L-Università ta' Malta

ixaris

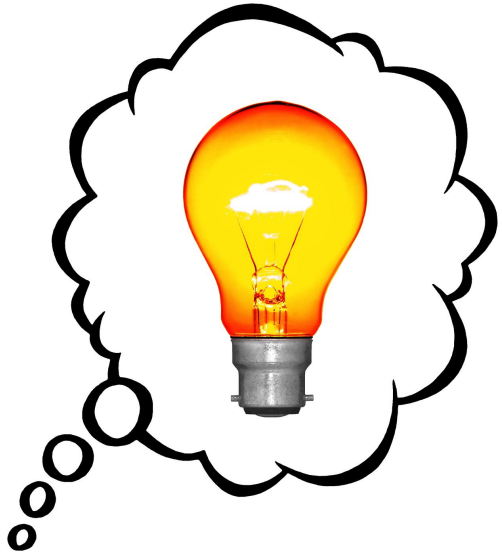
Valour -
An Industrial RV Tool for Transaction Systems

Shaun Azzopardi and Christian Colombo
University of Malta

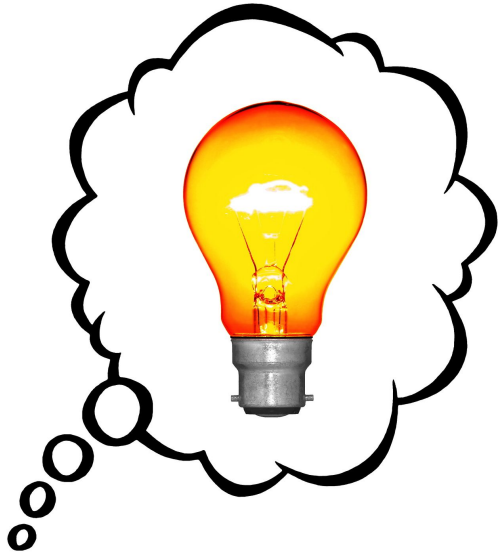
In collaboration with Ixaris Ltd, funded by the European Union's Horizon 2020 programme, under grant number 666363

One morning…

You wake up with a great idea…



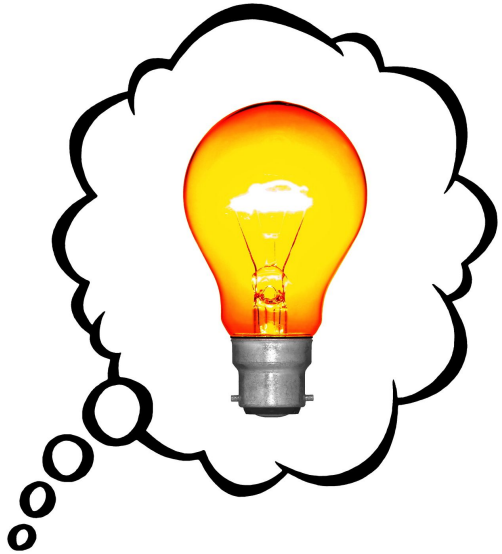
One morning…



You wake up with a great idea…

A mobile app which gives users complete visibility of their **money spending habits**

One morning…

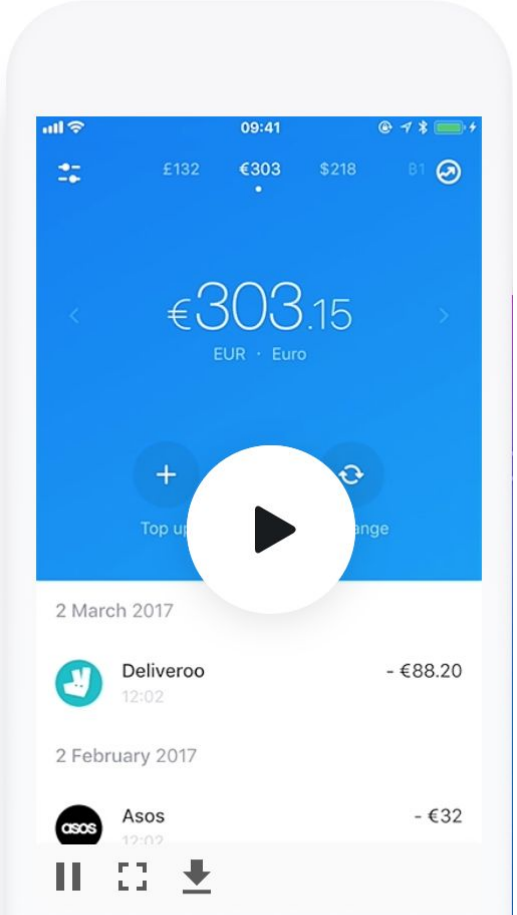


You wake up with a great idea…

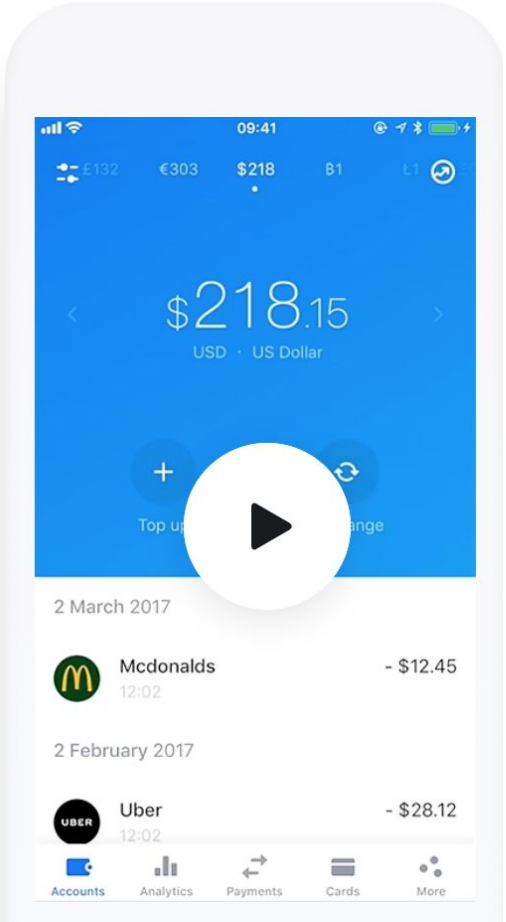
A mobile app which gives users complete visibility of their **money spending habits**

It allows them to **budget their salary**, save every month, …

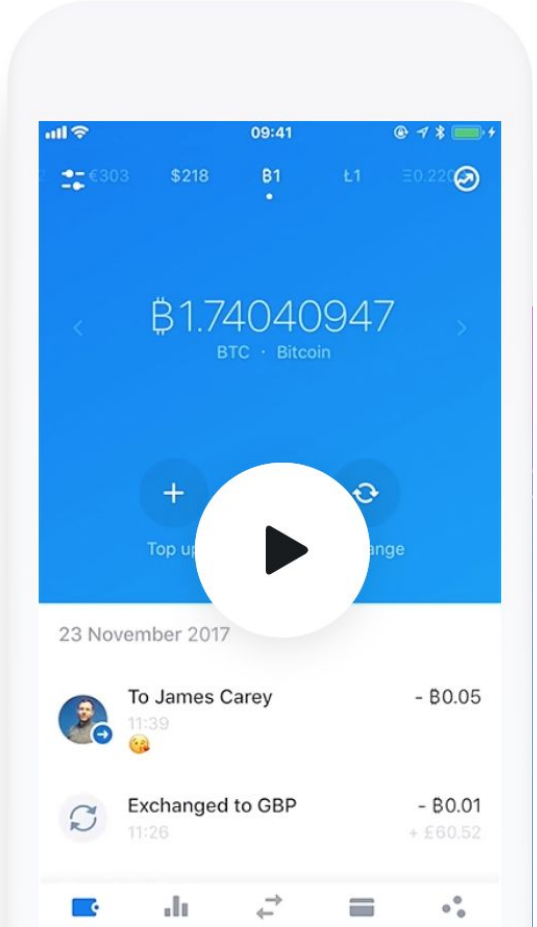
You start imagining how it would look...



You start imagining how it would look...



You start imagining how it would look...



There is just one problem...

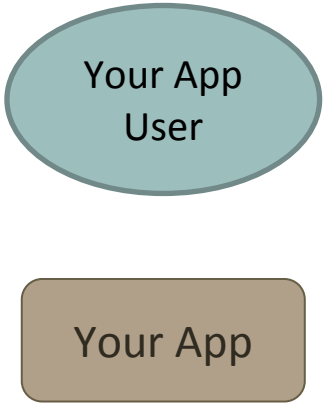
You need a **bank license**!

Payment programme setup costs

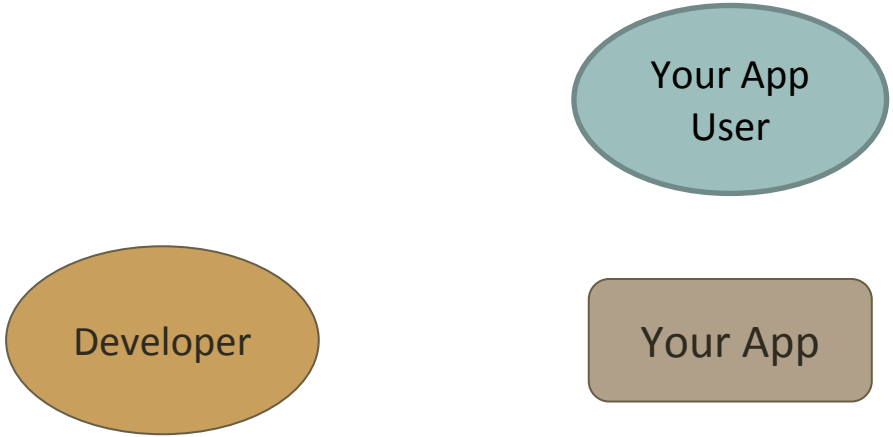
- Implementing card processes
- Agreement with bank
- Compliance to legislation
- Auditing
- Dispute resolution
- ...

Building a payment application

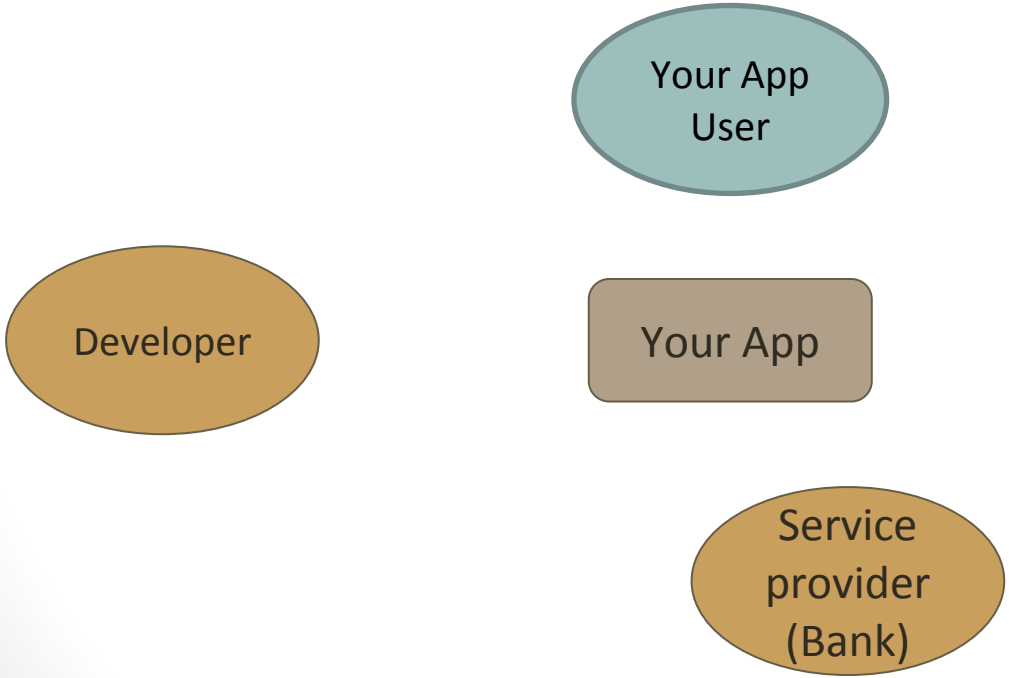
Building a payment application



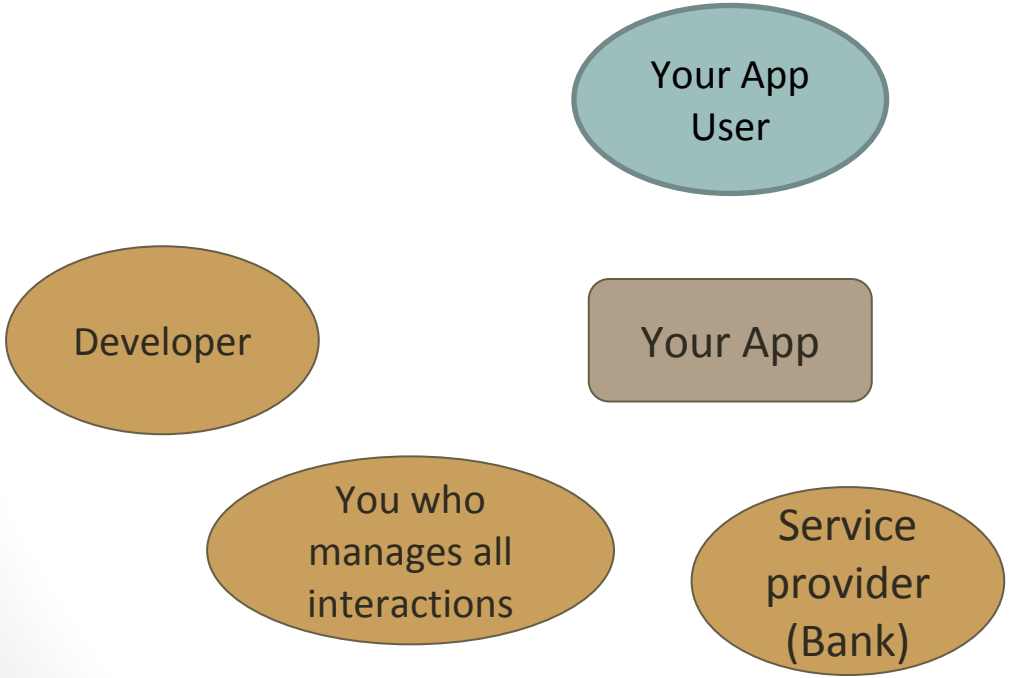
Building a payment application



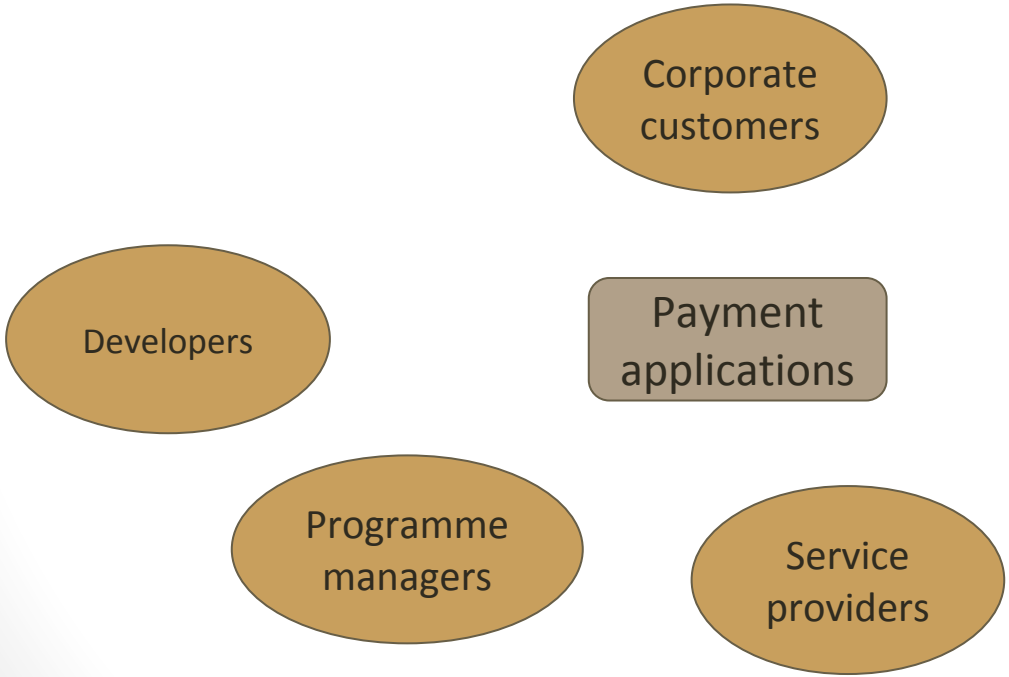
Building a payment application



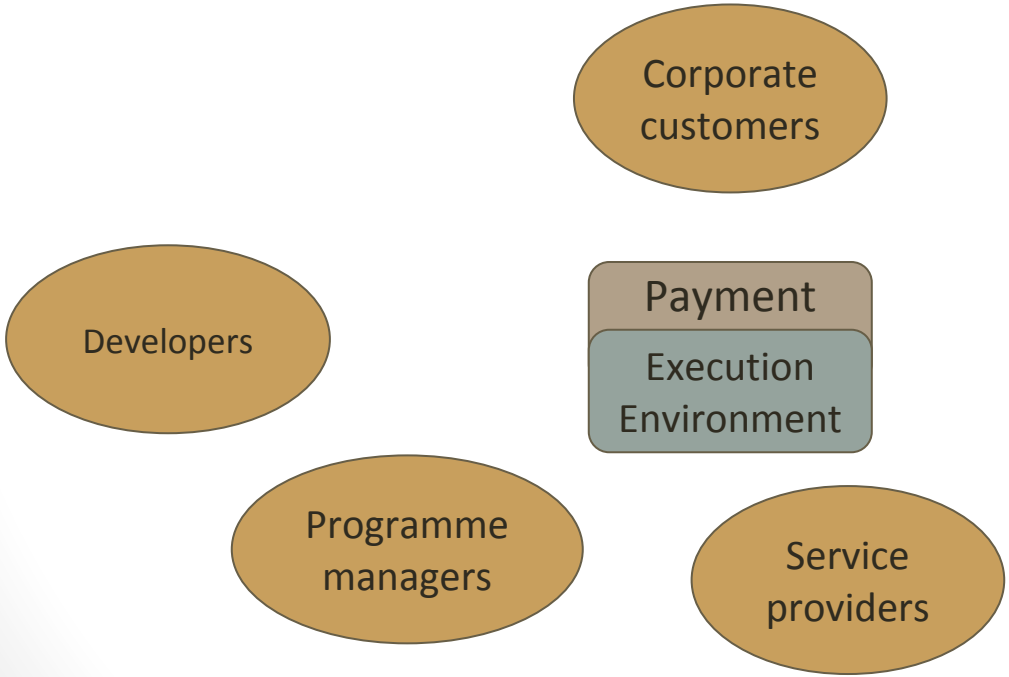
Building a payment application



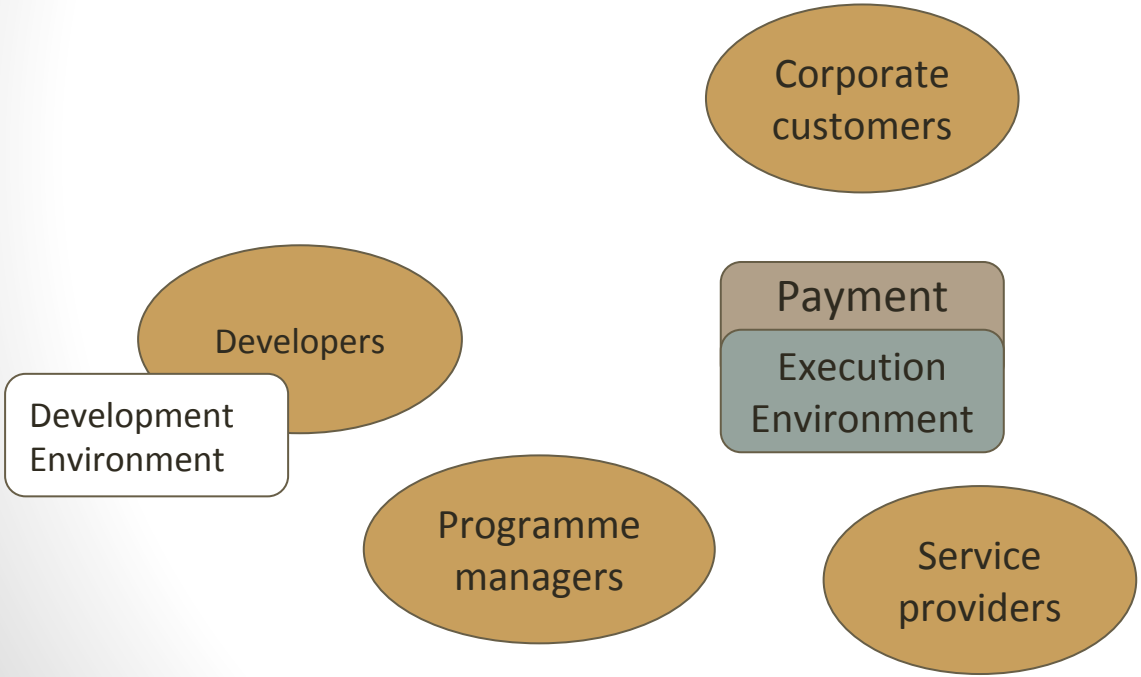
Open Payments Ecosystem



Open Payments Ecosystem



Open Payments Ecosystem



Open Payments Ecosystem

Corporate customers

Developers

Development Environment

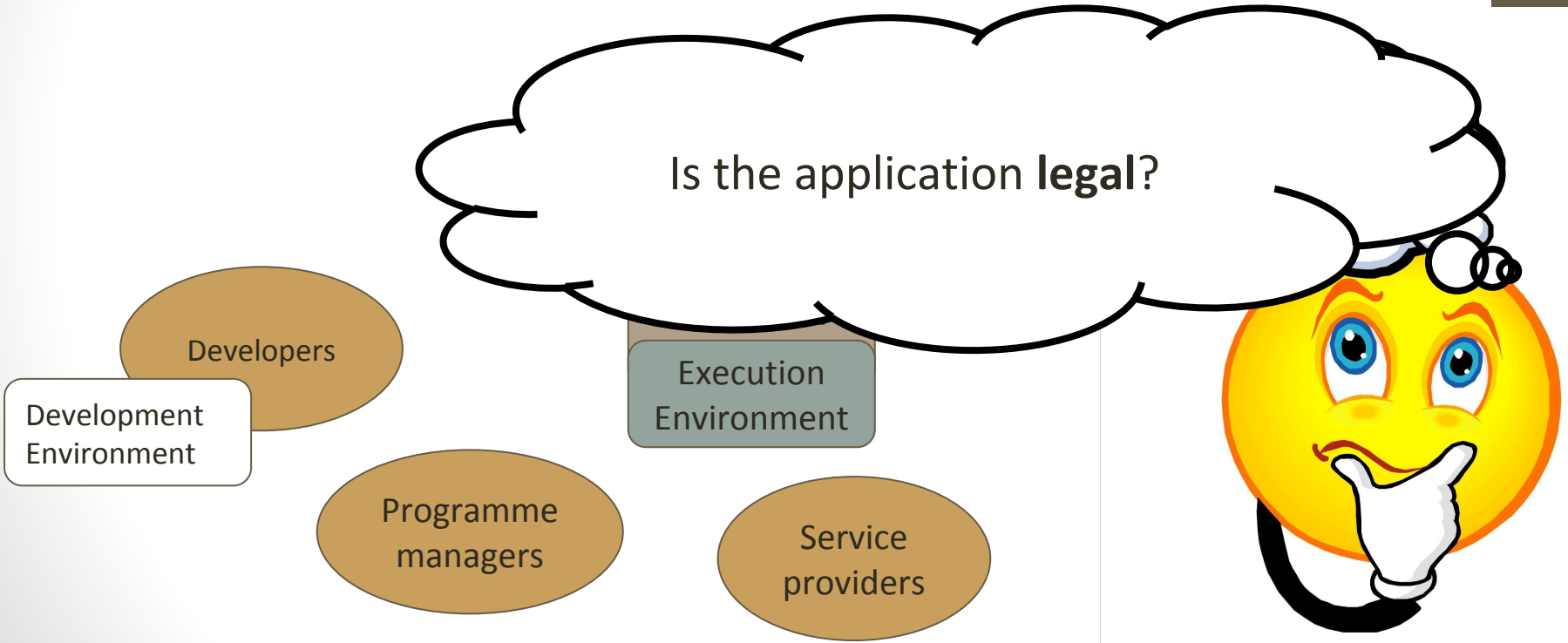
Payment

Execution Environment

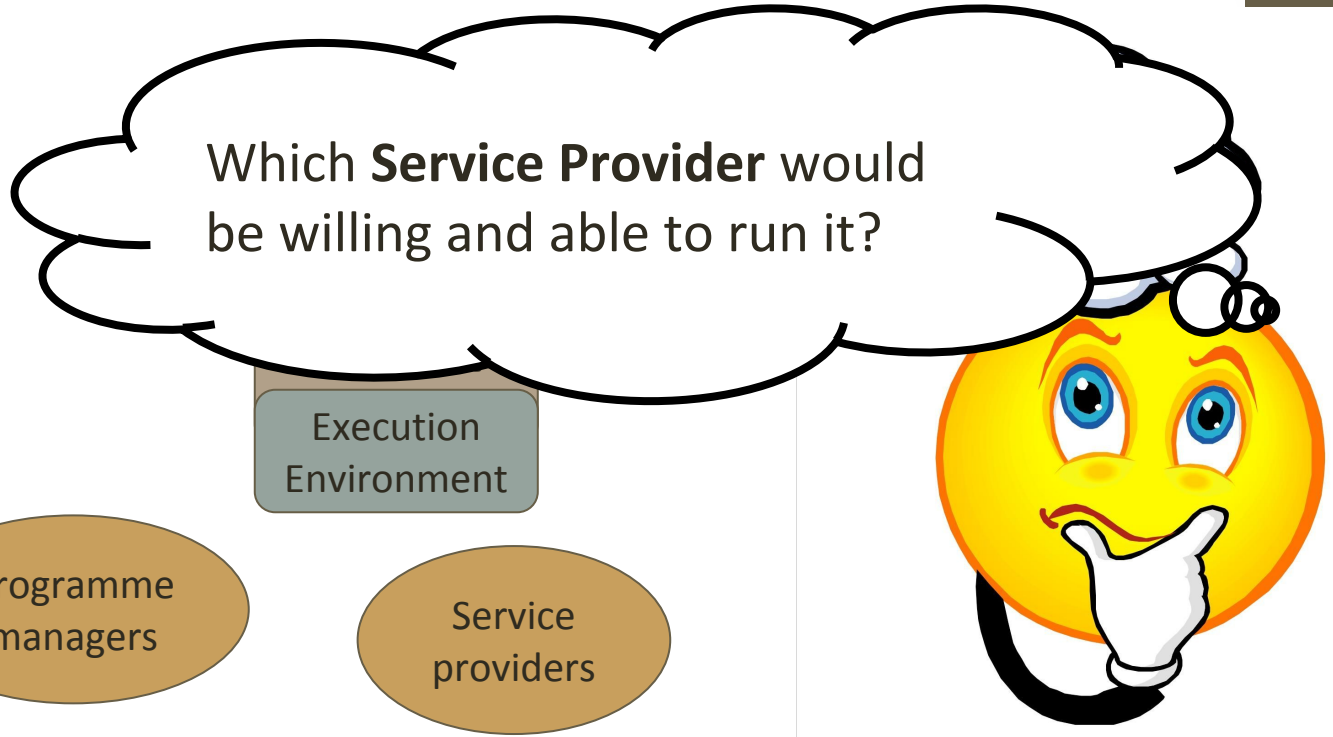
Programme managers

Service providers

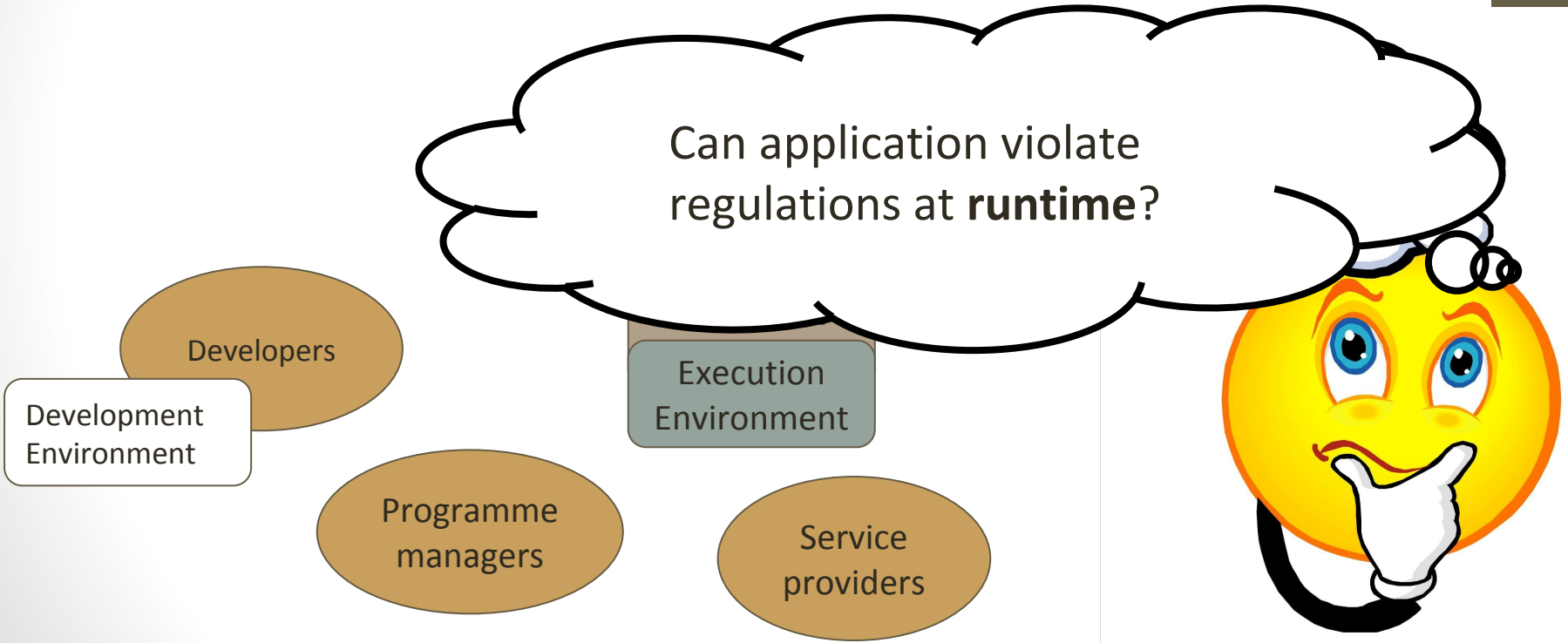
Open Payments Ecosystem



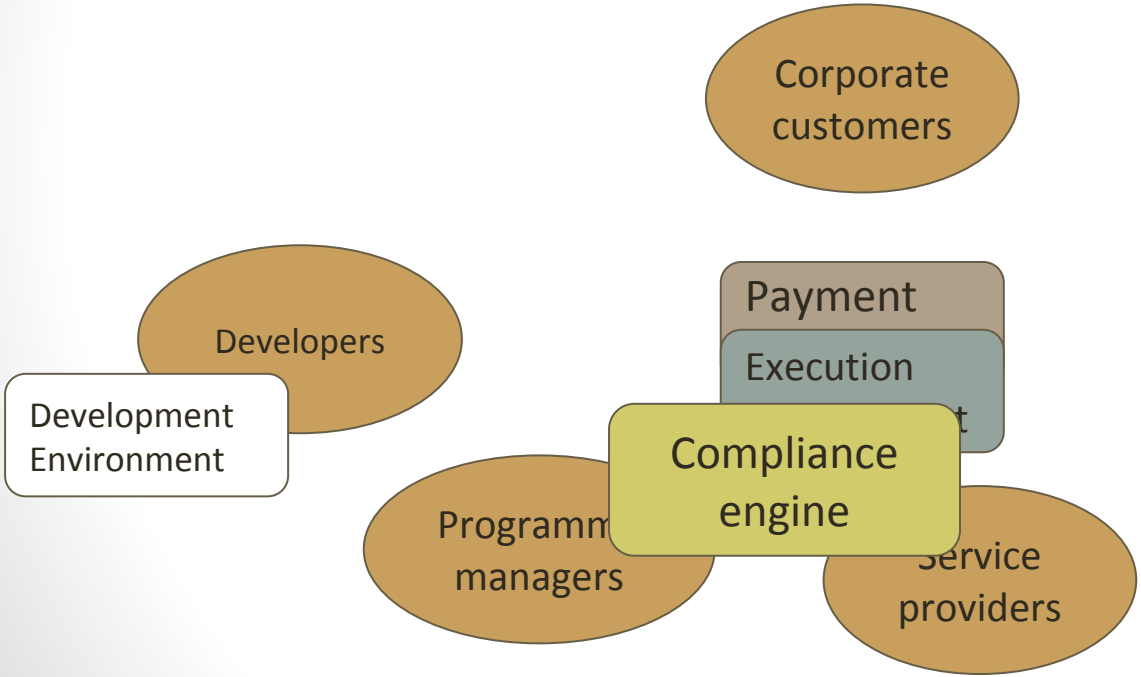
Open Payments Ecosystem



Open Payments Ecosystem



Open Payments Ecosystem + Compliance



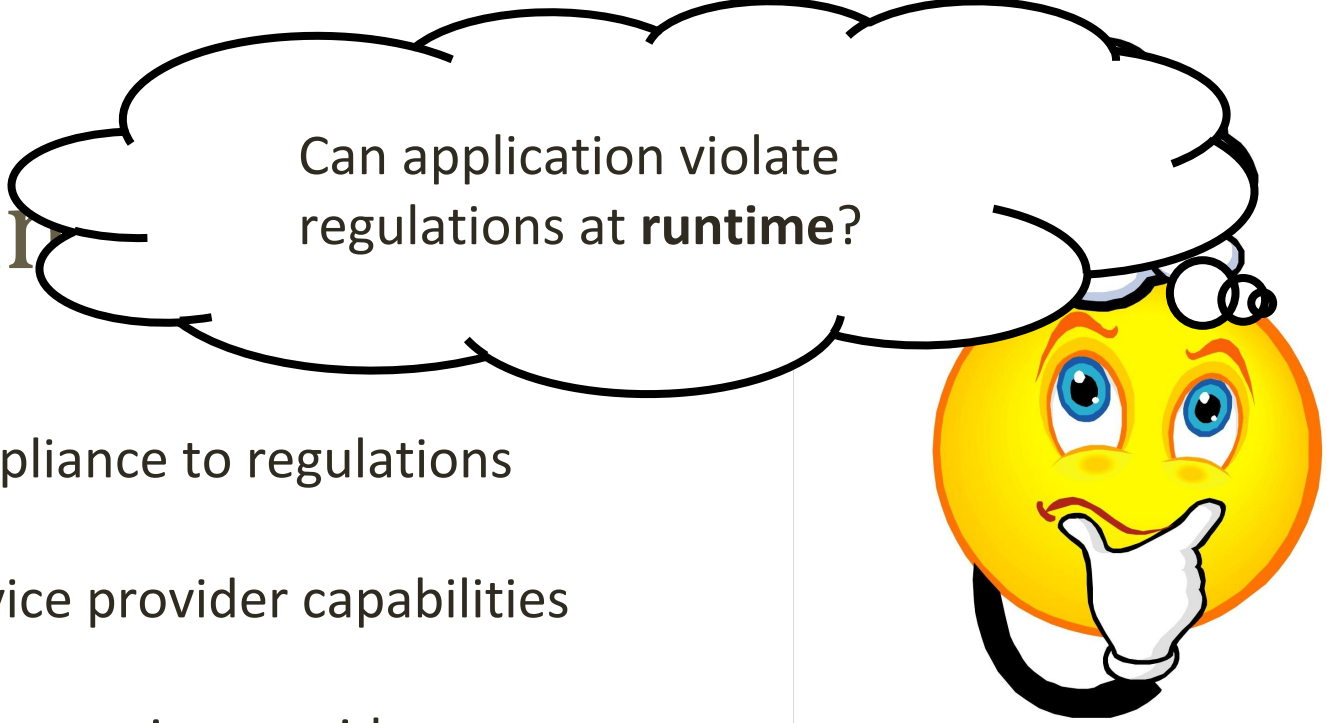
Compliance

1. Checking compliance to regulations

Compliance

1. Checking compliance to regulations
2. Matching service provider capabilities

Compliance



1. Checking compliance to regulations
2. Matching service provider capabilities
3. Limiting risk for service providers

Example

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

1. Is the jurisdiction the UK?

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

1. Does the application fall under the definition of e-money?

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Are funds redeemable through the application?

1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Is correct value given to the user

1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

2. Can service provider support e-money applications?

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

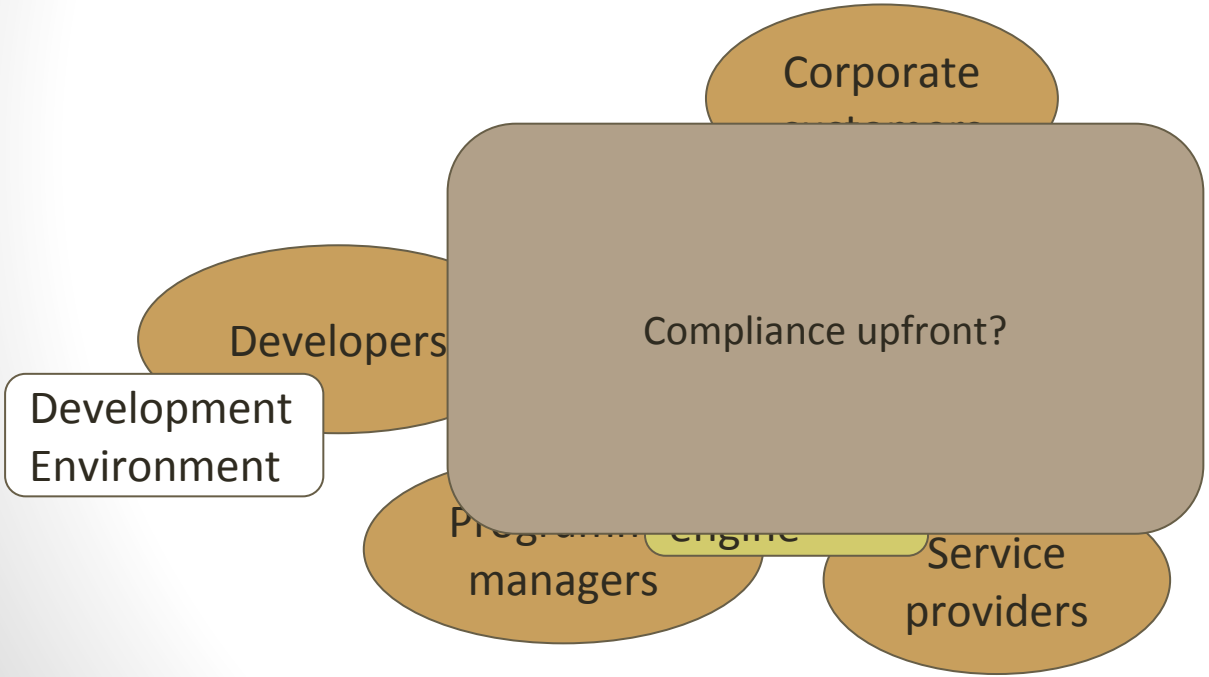
1. Compliance to regulations
2. Capability matching
3. Risk mitigation

Example

UK e-money regulations state that funds on financial instruments should be redeemable at par value.

3. How much money is allowed on instruments?

Open Payments Ecosystem + Compliance



Compliance Challenges

- Not all properties are checkable upfront

Implication: SA not enough

Compliance Challenges

- Not all properties are checkable upfront

Implication: SA not enough

- Application is run by third party:
  - Not all information is available

Implication: SA can only be done on info provided

Compliance Challenges

- Not all properties are checkable upfront

Implication: SA not enough

- Application is run by third party:
  - Not all information is available
  - We cannot trust the application

Implication: SA can only be done on info provided

Implication: We have to verify model adherence at runtime

Compliance Challenges

- Not all properties are checkable upfront

Implication: SA not enough

- Application is run by third party:
  - Not all information is available
  - We cannot trust the application

Implication: SA can only be done on info provided

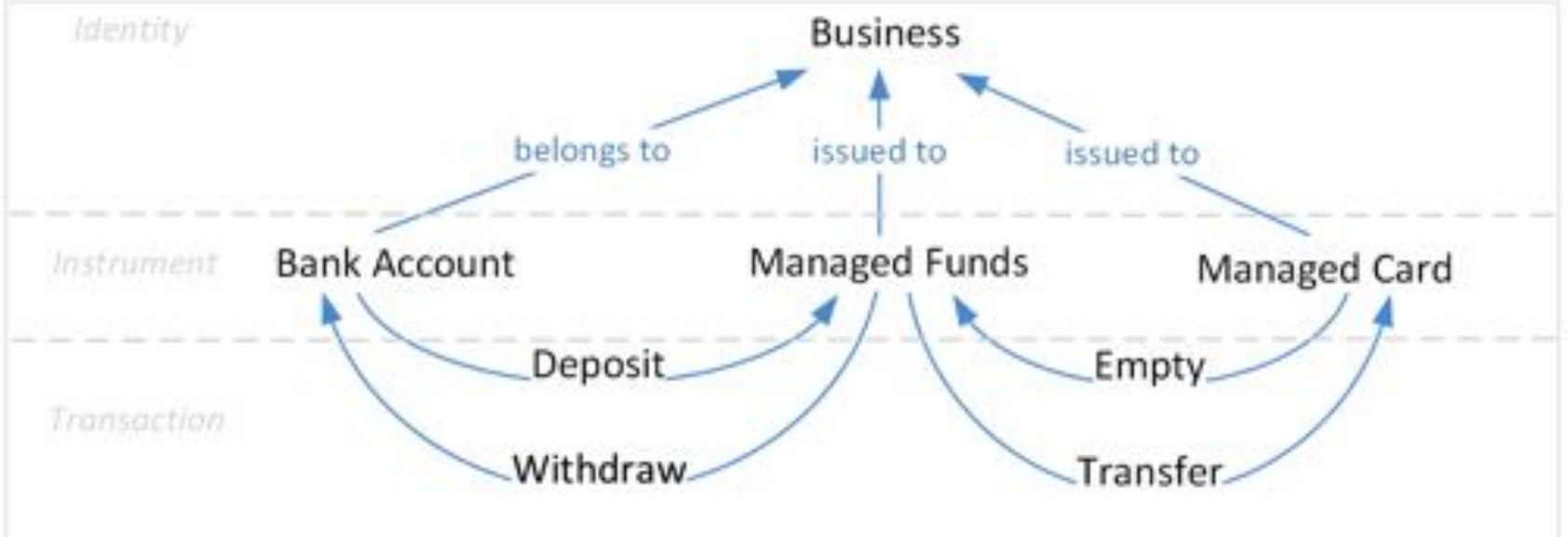
Implication: We have to verify model adherence at runtime

- Understanding the domain
  - Frequent legislation changes
  - Academics will not remain involved

Implication: We need a common language

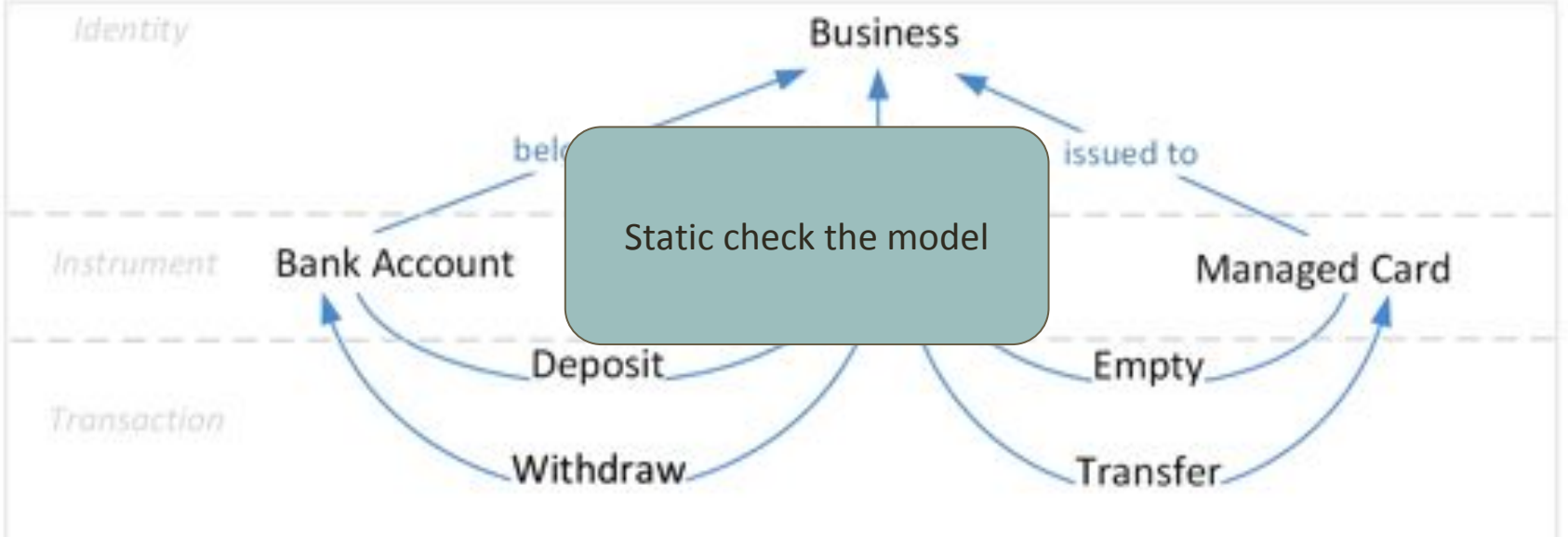
Info Provided by Developer - The Model

Developer submits model of application rather than implementation



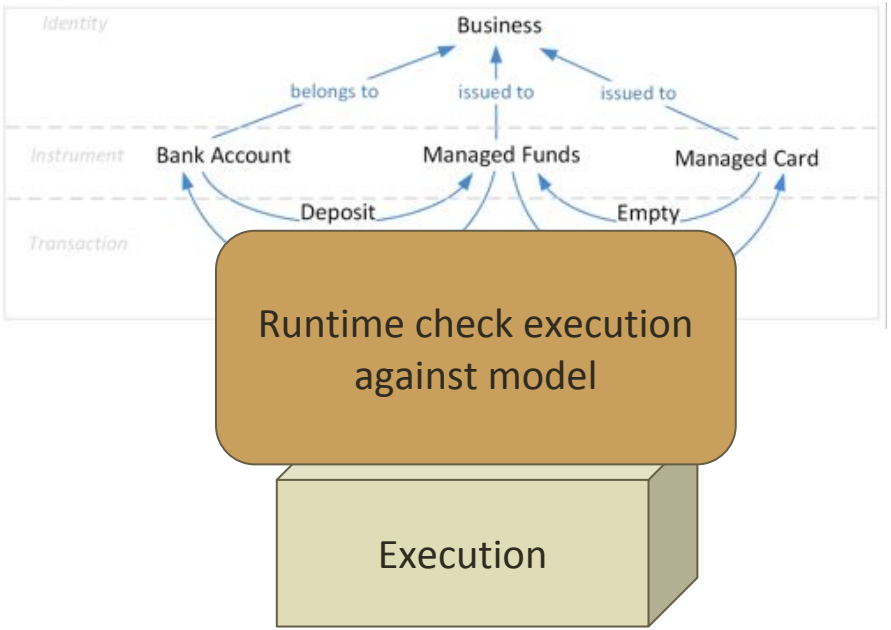
Info Provided by Developer - The Model

Developer submits model of application rather than implementation

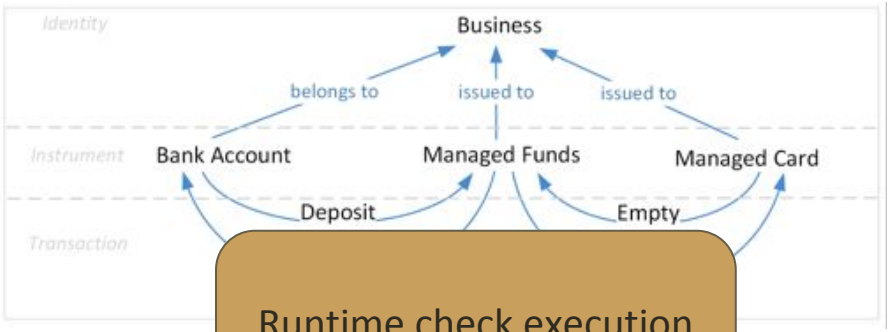


Static check the model

Application Execution



Application Execution



Runtime check execution against model

Execution

Runtime verify remaining checks, eg: limits, at par value, delays

Example

UK e-money regula ncial instruments should be rede

1. Is the jurisdiction the UK?

2. Can service provider support e-money applications?

1. Does the application fall under the definition of e-money?

1. Are funds redeemable through the application?

Static check the model

Example

Runtime check implementation against model

UK e-money regulations state that funds on financial instruments should be redeem

1. Does the application fall under the definition of e-money?

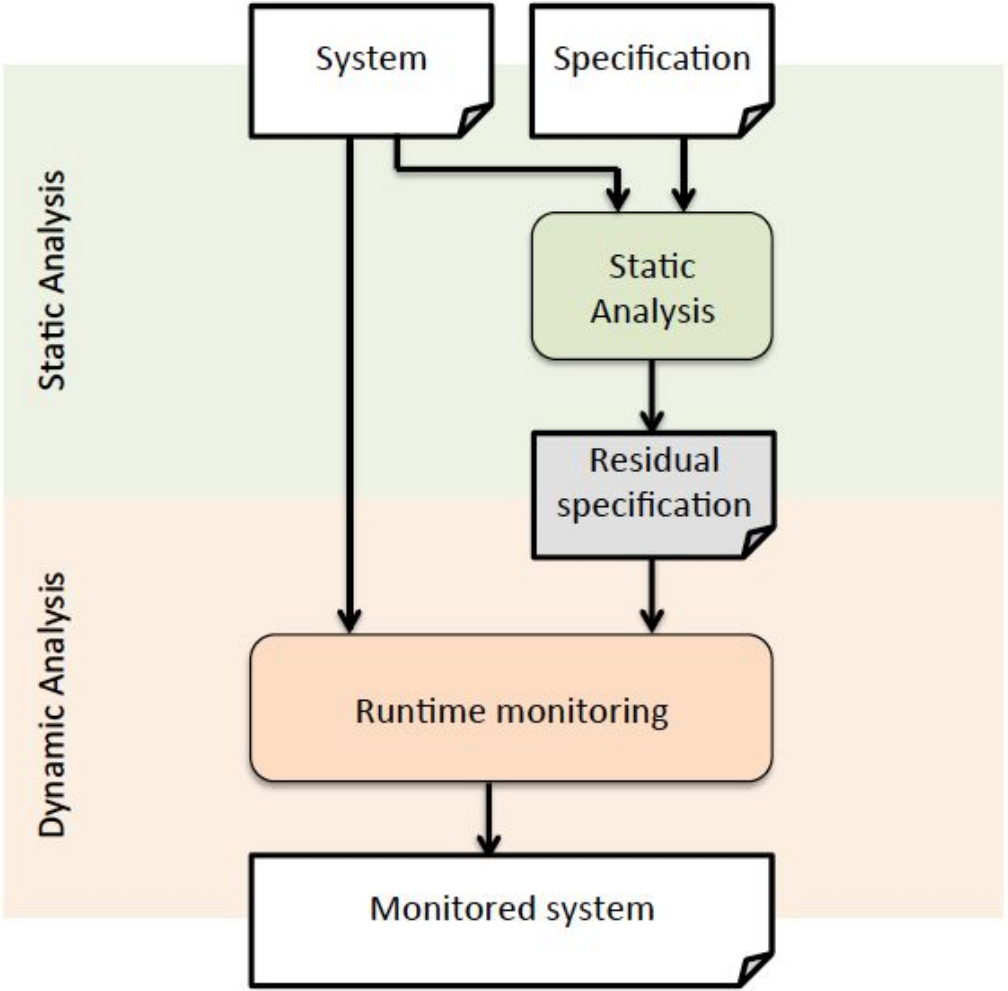
Example

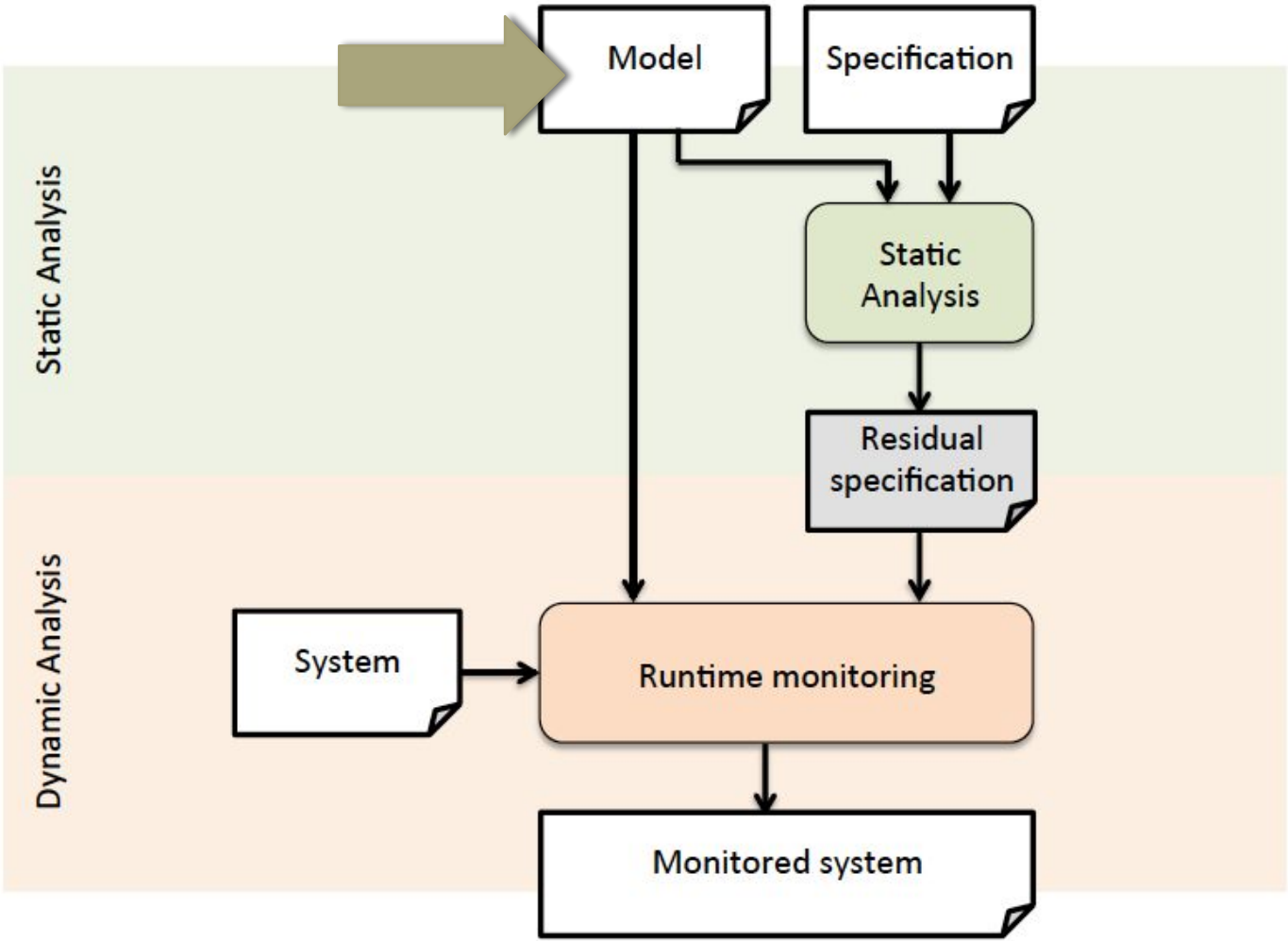
Runtime verify remaining checks,
eg: limits, at par value, delays

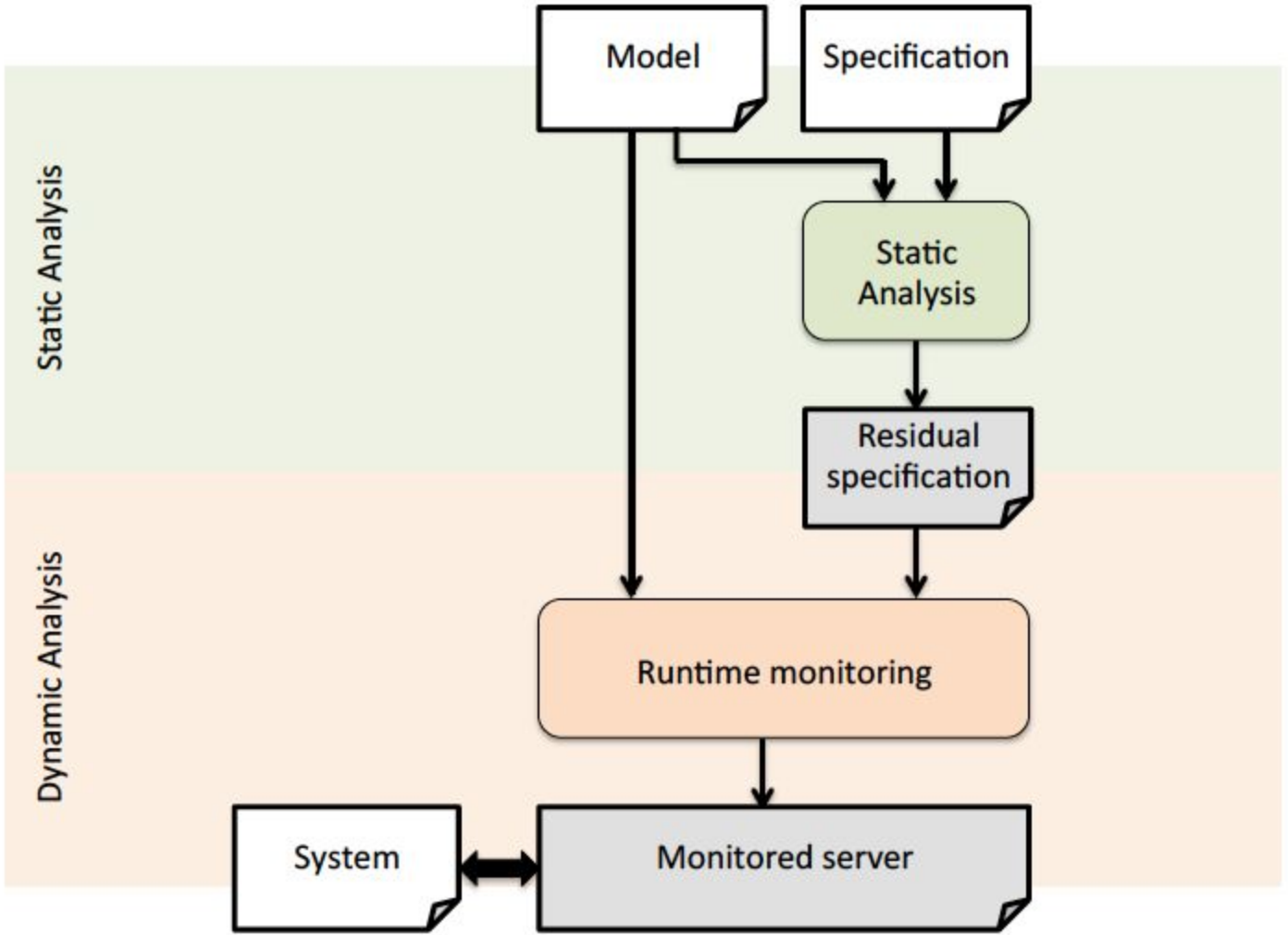
UK e-money regulations state that funds on financial instruments should be redeemable at par value.

1. Is correct value given to the user

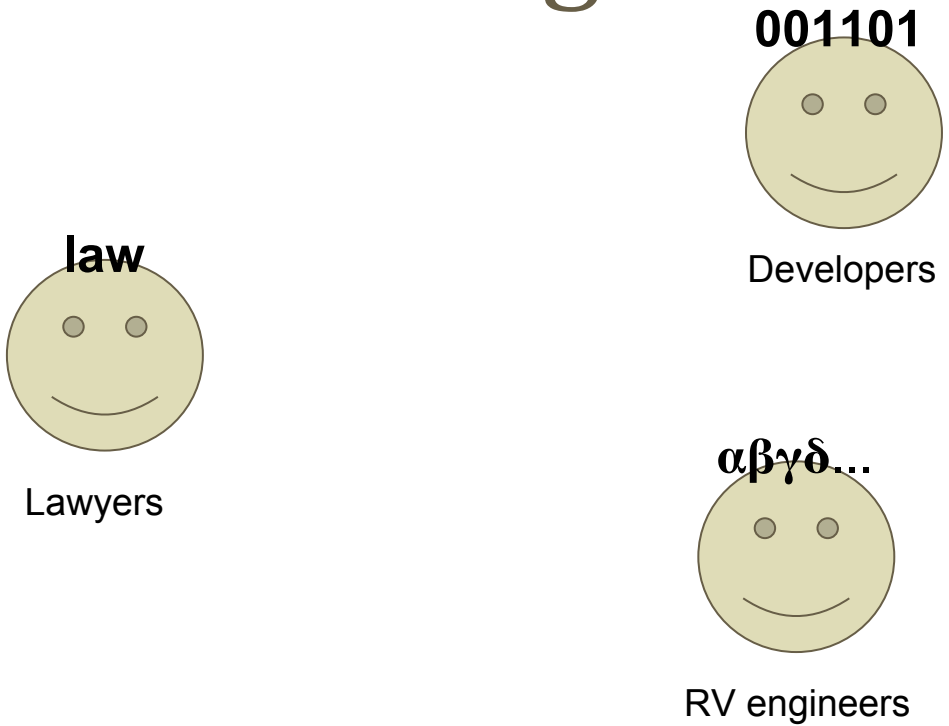
3. How much money is allowed on instruments?


System
Specification
Static Analysis
Static Analysis
Residual specification
Dynamic Analysis
Runtime monitoring
Monitored system

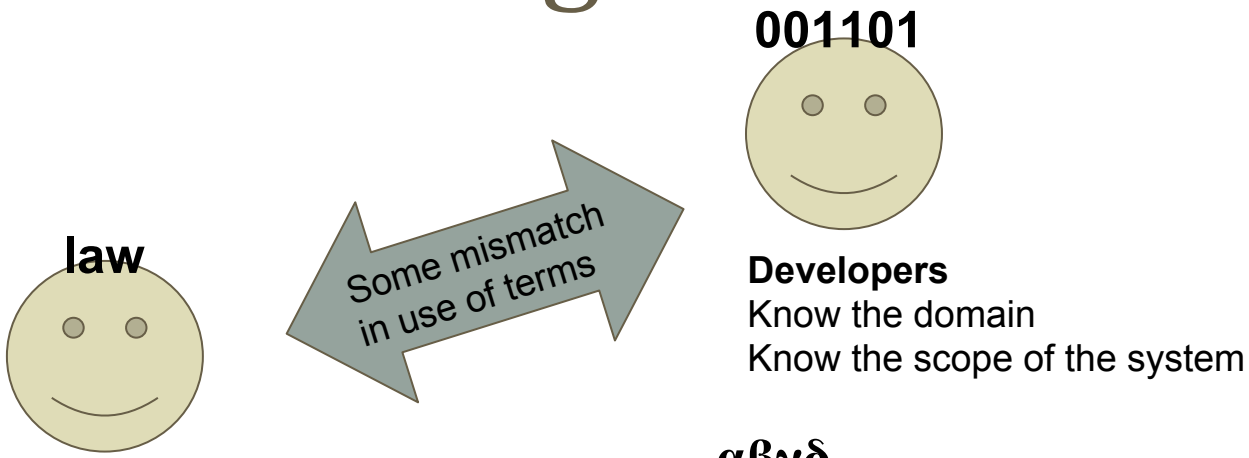

Static Analysis
Dynamic Analysis
Model
Specification
Static Analysis
Residual specification
System
Runtime monitoring
Monitored system


Model
Specification
Static Analysis
Static Analysis
Residual specification
Dynamic Analysis
Runtime monitoring
System
Monitored server

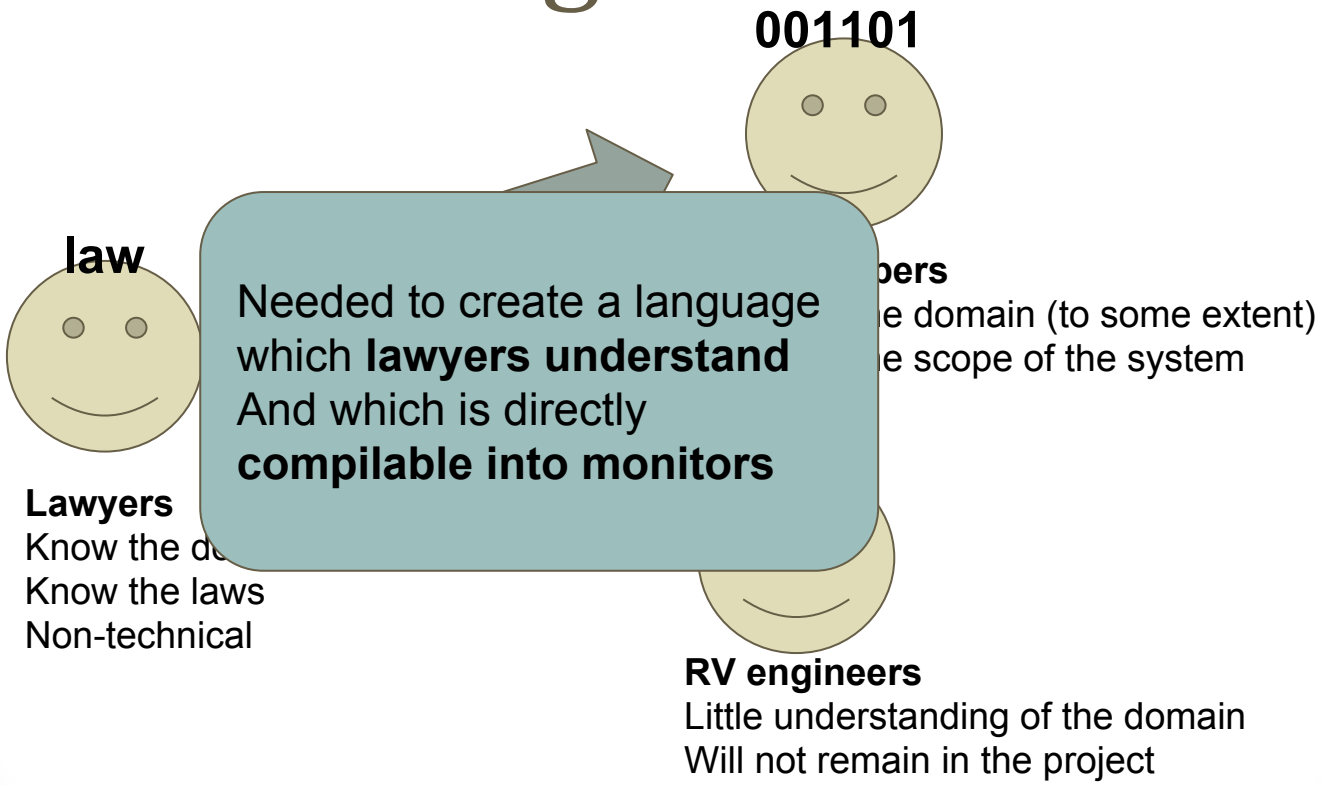
Understanding the domain



Understanding the domain



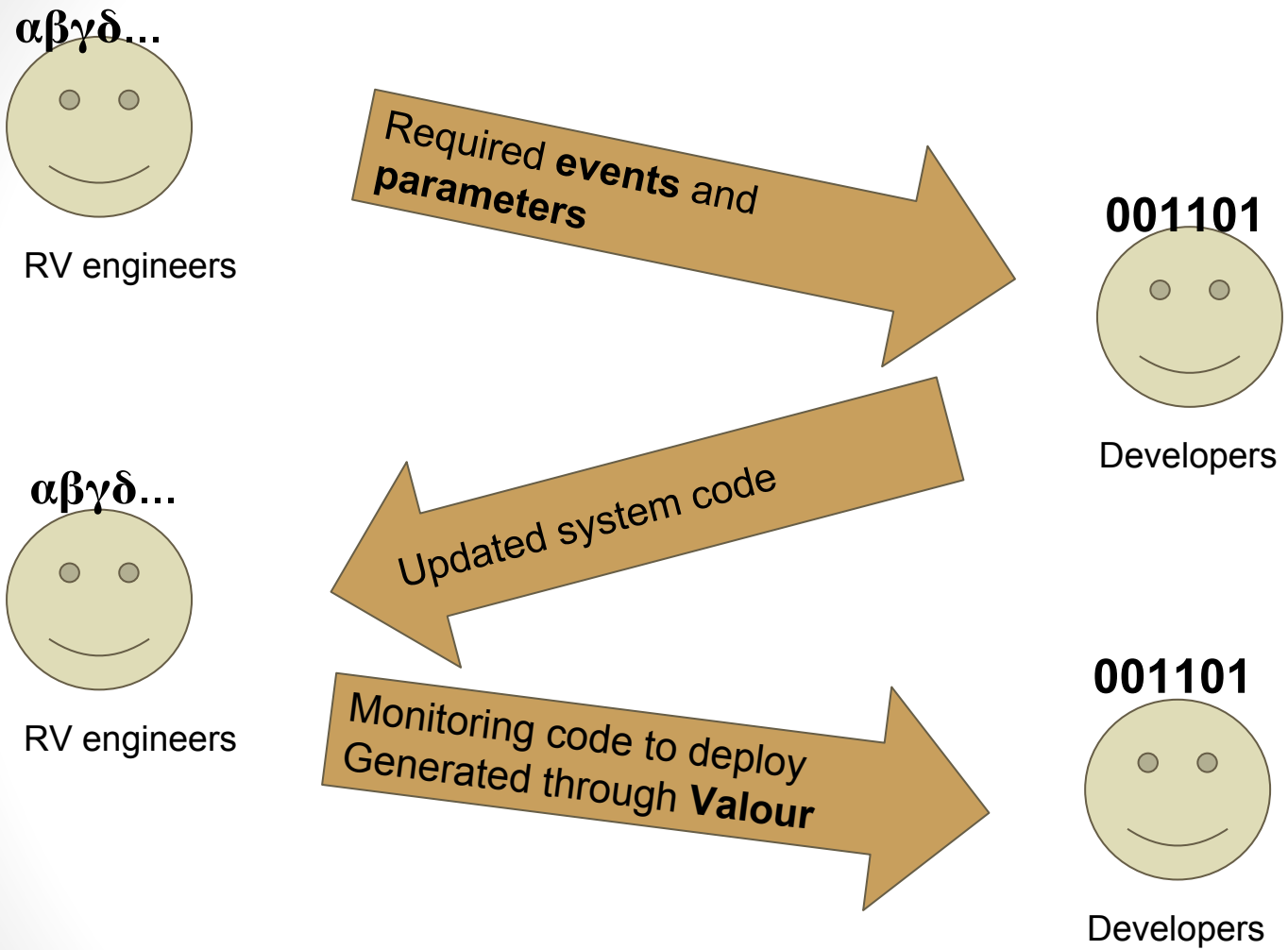
Understanding the domain



Example - Controlled Natural Language

LAW: An electronic money issuer must not award (a) interest in respect of the holding of electronic money; or (b) any other benefit related to the length of time during which an electronic money holder holds electronic money.

CNL: For each programme p, and instrument i, where p is regulated in the UK, and i deals with e-money, then i does not give time-based rewards.

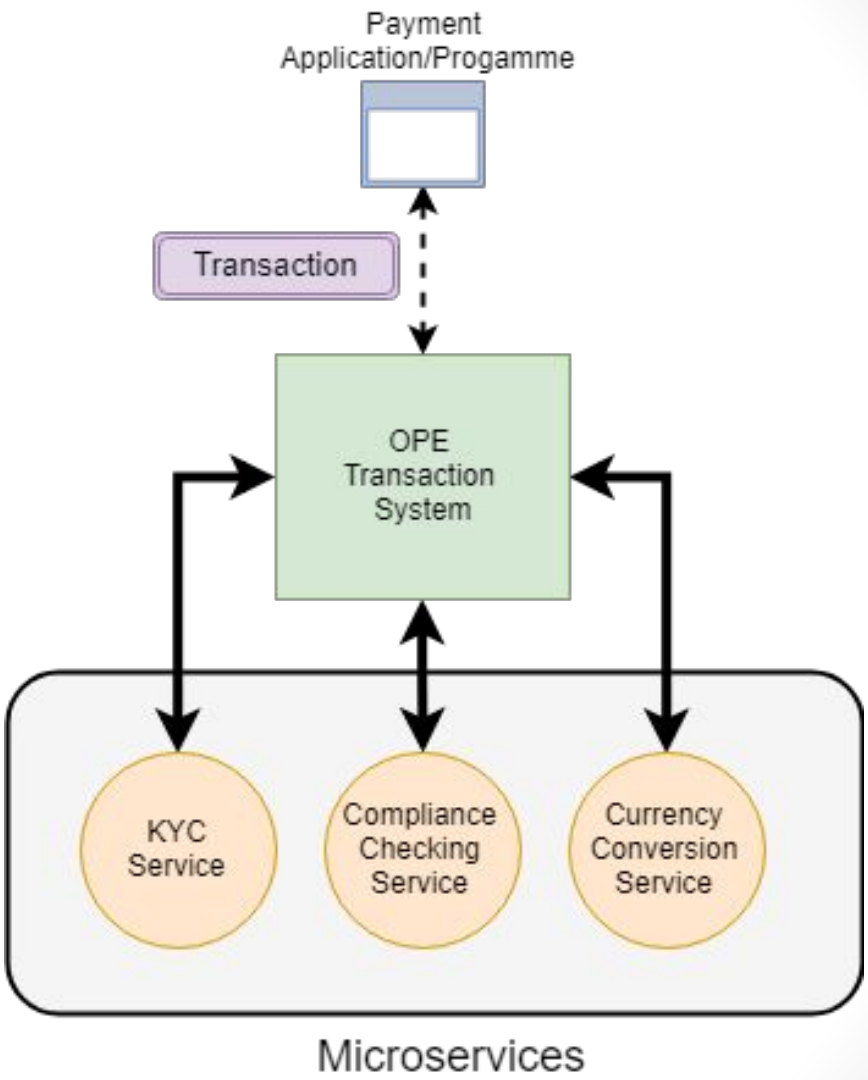

αβγδ…
RV engineers
Required **events** and **parameters**
001101
Developers
αβγδ…
RV engineers
Updated system code
Monitoring code to deploy
Generated through **Valour**
001101
Developers

Now for Valour..

Now for Valour..

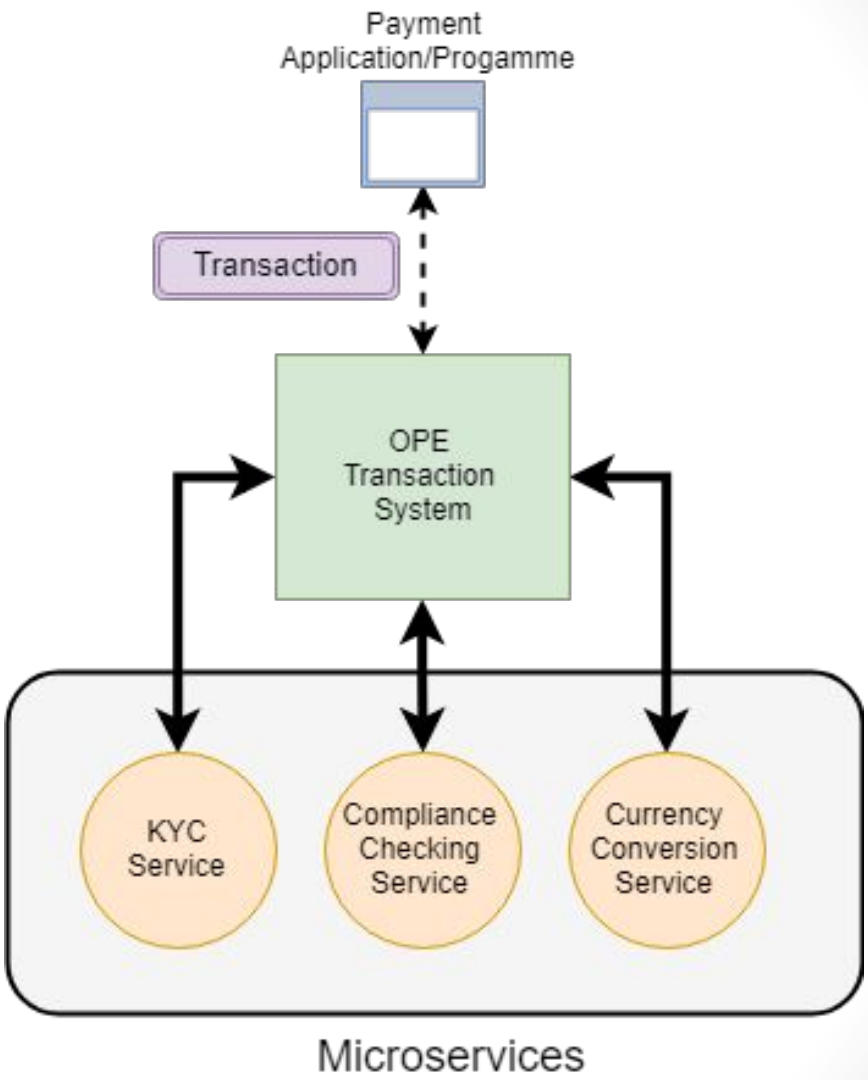
- We designed *Valour* in this project to perform the RV part of the compliance checking.
- But it was developed in a domain- and technology-agnostic manner, allowing for different plug-ins from different systems and with rules as guarded command triples.

Integration with OPE

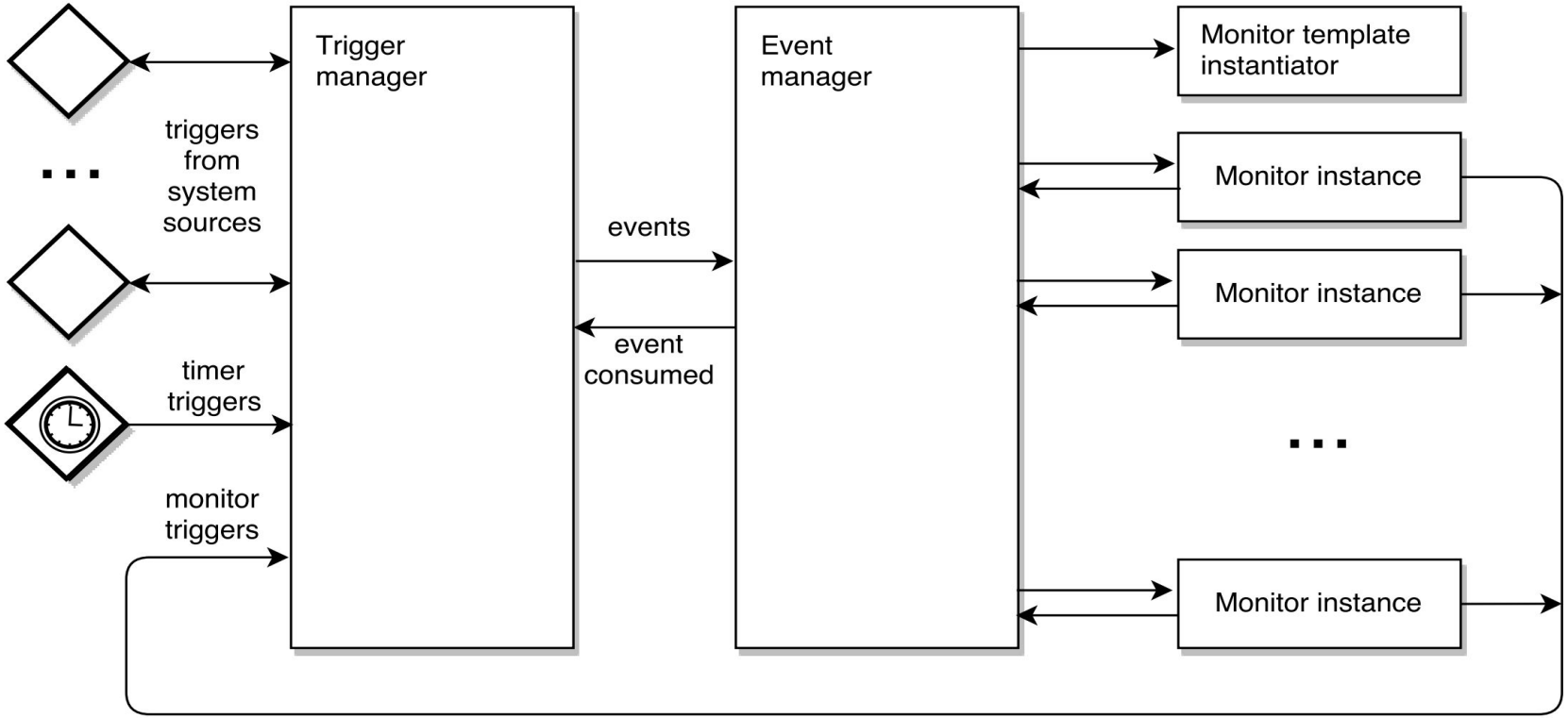


Integration with OPE

- *Spring* framework to organise monitoring system
- *Jooq* to log events and verdicts, as supporting documentation of legal compliance



Architecture



A little Valour Notation

- **Categories**: allowing us to define a certain typestate, identified by an index, e.g:

    category *TRANSACTION* ***indexed by*** *Long*

A little Valour Notation

- **Events**: Activated upon a trigger, with some associated data, satisfying some condition (see ***when***), and possibly belonging to a certain category e.g.:

```
event startOfPayment(ComplianceTriggerData d) = {
    external trigger StartOfTransaction generates ComplianceData d
        when {{d.payment}}
        belonging to TRANSACTION with index {d.txId}
}
```

A little Valour Notation

- **Conditions**: Re-usable condition functions:

```
condition isProgrammeRegulatedInTheUK(String programmeCountry) = {
    "UK".equals(programmeCountry)
}
```

A little Valour Notation

- **Guarded Command Monitors:** activated upon an event, dependent on a condition holding, and activates some action, e.g.:

```
startOfNonExemptTransaction(d)
    | {d.srcCreditCard &&
        #isProgrammeRegulatedInTheUK(d.programme.country)}
    -> { if(d.fee > (d.txAmountInEuro * 0.003)) {
            error()
        }
        else {
            ok()
        }
    }
```

A little Valour Notation

- **Replicated State Monitors:** Guarded command monitors with state replicated for each object of a category.

```
replicate {
    LocalDateTime startTime = {null}
    LocalDateTime endTime = {null}
}
foreach TRANSACTION t {
    <rules>
}
```

Demo

- OPE is still in production (and Valour too).
- We can however mock OPE transactions using Mockito.

- Let’s see some examples!

Overheads

- The memory-intensity of rules and the amount of rules affect overheads
- But the overheads here do not come only from RV.

Overheads

- The memory-intensity of rules and the amount of rules affect overheads
- But the overheads here do not come only from RV.

- The architecture chosen for the OPE, i.e. a microservice architecture, will create some communication overheads to send the transaction and by monitoring to send back the result.

Overhead Management

- Our solution**: We will only have synchronicity for a predefined period of time** (in milliseconds, to be decided). After this, **compliance checking continues asynchronously.**

- Violations found asynchronously then block future events, until handled by OPE administrator.

Event (Mis)Ordering

- The OPE and our compliance checking may be on different servers, and thus events may not arrive in the same order they were sent,

Event (Mis)Ordering

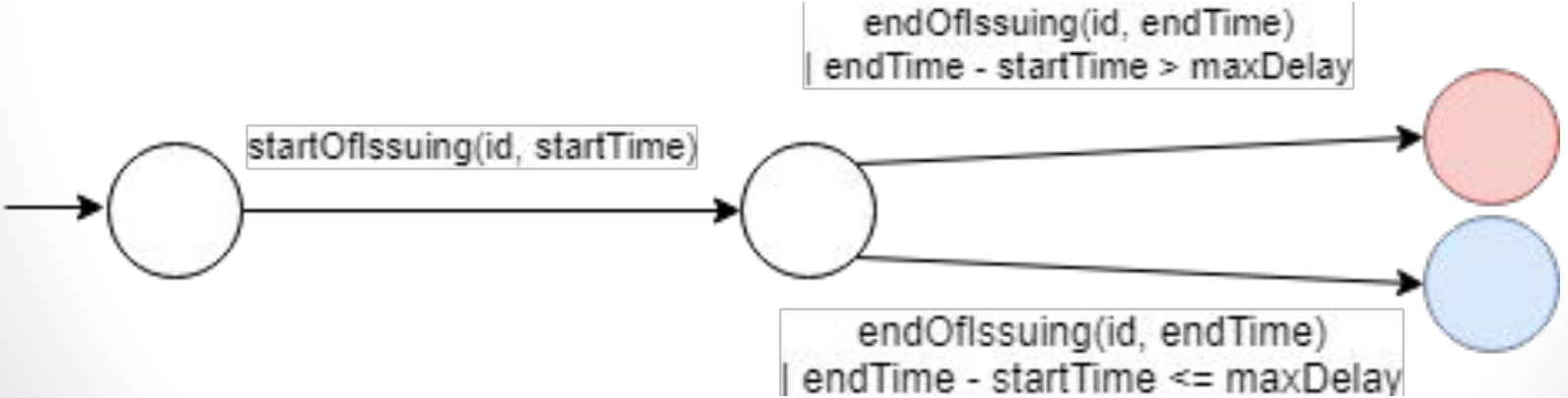
- The OPE and our compliance checking may be on different servers, and thus events may not arrive in the same order they were sent,
- e.g. an end of transaction event may arrive before the start of transaction event.

Event (Mis)Ordering Example

- **For each** *instrument i* **and** *programme p*, **where** *i is an instrument of p* **and** *p is regulated in the UK*, **then** e-*money in i is issued is less than the maximum issuing delay*.

Event (Mis)Ordering Example

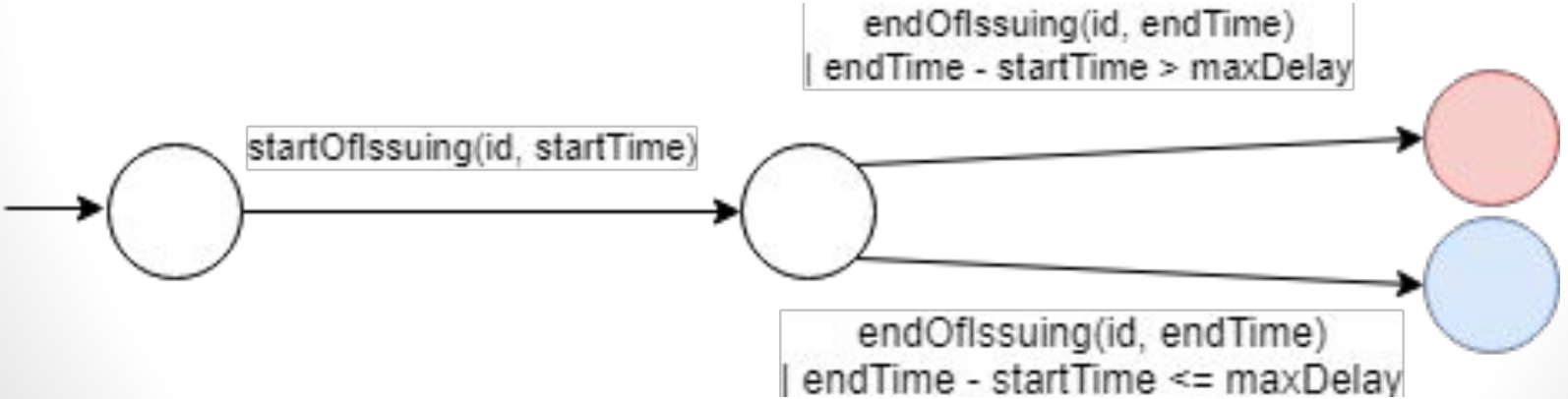
- **For each** *instrument i* **and** *programme p*, **where** *i is an instrument of p* **and** *p is regulated in the UK*, **then** e-*money in i is issued in less than the maximum issuing delay*.



Event (Mis)Ordering Example

Traces:

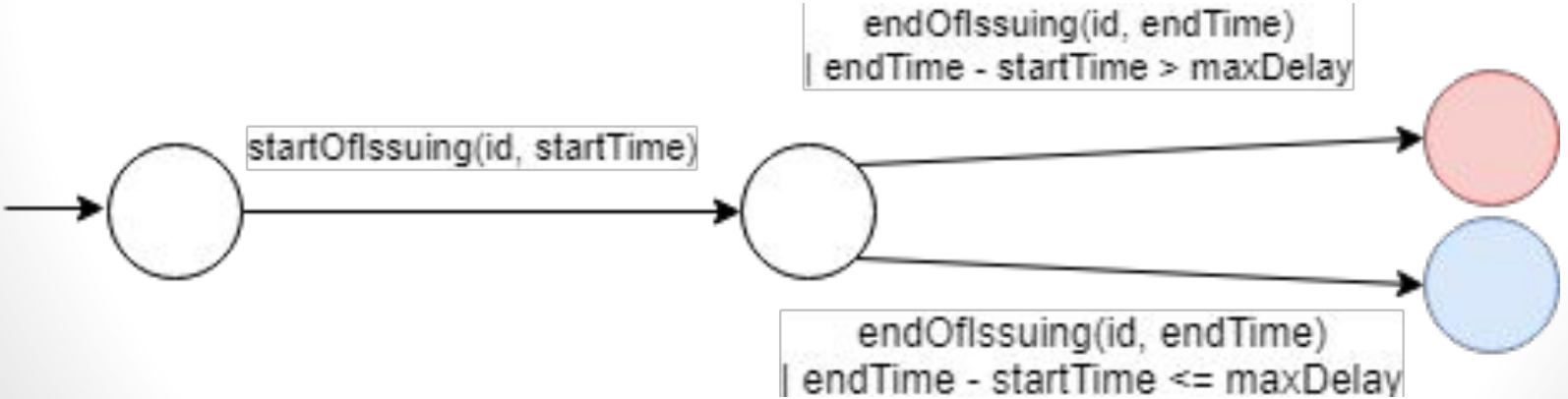
- **startOfIssuing(1, 6:00) ; endOfIssuing(1, 6:05)** *--> Violation*



Event (Mis)Ordering Example

Traces:

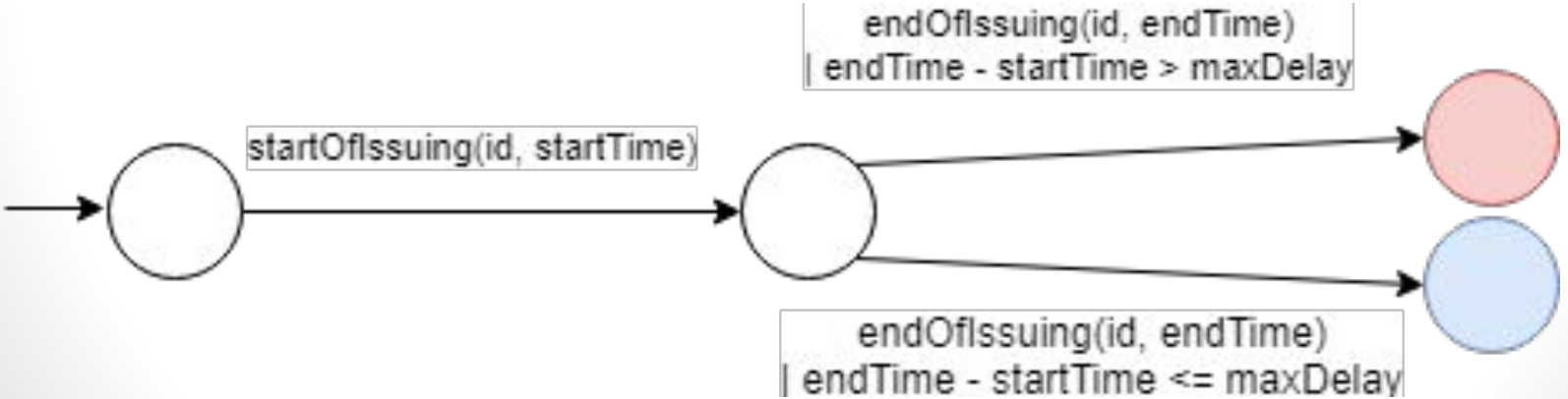
- **startOfIssuing(1, 6:00) ; endOfIssuing(1, 6:05)** *--> Violation*
- **endOfIssuing(1, 6:05) ; startOfIssuing(1, 6:00)** *--> Satisfaction*



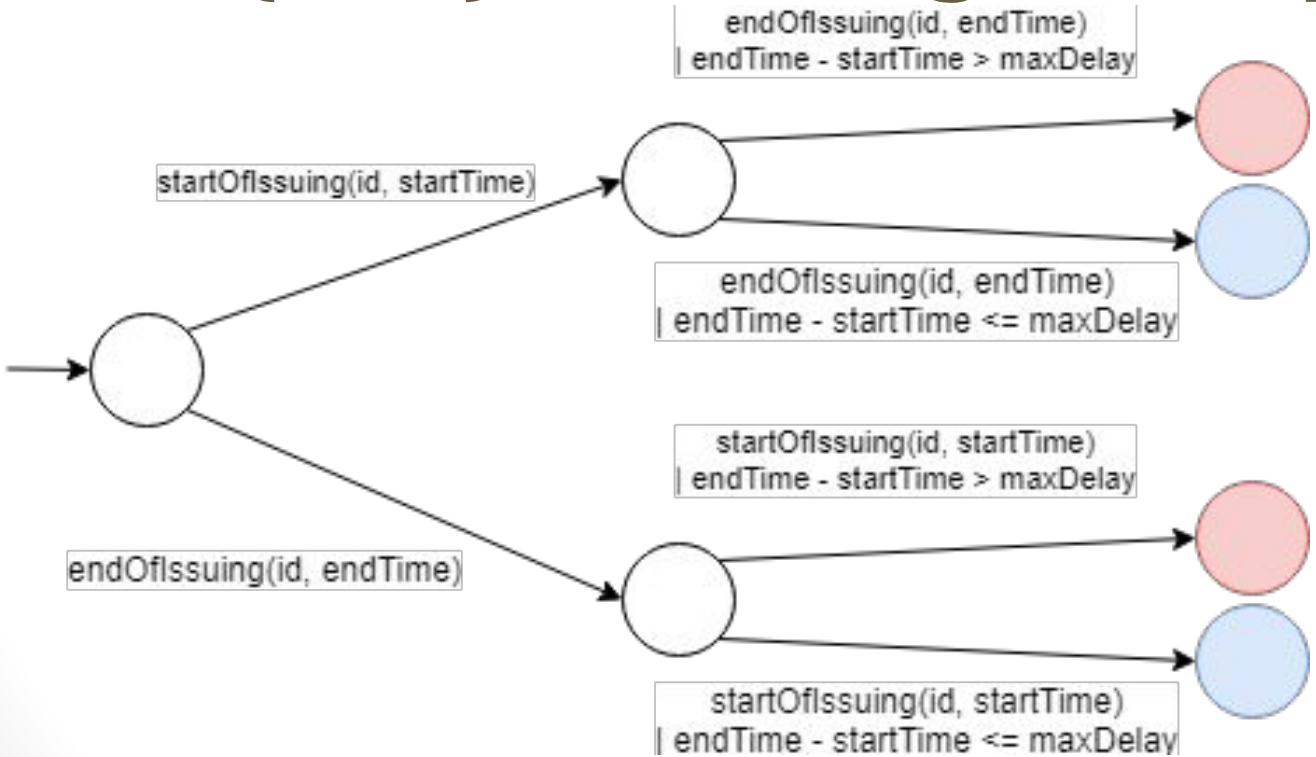
Event (Mis)Ordering Example

Traces:

- **startOfIssuing(1, 6:00) ; endOfIssuing(1, 6:05)** *--> Violation*
- **endOfIssuing(1, 6:05) ; startOfIssuing(1, 6:00)** *--> Satisfaction*



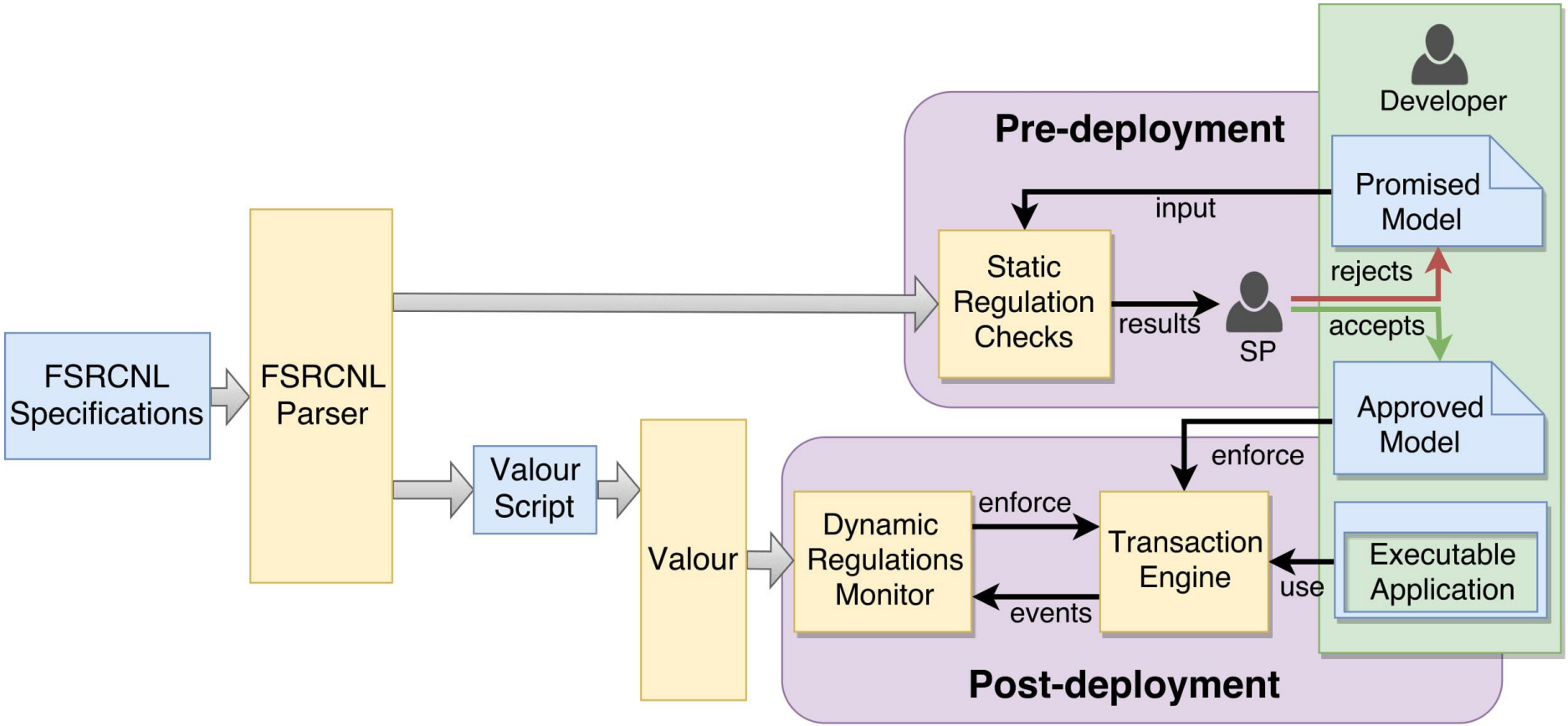
Event (Mis)Ordering Example



Conclusions

- We have motivated and presented an RV tool, Valour, used in the context of technology-agnostic compliance checking.
  Latest version on: https://github.com/shaunazzopardi/valour
- What we learnt:
  - Payments industry cares more about performance than compliance (varies with risk);
  - Natural Languages > Logical languages;
  - Simple rules/specifications;
  - Assumption that events will always arrive in the correct order is not always true.

Appendix


FSRCNL Specifications
FSRCNL Parser
Valour Script
Valour
Pre-deployment
Static Regulation Checks
input
results
SP
Developer
Promised Model
rejects
accepts
Approved Model
enforce
Dynamic Regulations Monitor
enforce
events
Transaction Engine
use
Executable Application
Post-deployment

A Little Valour Notation (1)

```
state {
  Integer userCount = { 0 }
} in {
  userLogin(u) | { u.isGoldUser() } -> { userCount++; }
  userLogin(u) | { u.isNormalUser() && userCount < 100 } -> { userCount++; }
  userLogin(u) | { u.isNormalUser() && userCount >= 100 }
    -> { generateAlarm("User saturation level exceeded"); }
  ...
}
```

A Little Valour Notation (2)

```
replicate {
  BigDecimal ownTransactionsTotal = { 0 }
} foreach USER u {
  transfer(src, dst, value)
    |  { dst.owner() == u }
    -> { ownTransactionsTotal += value; }
}
```