Motivation for Loosely Coupled Software Systems

Or: Life is just a bakery

Stefan Bente

TH Köln
Cologne Institute for Digital Ecosystems (CIDE)
Software Architecture Lab (ArchiLab)

Technology
Arts Sciences
TH Köln

Agenda

- *Scenario 1:* It works (somehow)
- *Scenario 2:* The Orchestrator Pattern
- *Scenario 3:*
  Loosely Coupled and Event-driven
- DDD as a software specification & development approach

Scenario 1

It works (somehow)

1

I need a large birthday cake.
Cream, three layers, with fruit.
Can you make it Thursday?

Yes, that works. **2**

3

Julia, we need a 3-stack with fruit for Mrs. Gulbins. Please prepare the cream and tell Alfred.

4

Mrs. Gulbins, any food intolerances?

5

Alfred, a 3-stack with fruit for Mrs. Gulbins. Here is the cream.

6

Mrs. Gulbins, a chocolate base layer is always nice in a 3-stack. You want that?

7 Yes please!

8

Julia, I need chocolade whipped cream, too.

9

Martha, you need to add the chocolote base layer to the bill.

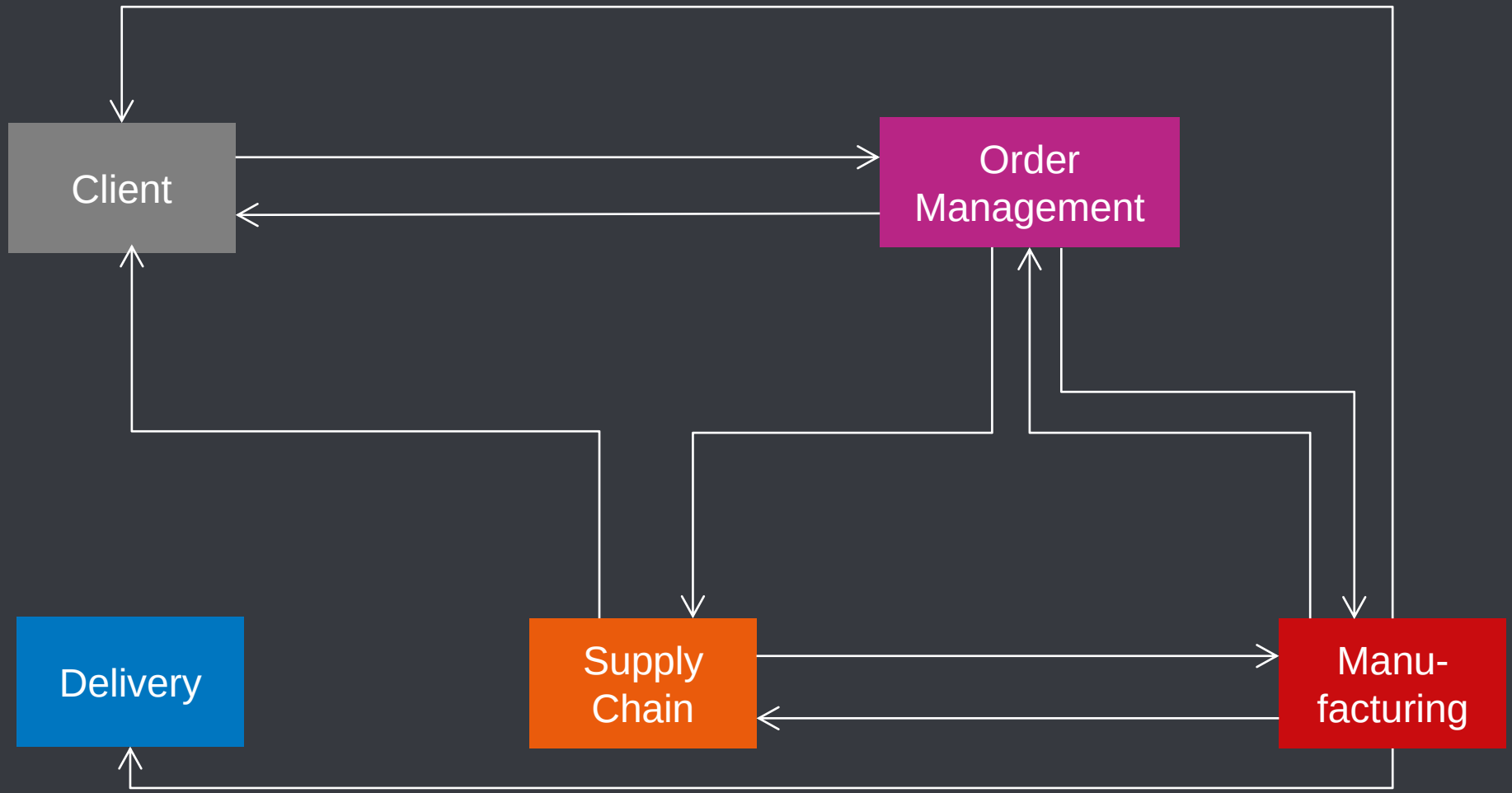
10

Julia, Alfred, what is your status with Mrs. Gulbins cake?

11

Done. You can send the the bill. I‘ll tell Juan.

12 Juan, can you please deliver this cake to Mrs. Gulbins?


Client
Order Management
Delivery
Supply Chain
Manu-facturing

Observation

- Each subsystem manages its own business logic
- **BUT:** Process knowledge distributed
  - Many subsystems know more then they they should
- (Wildly) cyclic dependencies

Scenario 2

The Orchestrator Pattern

1

I need a large birthday cake.
Cream, three layers, with fruit.
Can you make it Thursday?

Yes, that works.

Food intolerances?
Chocolate base layer?
…?

2

3

Julia, we need whipped cream.

4

Alfred, start with three layers.

5

Julia, now also chololate cream.

6

Alfred, please a chocolate base layer.

7

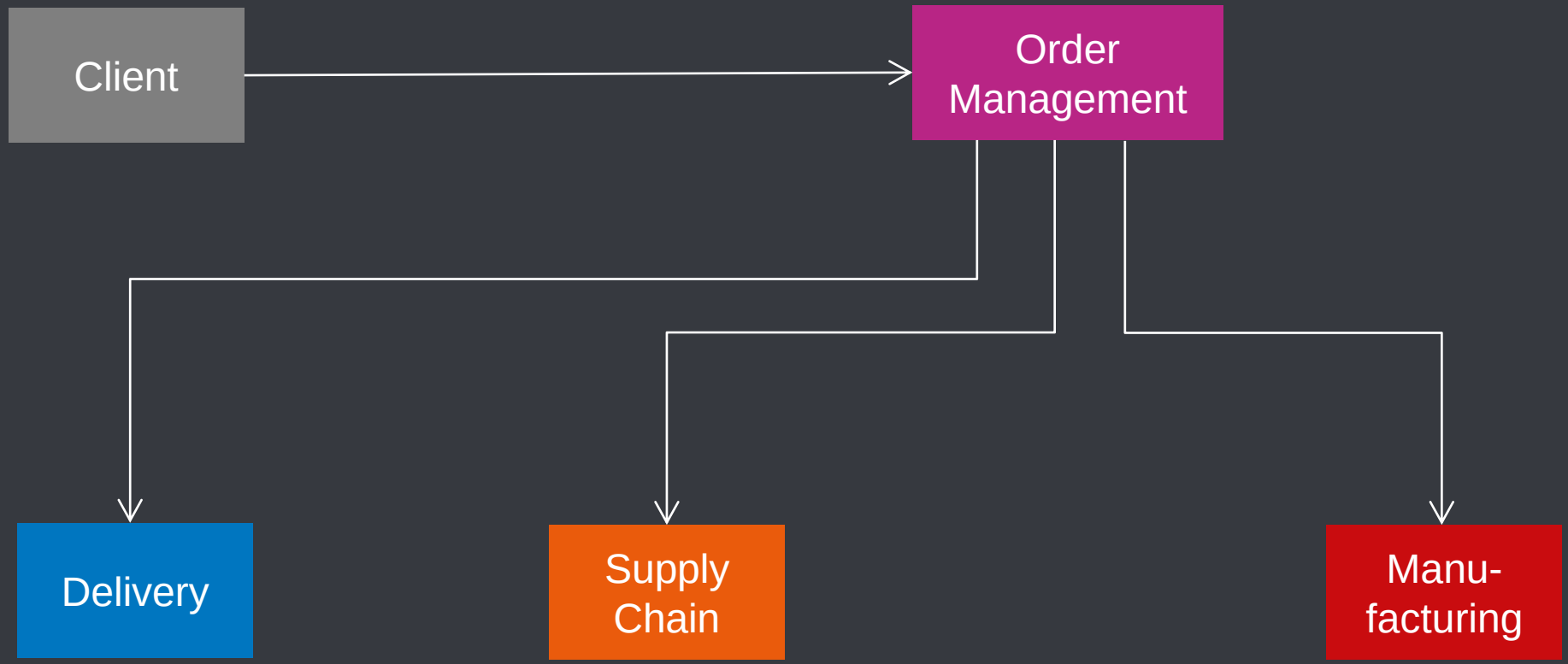
And now some fruit as decoration.

8

When ready, bring me the cake.

9

Juan, please deliver this cake to Mrs. Gulbins.


Client
Order Management
Delivery
Supply Chain
Manu-facturing

Observation

- Process knowledge nicely centralized
- No cycles
- **BUT:** Subsystems become anemic
  - (not the master of their own business logic)
- Orchestrator (Order Management) prone to become a „God System“
  - VERY hard to maintain

Scenario 3

Loosely Coupled and Event-driven

1

I need a large birthday cake.
Cream, three layers, with fruit.
Can you make it Thursday?

Yes, that works.

Food intolerances?
Chocolate base layer?
…?

2

3

3-Stack, fruit,
no food intolerances,
chocolate base layer

I'll prepare the bill. The team knows what to do ☺

OK, I need to prepare whipped cream and chocolate cream …

OK, I need to prepare the layers …

4

Whipped cream ready!

OK, I can mount the layers …

… and now the base layer and the fruit …

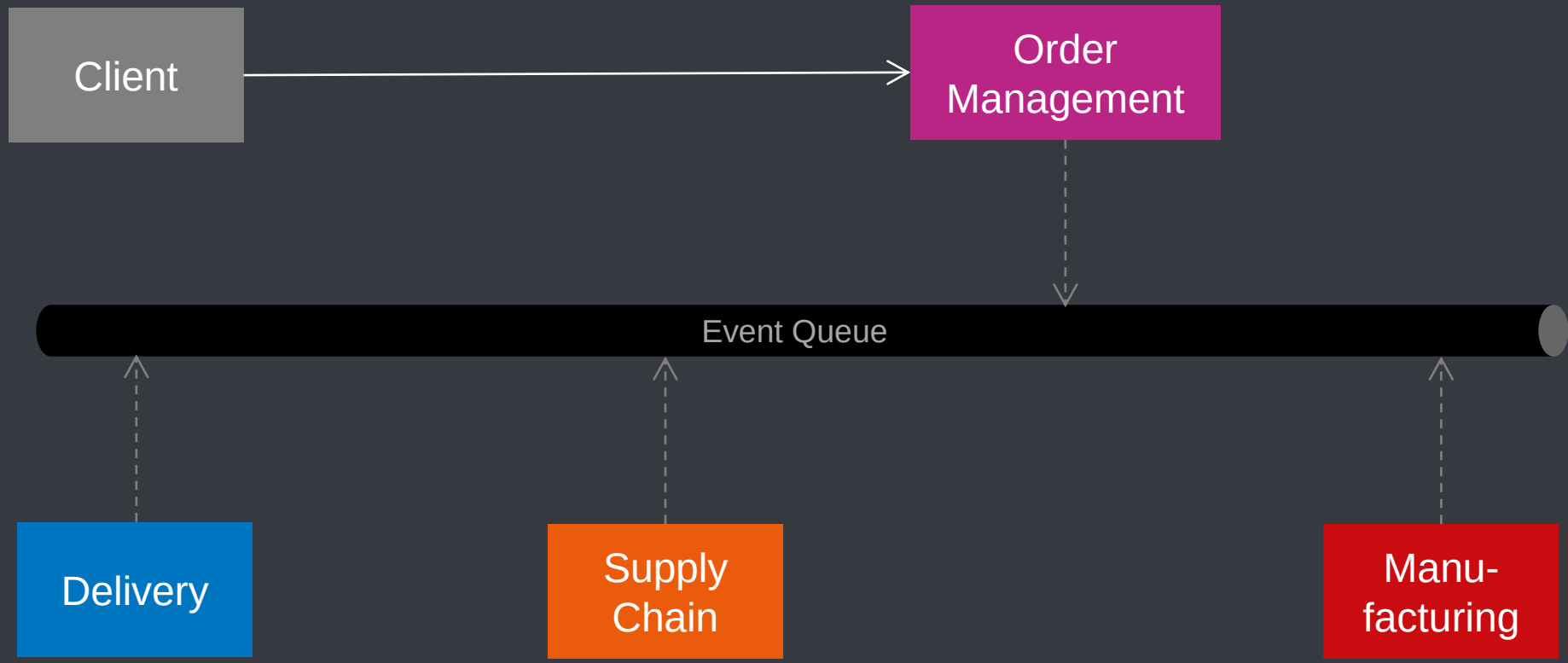
5

Cholcolate cream ready!

6

Cake for Mrs Gulbins is ready.

OK, so I can deliver it.


Client
Order Management
Event Queue
Delivery
Supply Chain
Manu-facturing

Observation

- „Empowered“ yet loosely coupled subsystems - no cycles
- **BUT:** Harder to design
  - Thinking in events doesn‘t come natural
- More technical effort
  - No / very little synchronous calls
  - Asynchronous communication & data redundancy

www.archi-lab.io