Combining Orchestration and Choreography for a Clean Architecture

MIRAGON

Andreas Riepl & Thomas Heinrichs

CAMUNDA COMMUNITY SUMMIT 2023

About Us

MIRAGON

Thomas Heinrichs
BPM Consultant

Andreas Riepl
Full-Stack Developer

Augsburg, Germany

What to expect

1 Reviewing the low code approach

- How does a low code implementation look like?
- What are adventages and downsides of this approach?

2 Cleaning up

- What is clean architecture?
- How to implement a hexagonal architecute?
- How to write engine independent code?

3 Scaling up

- What is Domain Driven Design?
- What is Choreography?
- What are the best practices for domain events?

Demo –

Reviewing the initial state

Reviewing the low code approach

Advantages

- Rest Connectors enable **non-technical users** to create and modify automated processes
- Calling external services with the low code approach **reduces development time** and **costs**
- **Lowers the barrier** to entry for process automation

Downsides

- **Tighter coupling** and violation of the separation of concerns principle due to Connectors
- Can easier lead to a **vendor-lock-in** problem
- Need for a **greater dataset** flow through the process instance
- Misuse of the BPMN standard

Cleaning Up

“At its core, clean architecture divides a software system into parts, with each part having a specific responsibility and clear dependencies.”

Definition: Clean Architecture

Benefits: **Increased Maintainability, Testability, Scalability, Felixibility, Collaboration**

Writing Hexagonal Integrations

Ports and Adapters

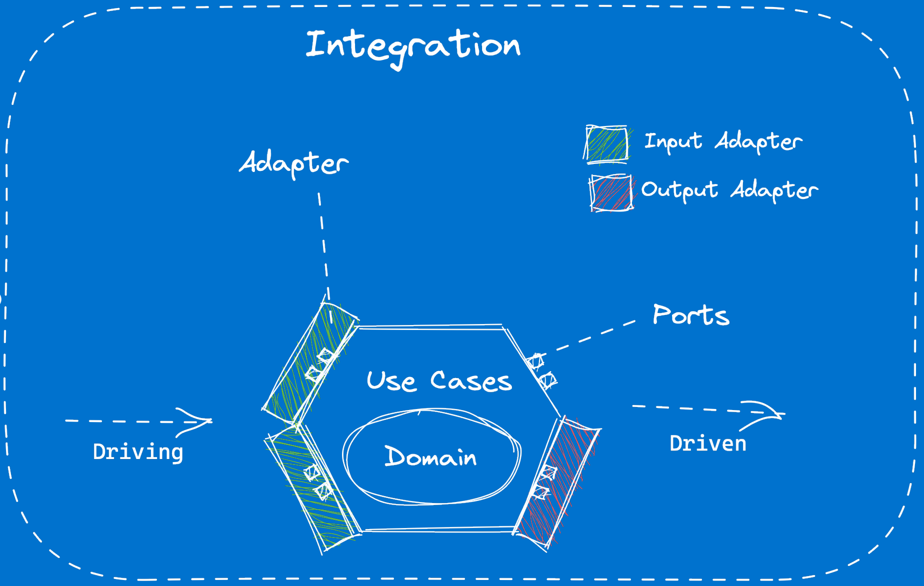
- We organise the hexagonal architecture into layers
- The outermost constist of adapters that translate between the application and other systems

No outgoing dependencies

- All dependencies point toward the center
- The Domain has no dependency towards the Use-Case or an Adapter

Benefits

- Truly technology neutral application core
- Easily adaptable to new technical surroundings
- Far easier maintainable

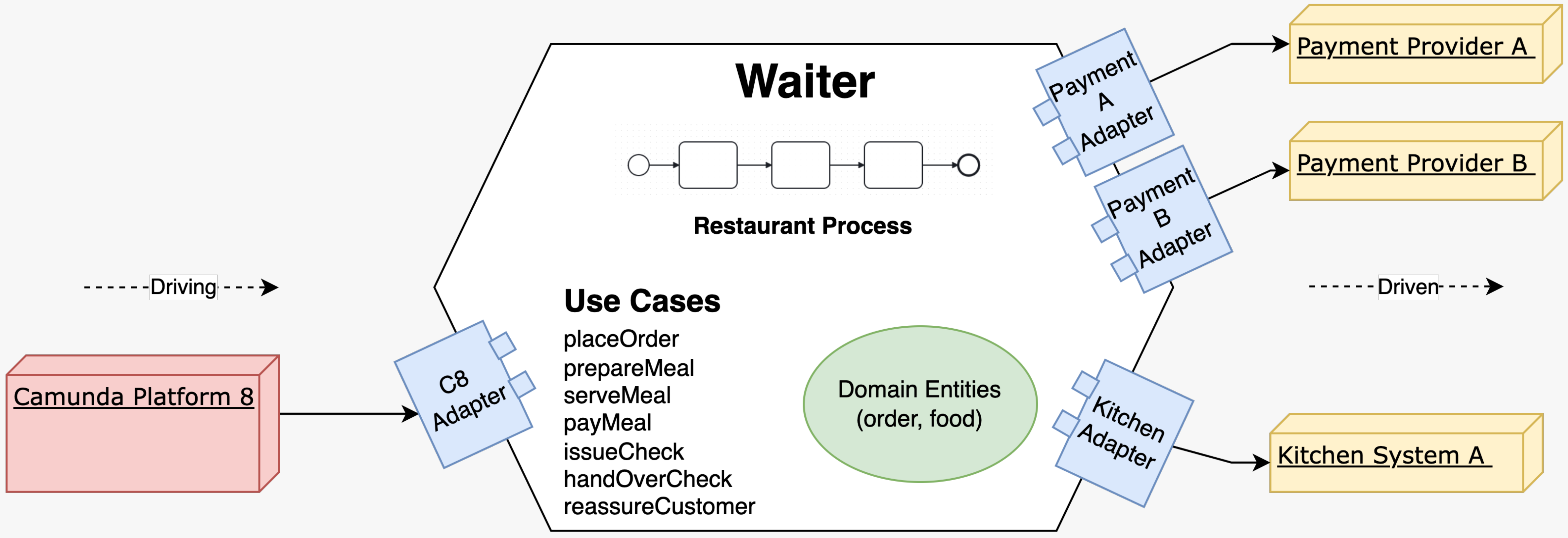


Integration Component

Using Hexagonal Architecture

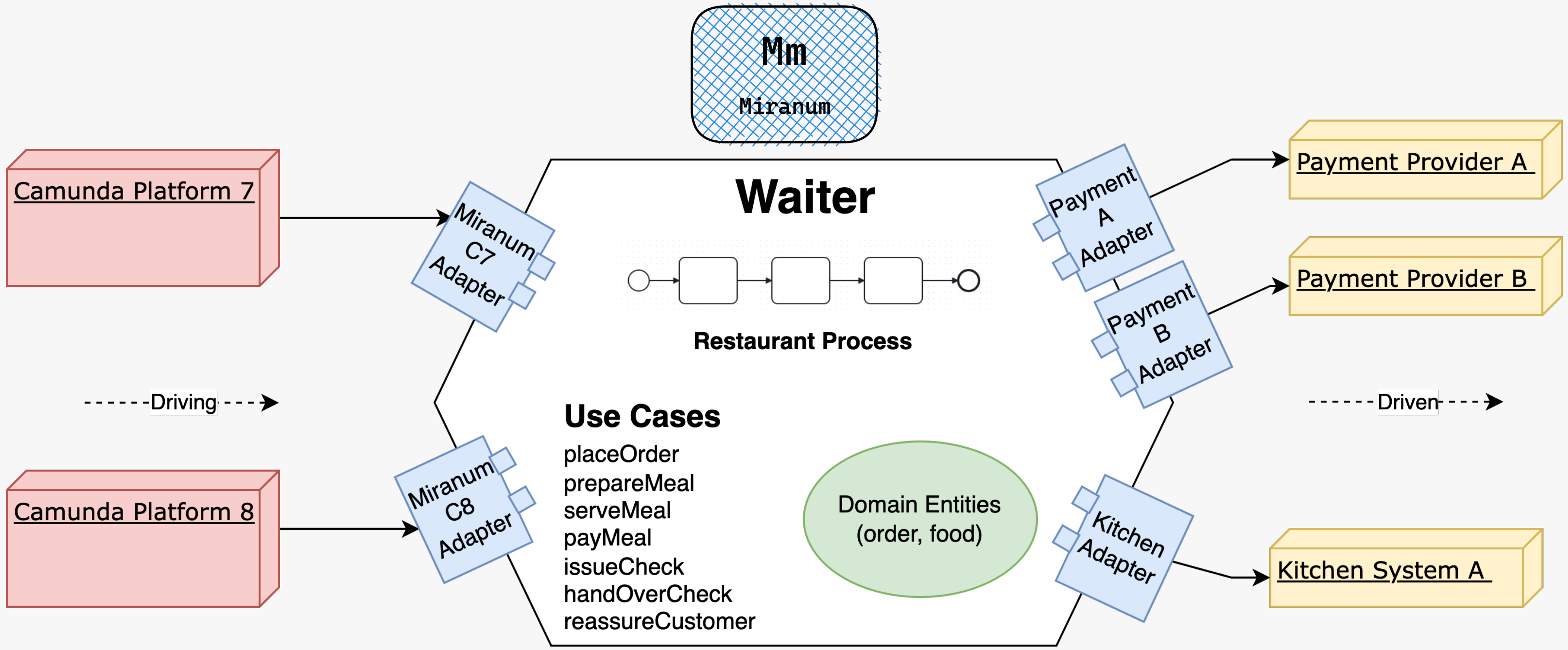
Reading recommendation: [Get your hands dirty with clean architecture]()

Clean Architecture Diagram



Clean Architecture Diagram

CCS 2023



Demo – **Implementation of a hexagonal Architecture**

Scaling up

Orchestration and Choreography

- excurse

CCS 2023

Orchestration

- Uses **command-driven** communication

Command = Sender wants something to happen. It has an intent. Recipient does not know who issued the command

Choreography

- Uses **event-driven** communication

Event = Something happended in the past. It is a fact. Sender does not know who picks up the event.

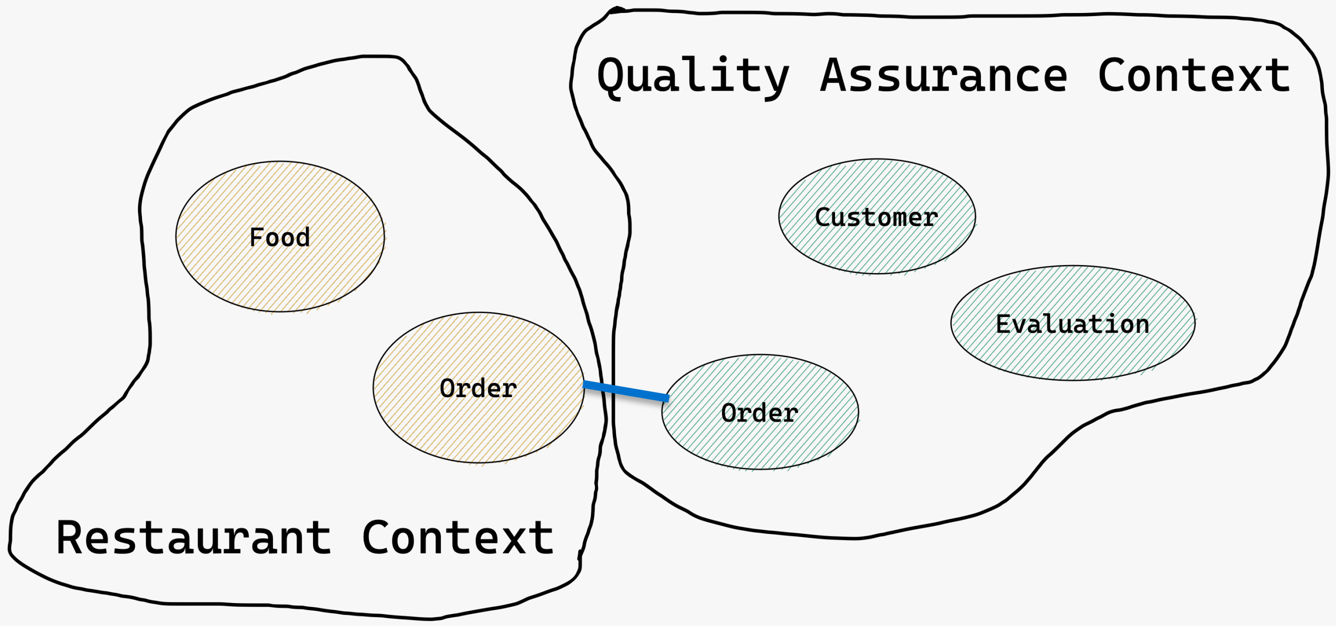
Domain Driven Design

CCS 2023

“Domain Driven Design is an approach to software development that centres on programming a domain model that has a rich understanding of the processes and rules of a domain.”

Martin Fowler

Bounded Context



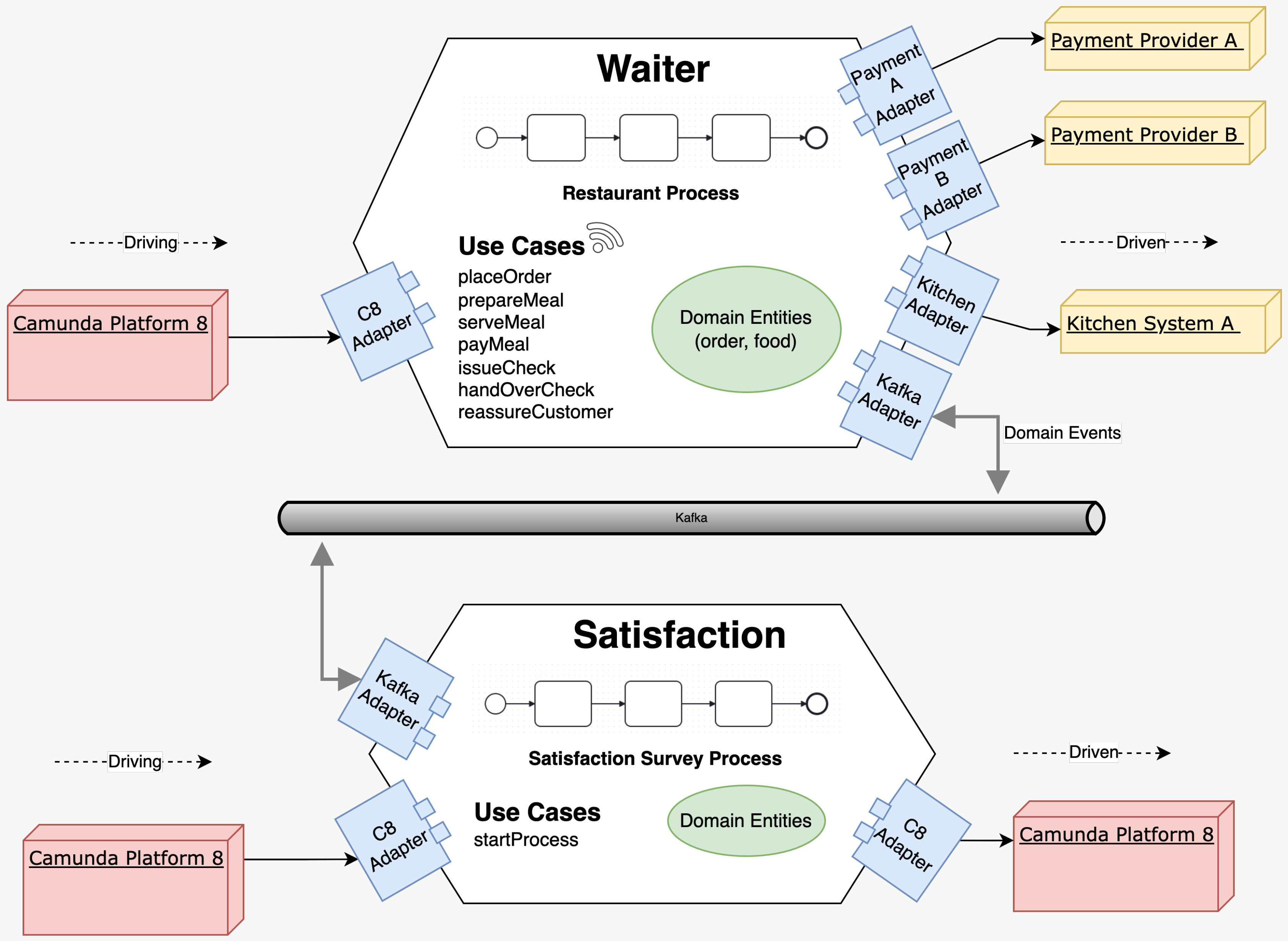
DDD devides up a large system into Bounded Contexts, which can have a unified model

Bounded Contexts have both, unrelated concepts but also some shared ones

Acts as a central patterns and allows to create dependencies based on the domain logic

Evolving our architecture

CCS 2023



Throwing the right domain events

- Best Practices

CCS 2023

- Use clean names and do not reuse events
- Throw events when you manipulate data (e.g. create, update, delete)
- Do not expect something in return after having thrown the event
- Try to use it only for communication out of the bounded context

Demo – **Choreography combined with Orchestration**

CCS 2023

Key Takeaways

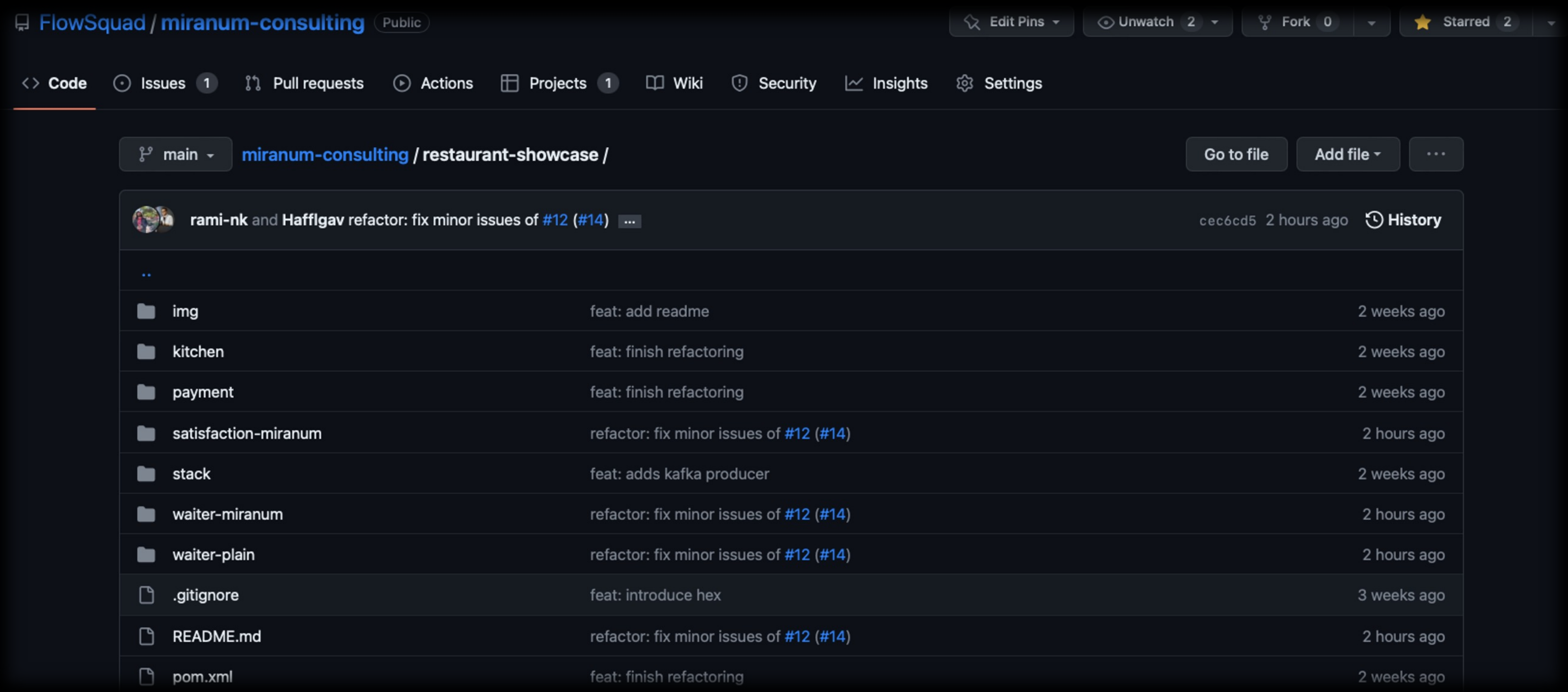
- Design your services technology and engine neutral
- Use Domain Events for communication outside the bounded context
- Gain better maintainability by using a hexagonal architecture
- Decouple your architecture with clear responsibilities for multiple teams

MIRAGON

CCS 2023

Thanks for listening!

https://github.com/FlowSquad/miranum-consulting/tree/main/restaurant-showcase


FlowSquad / miranum-consulting Public

Edit Pins · Unwatch 2 · Fork 0 · Starred 2

Code · Issues 1 · Pull requests · Actions · Projects 1 · Wiki · Security · Insights · Settings

main · miranum-consulting / restaurant-showcase /

Go to file · Add file · ...

rami-nk and Hafflgav refactor: fix minor issues of #12 (#14) ... cec6cd5 2 hours ago History

..

img	feat: add readme	2 weeks ago
kitchen	feat: finish refactoring	2 weeks ago
payment	feat: finish refactoring	2 weeks ago
satisfaction-miranum	refactor: fix minor issues of #12 (#14)	2 hours ago
stack	feat: adds kafka producer	2 weeks ago
waiter-miranum	refactor: fix minor issues of #12 (#14)	2 hours ago
waiter-plain	refactor: fix minor issues of #12 (#14)	2 hours ago
.gitignore	feat: introduce hex	3 weeks ago
README.md	refactor: fix minor issues of #12 (#14)	2 hours ago
pom.xml	feat: finish refactoring	2 weeks ago