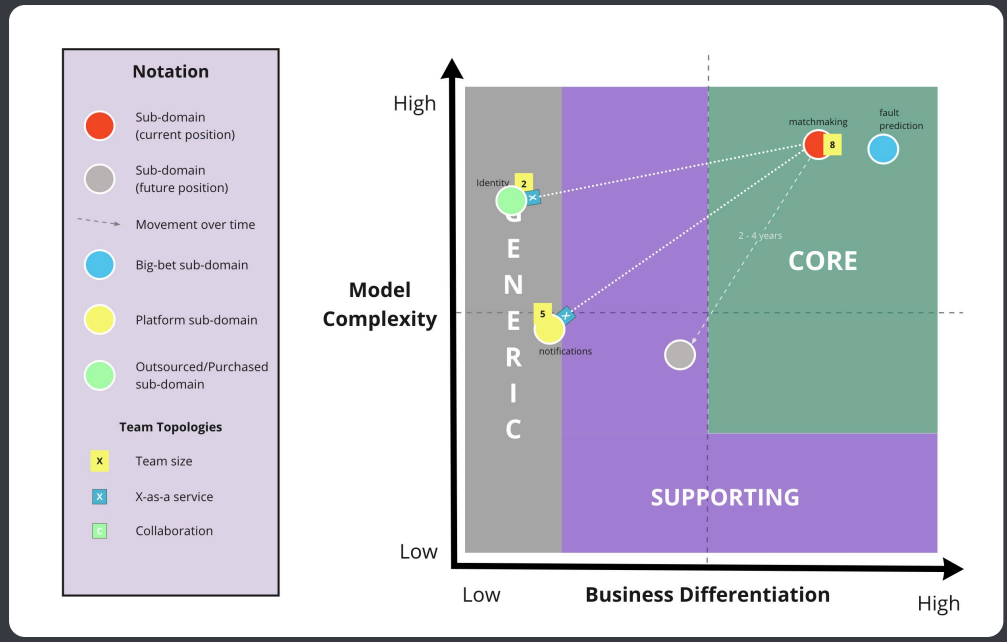
# Beispiel: Domain Message Flow einer Bestellung




Bounded Contexts als Wolken, dazwischen nummeriert die Nachrichten mit ihrem Inhalt: Commands (blau), Events (orange), Queries (grün).

DDD Crew, github.com/ddd-crew/domain-message-flow-modelling, CC BY-SA 4.0.

# Core Domain Chart




- Waagerecht: Gibt's das so nur bei uns oder an ganz vielen anderen Stellen auch?
- Senkrecht: Wie komplex ist die Logik dahinter?

- Core Domain
  - Herzstück meines Geschäftsmodells
  - damit gewinne oder verliere ich Kunden
  - baut man selbst und mit den meisten Leuten
- Supporting Domain
  - muss robust funktionieren
- Generic
  - kauft man
  - … oder man hält es so klein wie möglich

Core domain chart: DDD Crew, github.com/ddd-crew/core-domain-charts, CC BY-SA 4.0.

# Am Whiteboard: eine Punktwolke pro Subdomäne

| | |
|---|---|
| 5 Min. | Subdomänen benennen, je eine Farbe / ein Symbol (Legende am Rand) |
| 15 Min. | jede:r: ein Punkt pro Subdomäne ins Chart |
| 15 Min. | große Streuung → darüber reden |
| 10 Min. | Ergebnis: groß und wichtig, oder klein? |

# Bounded Context = Grenze eines Teams

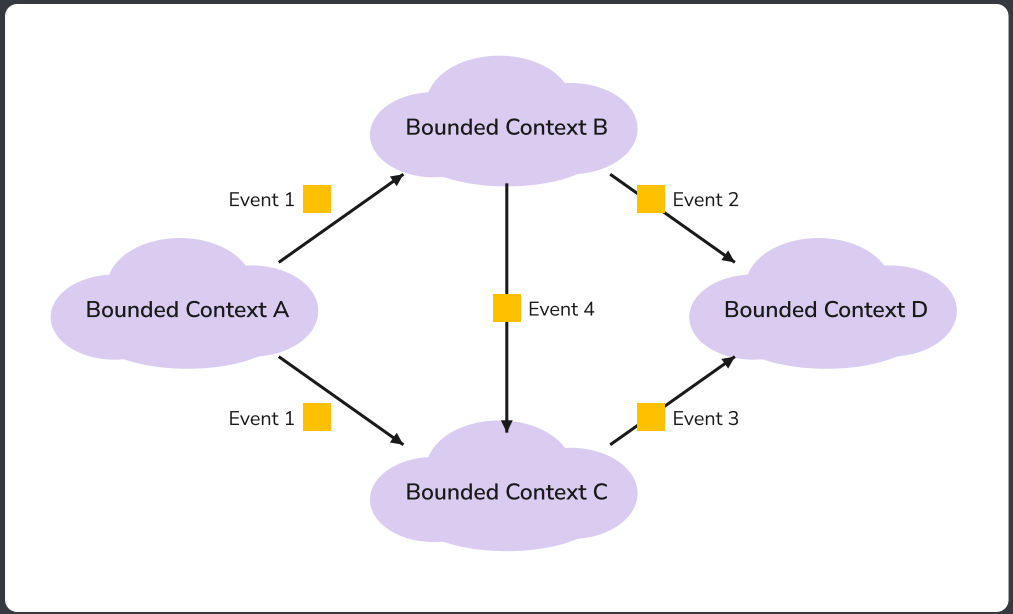
- Bereich, in dem ein Modell eindeutig gilt
- an der Grenze: andere Sprache, andere Verantwortung
- Beispiel Online-Handel: "Produkt" im Katalog ≠ "Produkt" im Versand
- Ideal: Bounded Context = 1 Subdomain = 1 Aggregate
  - Kann aber auch mehrere Subdomains und Aggregates umfassen

> 1 Bounded Context = 1 Service = 1 Team

- 4 Teams = 4 Services (2 × drei, 2 × zwei Personen)
- große Subdomänen → Dreierteams
- mehr als 4 Subdomänen → Nachbarn mit gleicher Sprache zusammenlegen

# Mittagspause

# Context Map




- *Bounded Context* = Service = Team
- Pfeil = Event über eine Grenze
  - sollte i.d.R. Pivotal Event sein
- Richtung: Publisher → Consumer
  - ein Event, mehrere Consumer → mehrere Pfeile
- interne Events: kein Pfeil

> Wo sind jetzt die synchronen Calls aus ST2?

# Wo liegt welche Dokumentation?

```
restaurant-system-ai-workspace/
├── CLAUDE.md              # Grundsätze für den Agenten, kurz
├── .claude/rules/         # ausführliche Regeln für den Agenten
├── tools/                 # repos.txt, workspace.py: alle Repos auf einmal klonen
├── docs/
│   ├── documentation/     # Hugo-Site: Context Map, Event-Katalog, Entscheidungen, Glossar
│   └── plans/             # temporäre Pläne des Agenten
├── services/<service>/    # Code, Doku, docs/asyncapi.yaml; ein Repo pro Team
└── devops/                # lokale Umgebung, Pipeline-Vorlagen
```

- Workspace = nur der Rahmen (CLAUDE.md, Regeln, Tools), kein Code
- Themenordner gitignored, darin eigenständige Repos
- ein Befehl klont alles: `python tools/workspace.py clone`
  - … und weiter hilfreiche Optionen
- Contract im Service-Repo!
  - Code + Contract im selben Merge Request
- Event-Katalog sammelt die Contracts beim Bauen ein
  - kein Team arbeitet im Repo eines anderen

# Input: AsyncAPI, und Live-Demo

- AsyncAPI: Channels, Operations, Messages, Payload
  - Details: eigenes Deck, Beispiel aus anderer Domäne
- Live-Demo: Claude Code liest das Board (Miro MCP) → erster Contract als docs/asyncapi.yaml

```
1. claude mcp add --transport http miro https://mcp.miro.com
2. Run /mcp, select "miro" and log in to Miro in the browser.
```

> Wo rät der Agent? → Dort ist das Board unklar.

# Teamarbeit: der erste Event-Contract

| | |
|---|---|
| 1. Board | eigenen Teil prüfen: jeder gelbe Sticky wirklich ein Event? |
| 2. Entwurf | Agent schreibt AsyncAPI aus Board + Context Map → docs/asyncapi.yaml im Service-Repo |
| 3. Prüfen | Fakt (ist passiert!) oder verstecktes Kommando? Nur notwendige Daten für potentielle Consumer (nicht mehr)? |
| 4. Metadaten | offen lassen, kommt später |
| 5. Push | npx @asyncapi/cli validate docs/asyncapi.yaml, dann push |

> Abweichungen vom Entwurf des Agenten notieren, mit Grund

# Ausblick auf Implementierung mit Daniel

- Helper-Libs aus dem Microservice Dungeon, auch fürs Restaurant
  - Event senden, Event empfangen
- lokales Dev Environment
  - mitmachen: erstes Event auf dem eigenen Rechner

# Rückblick auf den Tag

- was nehmen Sie mit?
- was ist noch unklar?
- was machen wir am 29.10. anders?

# Hausaufgabe bis 29.10.

- Contract-Entwurf mit Claude Code und dem Miro-Board: docs/asyncapi.yaml im Service-Repo, validiert
- Abweichungen vom Entwurf des Agenten, mit Grund
- Events an der Grenze mit den Nachbarteams abstimmen, offene Punkte notieren
- am 29.10. zeigt jedes Team seinen Contract am Bildschirm
- Kernaussagen des Workshops festhalten (Tutor oder eigene Wege) und ins persönliche Learnlog eintragen: nicht vorzuzeigen, aber Teil der Hausaufgabe

# Fragen?

archi-lab.io