How we built our AI assistant

Must-See Case Study!

Boozt

Boozt Fashion AB
Hyllie Boulevard 35
SE-215 32 Malmö

Viktor Pikaev

- Senior backend developer at Boozt
- 15+ years of experience
- Active contributor to Yii2 ecosystem
- A big fan of Rust and Haskel
- Plays Factorio

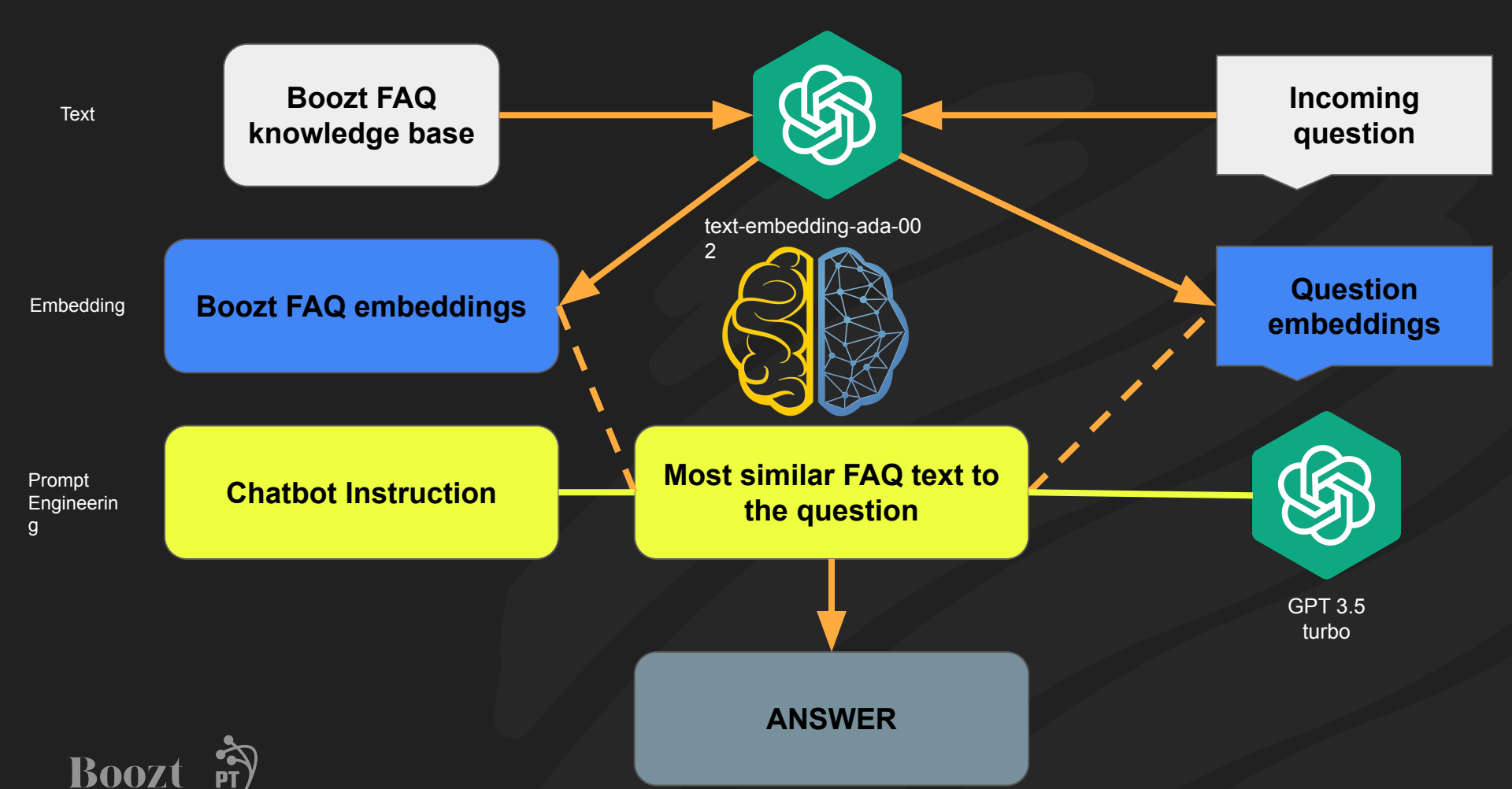
Boozt PT

The base idea

Boozt FAQ knowledge base

Boozt FAQ chatbot

Boozt PT


Text
Boozt FAQ knowledge base
Incoming question
text-embedding-ada-002
Embedding
Boozt FAQ embeddings
Question embeddings
Prompt Engineering
Chatbot Instruction
Most similar FAQ text to the question
GPT 3.5 turbo
ANSWER
Boozt

AI is not a magic wand

It's important!

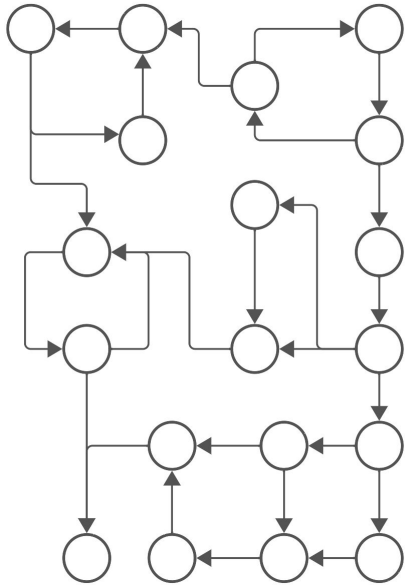
Complexity management is a cornerstone of any changeable system.

Reduce (hide) complexity

Make ways of extension and modification easy and obvious

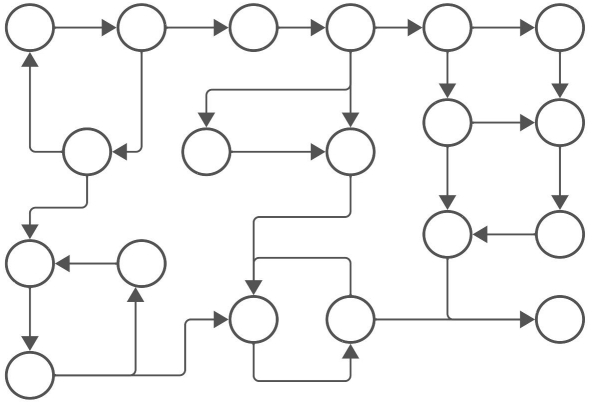
Why is it complex?

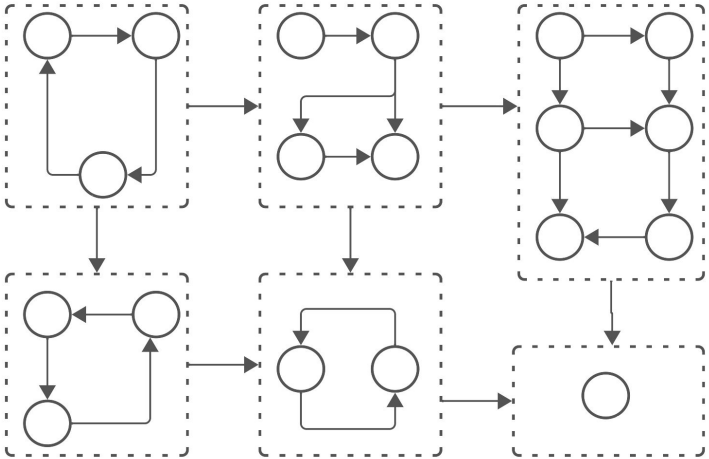
- Conversation is a directed graph (a big one)
- With cycles
- Edges can be enabled or disabled dynamically
- Conversation has a changeable context
- Some subgraphs have their own contexts
- Every income message should be processed with extra checks
- All of this is constantly changing and expanding



We can't reduce the complexity

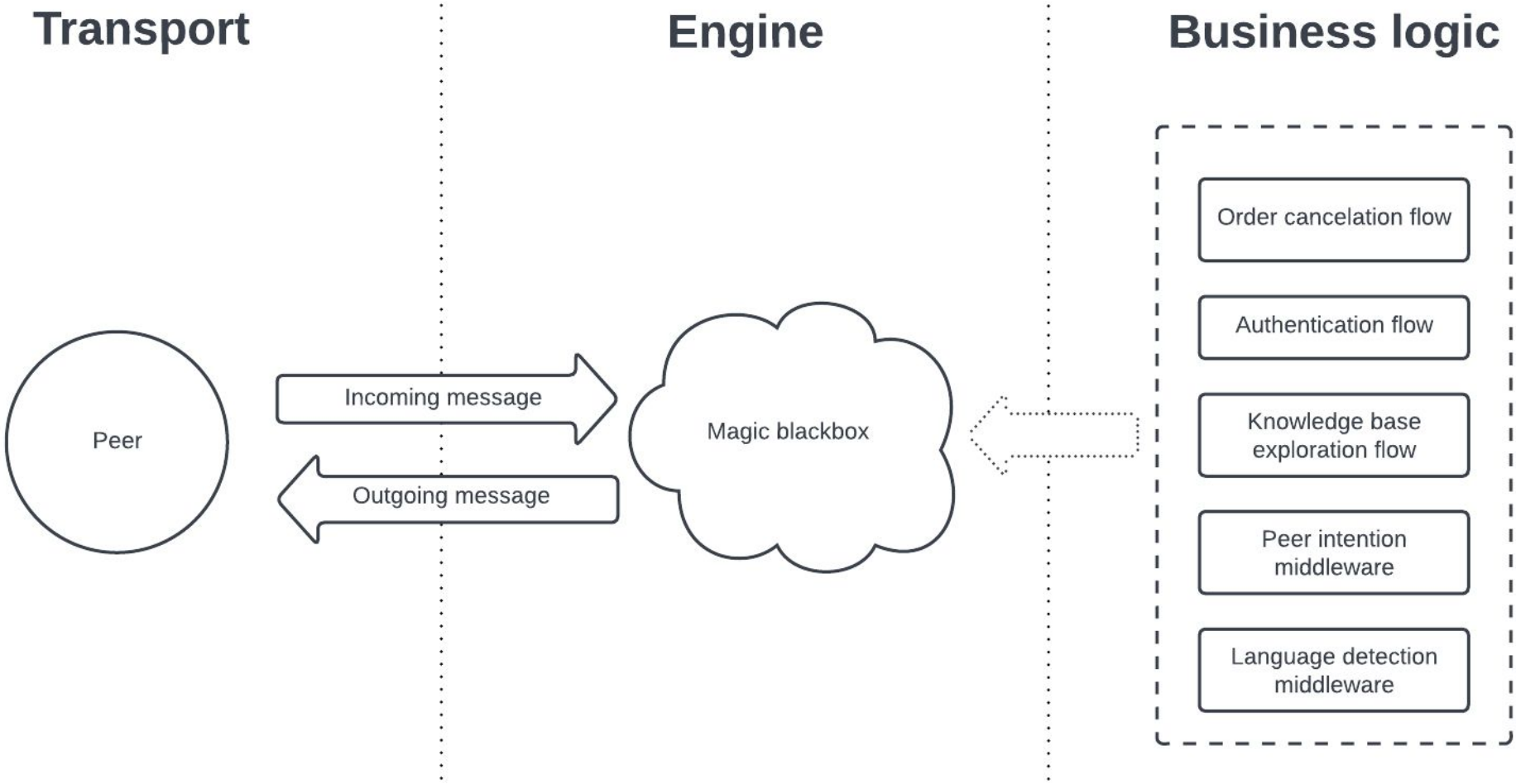
We can only hide it



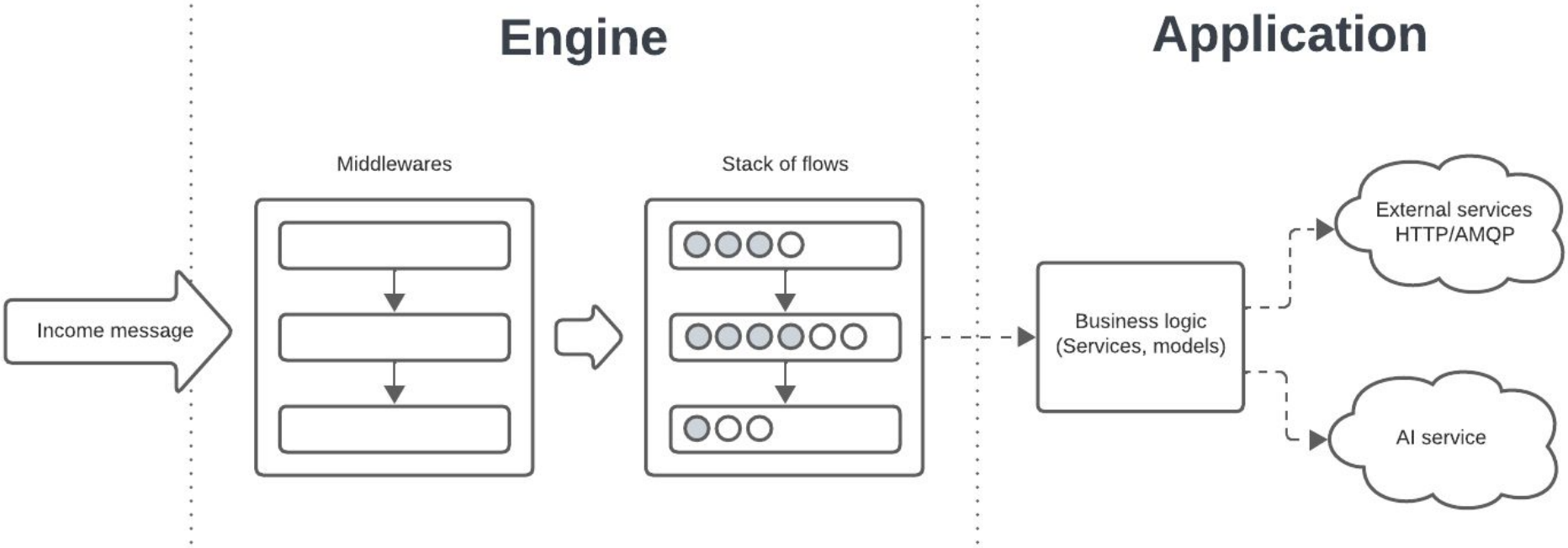


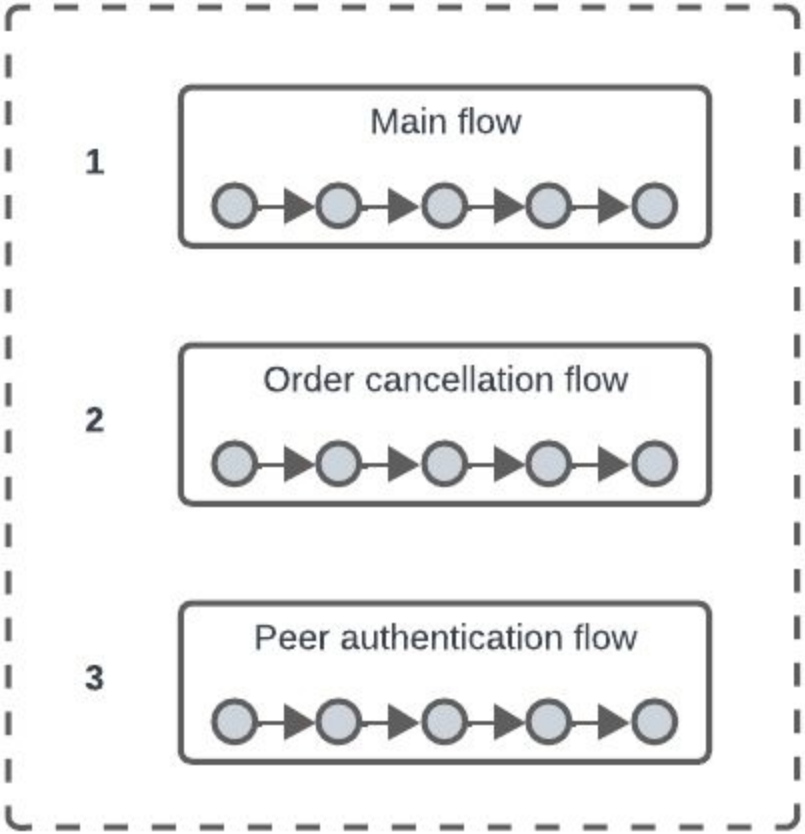
How to hide the complexity?

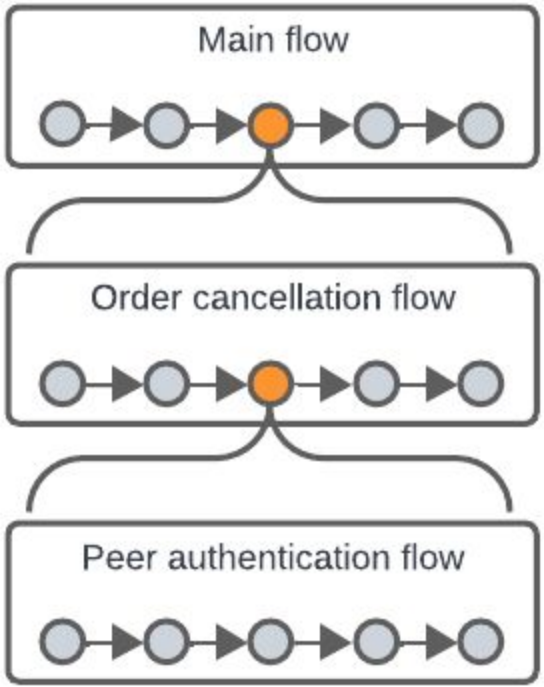
- We can split a big graph to several smaller and transit between them
- We can add a middleware layer for all income messages
- We can use Symfony custom tags and PHP attributes to avoid configuration files

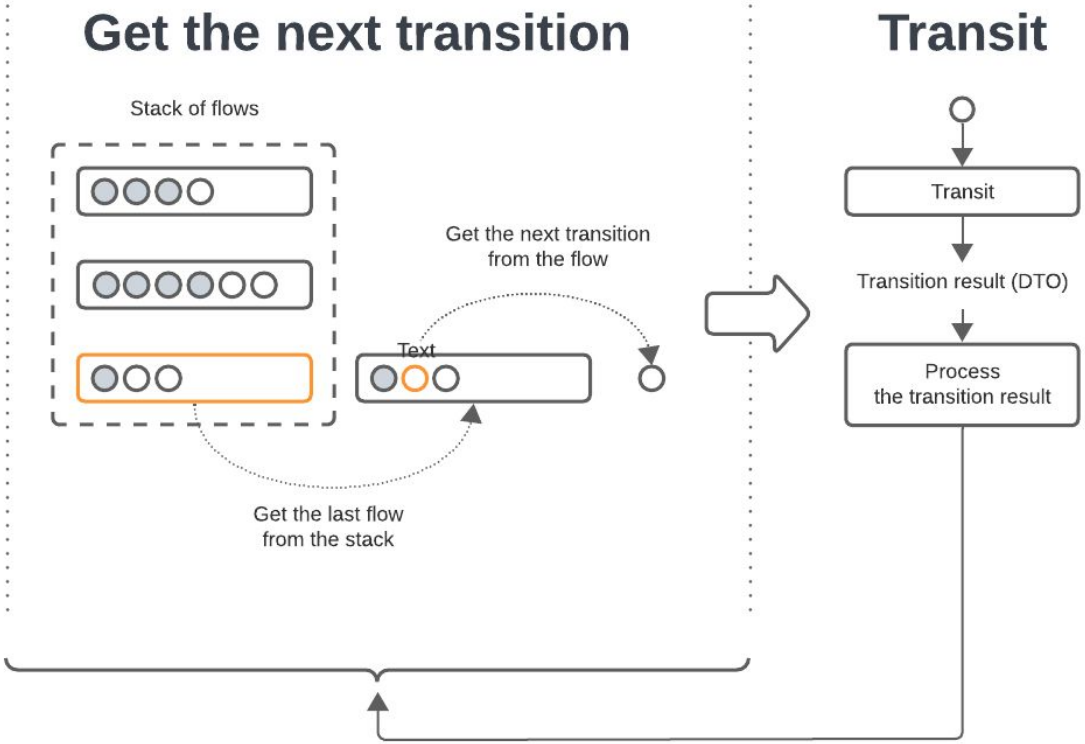

Transport
Engine
Business logic
Peer
Incoming message
Outgoing message
Magic blackbox
Order cancelation flow
Authentication flow
Knowledge base exploration flow
Peer intention middleware
Language detection middleware

The Engine


Engine
Application
Middlewares
Stack of flows
Income message
Business logic
(Services, models)
External services
HTTP/AMQP
AI service


1
Main flow
2
Order cancellation flow
3
Peer authentication flow


Main flow
Order cancellation flow
Peer authentication flow



Transition loop

- Get a flow from the top of a stack
- Get the last transition from the flow
- Run the transition and get a result DTO
- Handle the result DTO
- Go to the next iteration

Business logic extension

Add a new middleware

```php
#[Middleware]
class PeerHasTerminatedIntentionMiddleware implements MiddlewareInterface
{
    public function process(Peer $peer, ?IncomingMessage $message): bool
    {
        // $peerContext = $peer->getContext();
        // ...
    }
}
```

Boozt PT

Add a new flow

```php
#[Flow(configClass: FlowConfig::class, contextClass: FlowContext::class)]
final readonly class KnowledgeBaseExplorationFlow implements FlowInterface
{
    public function __construct(private FlowConfig $config, private FlowContext $context)
    {
    }

    public function getConfig(): FlowConfig
    {
        return $this->config;
    }

    public function getContext(): FlowContext
    {
        return $this->context;
    }
}
```

Boozt PT

Add a new transition

```php
/**
 * @implements TransitionInterface<KnowledgeBaseExplorationFlow>
 */
#[Transition(flowClass: KnowledgeBaseExplorationFlow::class, name: ProcessQuestion::NAME)]
class ProcessQuestion implements TransitionInterface
{
    public const string NAME = 'process-question';

    public function transit(
        ?IncomingMessage $message,
        KnowledgeBaseExplorationFlow $flow,
        Peer $peer
    ): TransitionResultInterface {
        // $flowContext = $flow->getContext();
        // $flowConfig = $flow->getConfig();
        // $peerContext = $peer->getContext();
        // ...
    }
}
```

Boozt PT

Easy extension with Symfony

- Middleware implements the MiddlewareInterface and has the #[Middleware] attribute

- Flow implements the FlowInterface and has the #[Flow] attribute
- If the flow needs to handle the result of the subflow it implements the SubFlowResultHandler interface

- Transition implements the TransitionInterface and has teh #[Transition] attribute
- Transition returns one of the Result DTOs

Boozt PT

Questions ?

Boozt PT