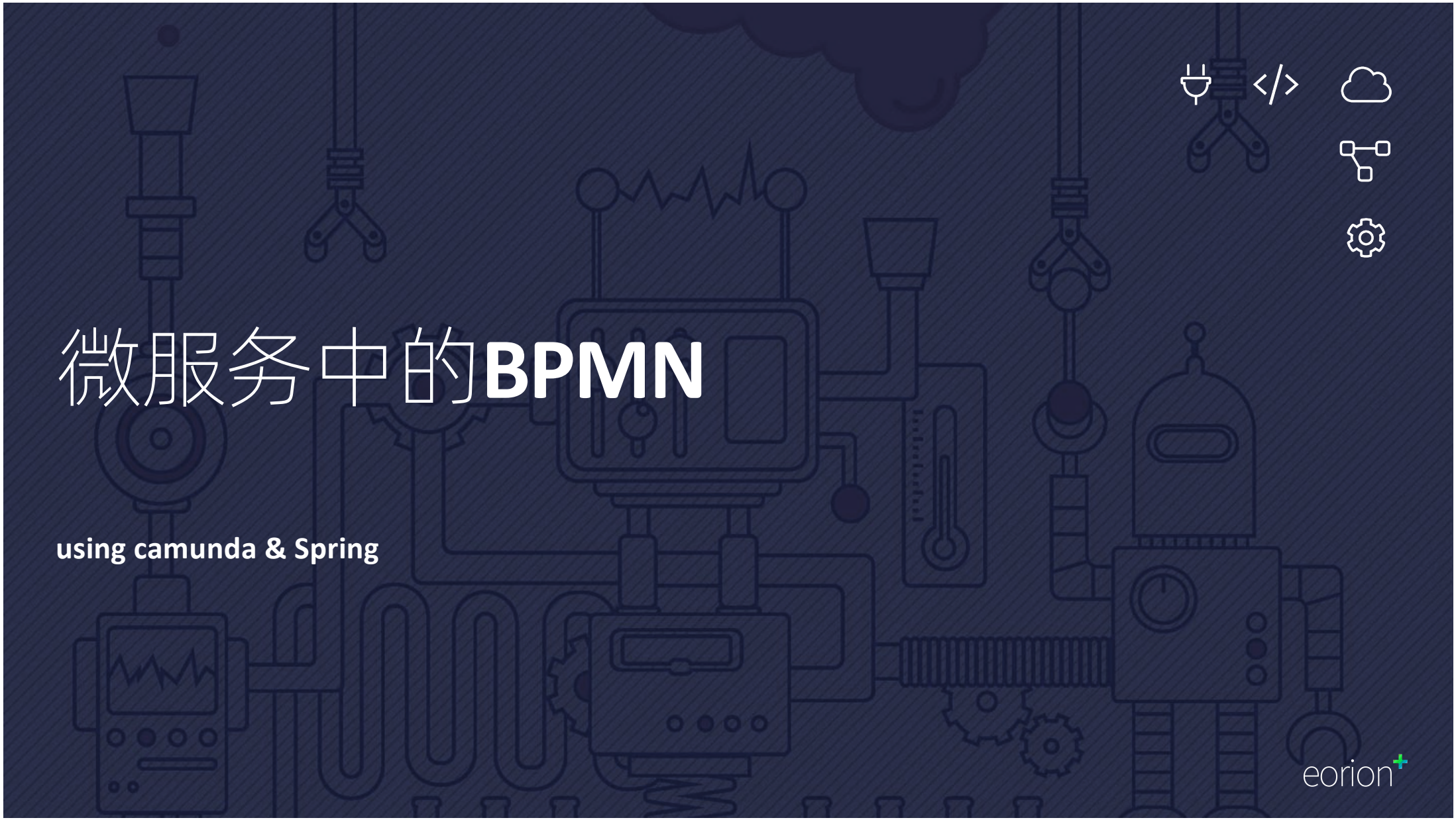

微服务中的BPMN
using camunda & Spring
eorion

前言

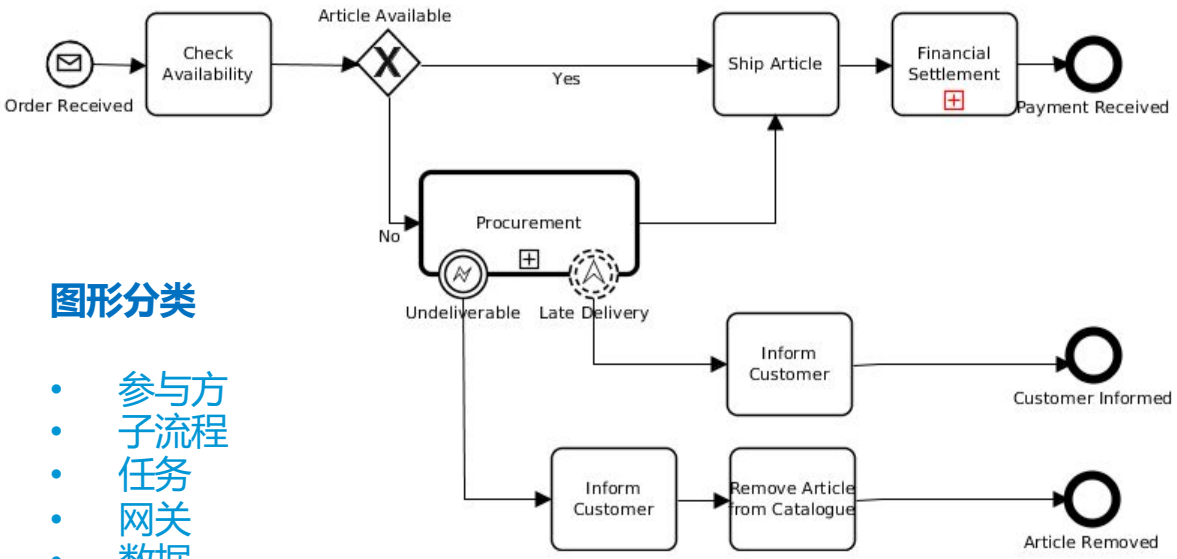
关于BPMN

BPMN

BPMN (Business Process Model and Notation) 已成为事实上的业务流程图的行业标准。

它旨在为业务负责方提供一个可以直接使用的流程设计、管理和实现的图形化规范同时这一规范又能够精准的转换为IT流程组件，让IT应用准确反应业务需求设计。

BPMN具有易于理解和使用的类似于流程图的表示法。**它与任何特定的实现环境无关**。



图形分类

- 参与方
- 子流程
- 任务
- 网关
- 数据

详细说明请访问 **http://www.bpmn.org/** 或 **https://camunda.com/bpmn/**

基本概念

服务
SOA
微服务

有关于服务的名词

服务 Service

一个软件程序，它通过已发布的技术接口暴露其功能，而这样的接口被称为“**服务契约**”。

服务能力 Service Capability

服务契约可以被分解为一组服务能力，而每种能力表示一项由服务提供给其他软件程序的功能。

服务消费者 Service Consumer

在运行时，访问或者调用——更明确而言，向服务契约所表达的服务能力发送信息——服务的软件程序。

服务组合 Service Composition

一组完成某一特定任务的服务的集合体。

有关于服务的名词

面向服务 Service-oriented, SO

一种设计范式，用于创建单独成型的解决方案的逻辑单元，并通过重复利用这些逻辑单元，实现具体的战略目标与面向服务计算的相关收益。

面向服务架构 Service-oriented Architecture, SOA

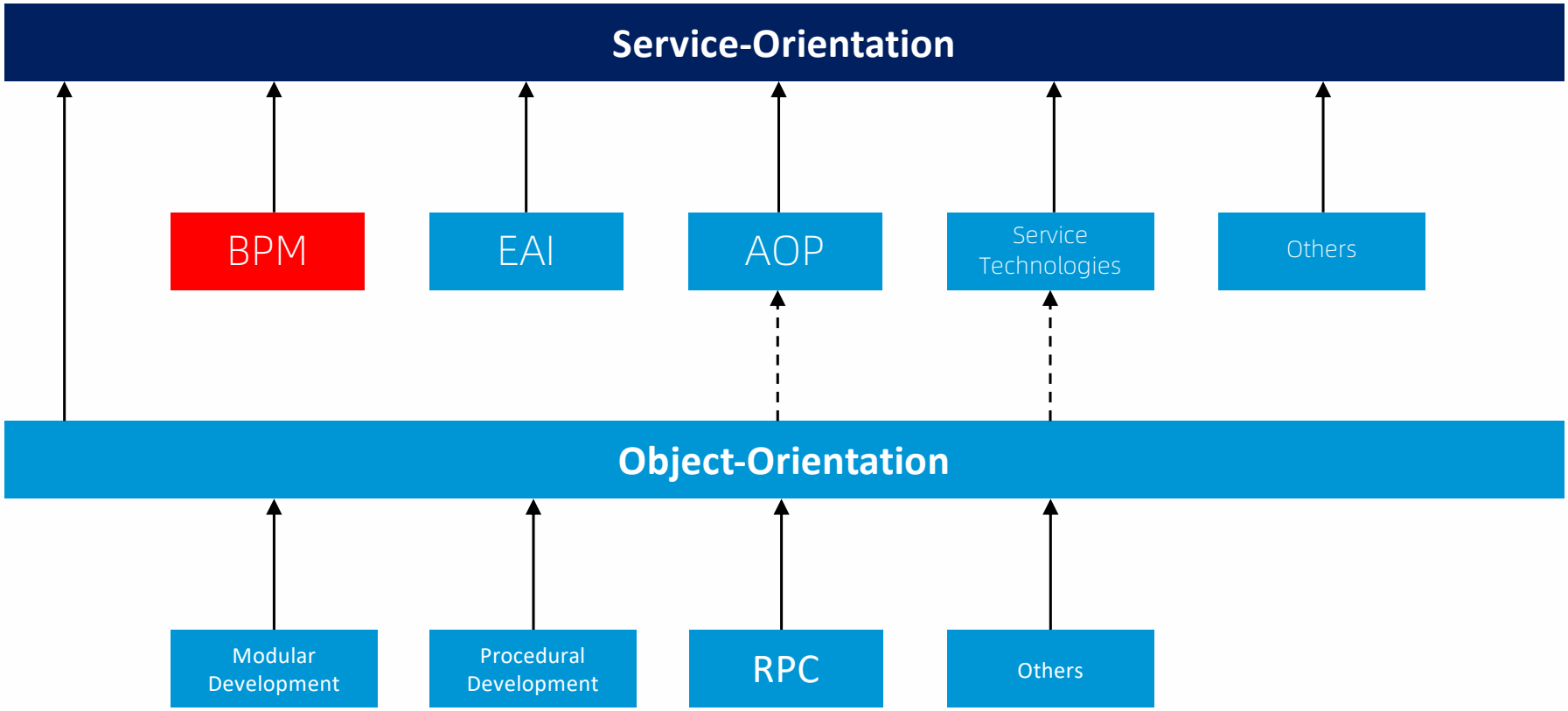
一种技术架构模型，一种用于具有独特特征的面向服务的解决方案。

微服务 Microservice

一种面向服务架构的特定方案/具体方法，表现为一些协同工作的细粒度而高度自治的服务。

SO vs OO

SO的诞生



集成

微服务中最重要的技术

eorion+

为什么这么说？

微服务的定义及特性

- 自治性
- 服务消费者不可知性
- 技术异构性

使用微服务的原因

- 提高服务可复用性
- 可扩展性
- 可组合性
- 可替代性

理想的集成技术

避免破坏性修改

- 避免修改某一服务而导致该服务的消费者也随之发生改变

保证服务契约的技术无关性

- IT业界唯一不变的就是变化；因为技术变化而修改服务是本末倒置

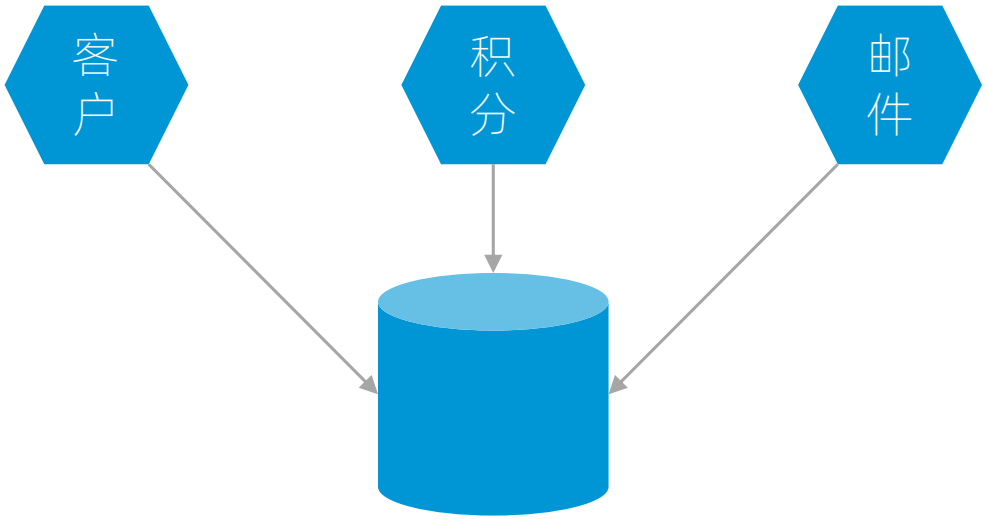
易于服务消费者使用

- 使用困难导致消费者减少，产生“孤儿”服务

隐藏内部实现细节

- 内部实现的修改不应该影响服务消费者的使用

共享数据库：一种最简单也最流行的集成方式



问题

- 外部系统可以查看内部实现细节
- 消费者与特定的技术绑定
- 破坏内聚性

服务间的通信方式

服务消费者应该如何与服务沟通？

同步

基于请求-响应模型

- 请求方发起请求，并在获得结果前阻塞自己
- 响应方处理请求并反馈结果

生活实例：打电话

- 打电话的特点？
- 使用场景？
- 本地电话 vs 越洋电话

技术实例

- RESTful API
- RPC
  - ✓ SOAP
  - ✓ Thrift
  - ✓ Java RMI

异步

基于事件模型

- 请求方发出一个消息，并亲自或由第三方保证该消息成功发出；
- 接受消息方处理所收到的消息，并亲自或由第三方保证消息在处理完成前不会消失

生活实例：信件

- 写信的特点？
- 使用场景？
- 本埠信件 vs 国际信件

技术实例

- JMS
- MQTT
- AMQT

编排(Orchestration)与协同(Choreography)

如何驱动服务完成特定的任务?

编排

依赖某一个中心服务驱动整个流程

- 适用请求-响应模型
- 迅速知晓结果

生活实例：乐队指挥

- 指挥的特点?
- 使用场景?

问题

- 中心服务承担太多职责
- 极少量的“上帝”服务及大量“贫血”服务

协同

使用异步方式触发一个事件

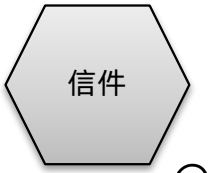
- 基于事件模型
- 显著消除服务间耦合

生活实例：芭蕾舞者

- 舞者的特点?
- 使用场景?

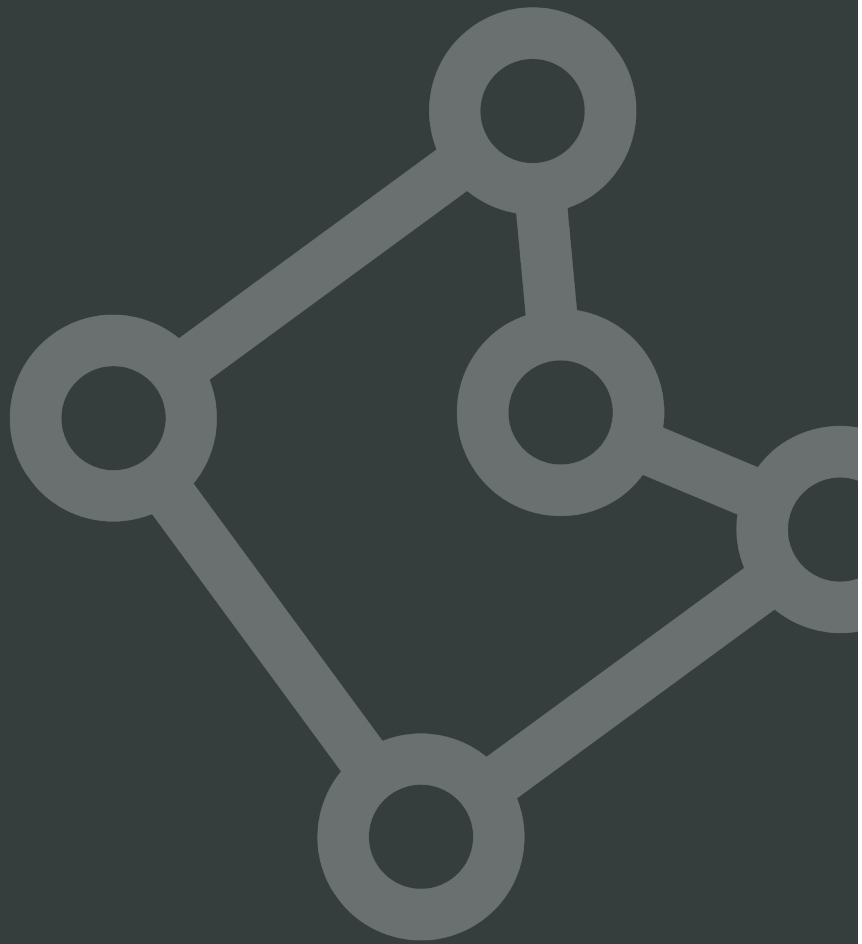
问题

- 额外的监控以保证业务的正确进行
- 异步架构引入的额外复杂性



eorion+ | powered by camunda

微服务中的BPMN



eorion+

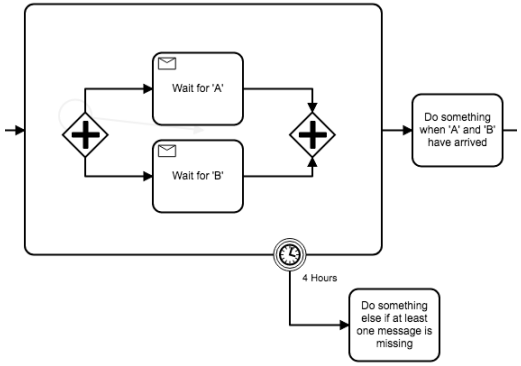
使用场景

- **驱动服务**
  - ✓ 端到端的业务一般涉及众多微服务，因此必须组织他们之间的交互。
- **业务视角**
  - ✓ 完整的业务分散于多个微服务中，有必要了解整个业务的运行情况。
- **跨服务事务**
  - ✓ 保证跨越多个服务的事务始终有效。
- **运维监控**
  - ✓ 了解业务关键指标的情况，并及时采取相应的措施。

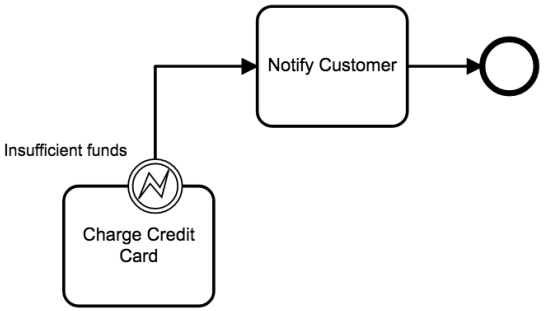
驱动服务

BPMN的核心是序列流，它定义了工作流中各个步骤的执行顺序。

除任务之外，BPMN还包括网关（控制流程）和事件（响应及通知）。

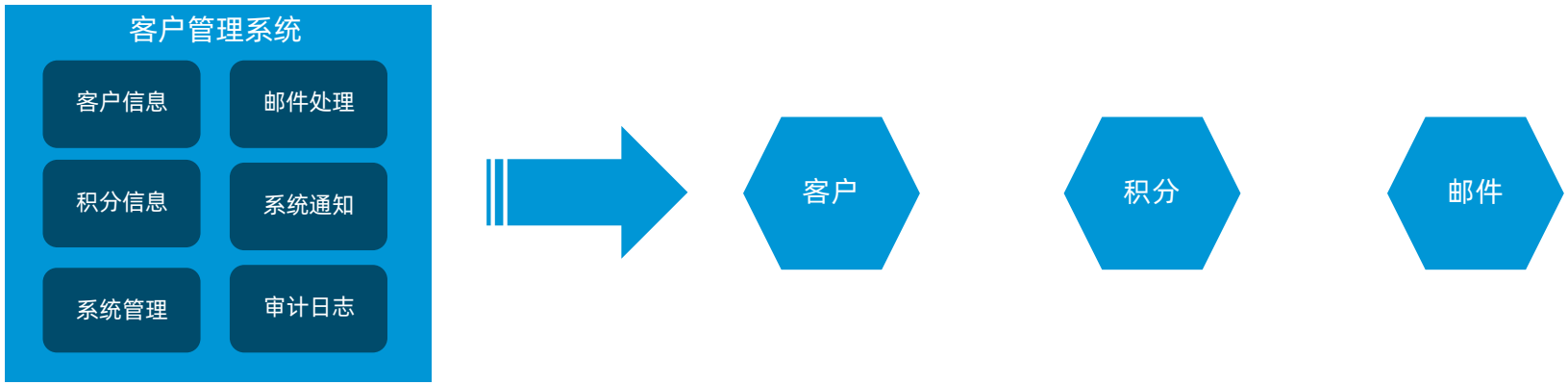


更重要的，BPMN提供异常处理

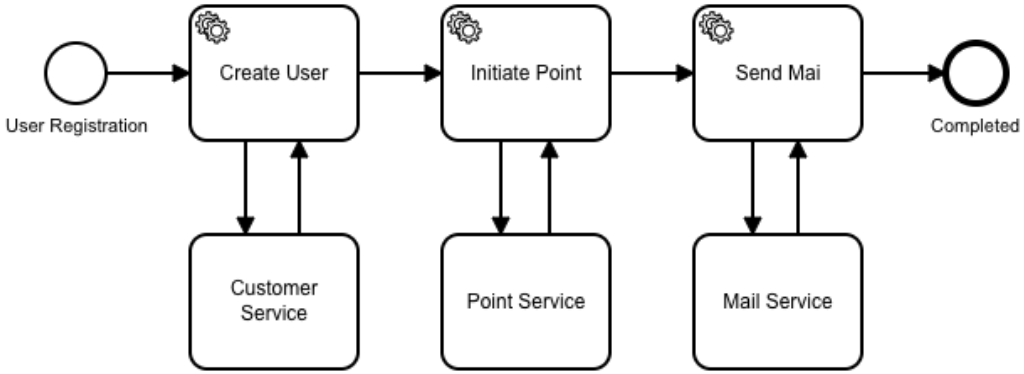


业务视角

微服务是对单体系统(Monolithic System)分解的结果，将使原来处于整体中的业务流程分散在各微服务中。



可以使用BPMN驱动微服务从而实现业务流程。

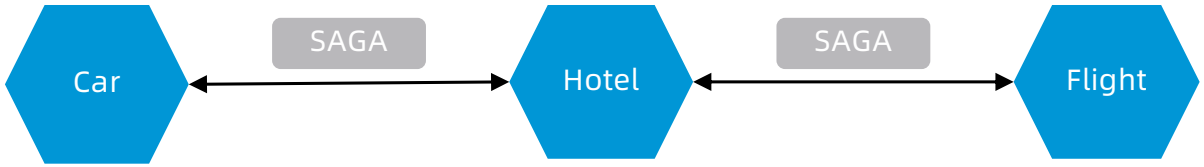


跨服务事务

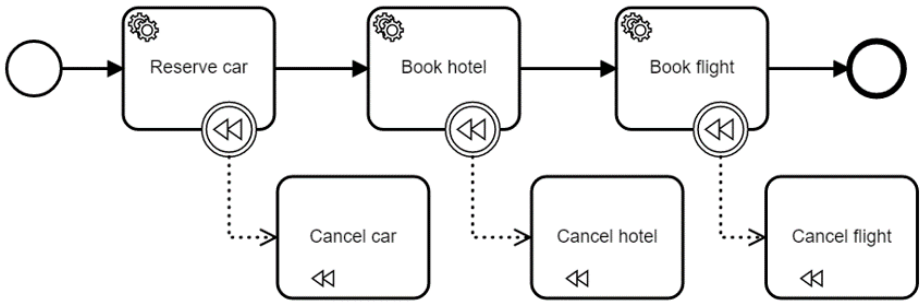
基于两阶段提交的X/Open XA规范 —— 事务可跨越应用而保持有效。

- ✓ 存在一个协调者节点及其他的参与者节点，所有节点间可以进行通信；
- ✓ 所有节点采用保持在可靠存储设备上的预写式日志，且节点损坏不会导致日志丢失；
- ✓ 所有节点不会永久损坏，且即使损坏后仍然可以恢复至之前状态。

SAGA模式——通过消息传递事务



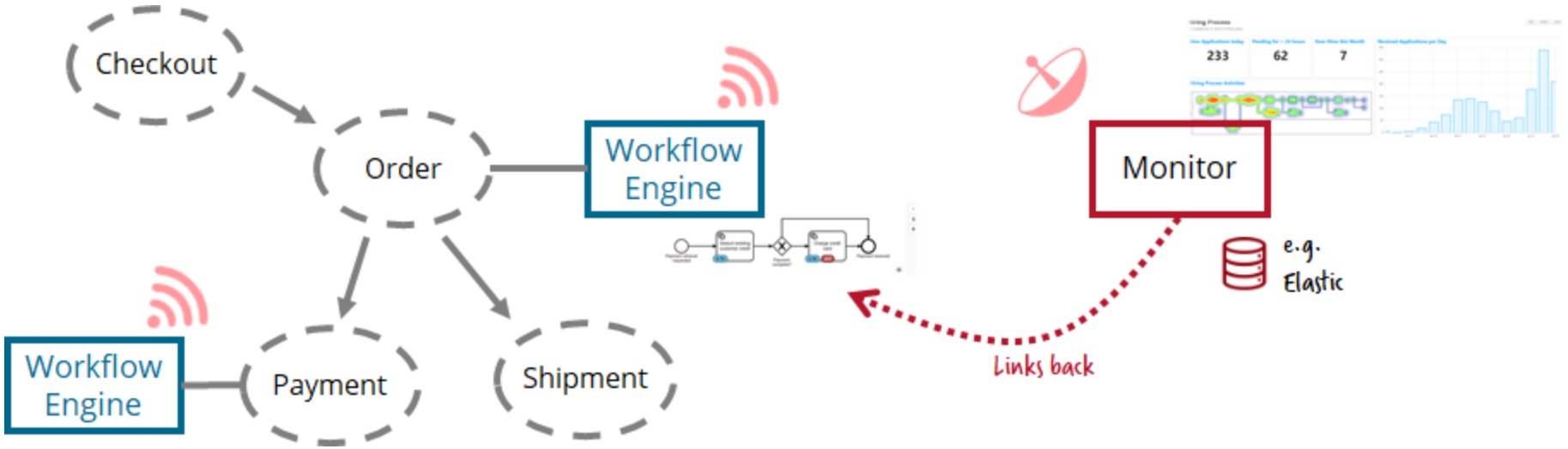
SAGE模式——BPMN补偿事件



运维监控

使用分散的BPMN引擎处理具体的业务；使用中心BPMN引擎监控更高级别的数据。

- ✓ 一个跨越微服务边界的端到端流的概述
- ✓ 监控分散引擎中的最重要事件（实例启动、达到里程碑、实例失败或结束等）



技术栈

eorion+

Spring - 事实上的Java工业标准

Spring Framework

- Java Bean 容器

Spring boot

- 自包含的Java应用

Spring cloud

- 原生云服务

Camunda – 高性能BPMN引擎及应用

Camunda流程引擎

- 流程引擎公开API
- BPMN 2.0 核心引擎
- Job执行器
- 持久层

管理及操作平台

- Admin
- Cockpit
- Tasklist

RESTful API接口

- 覆盖几乎所有流程引擎公开API

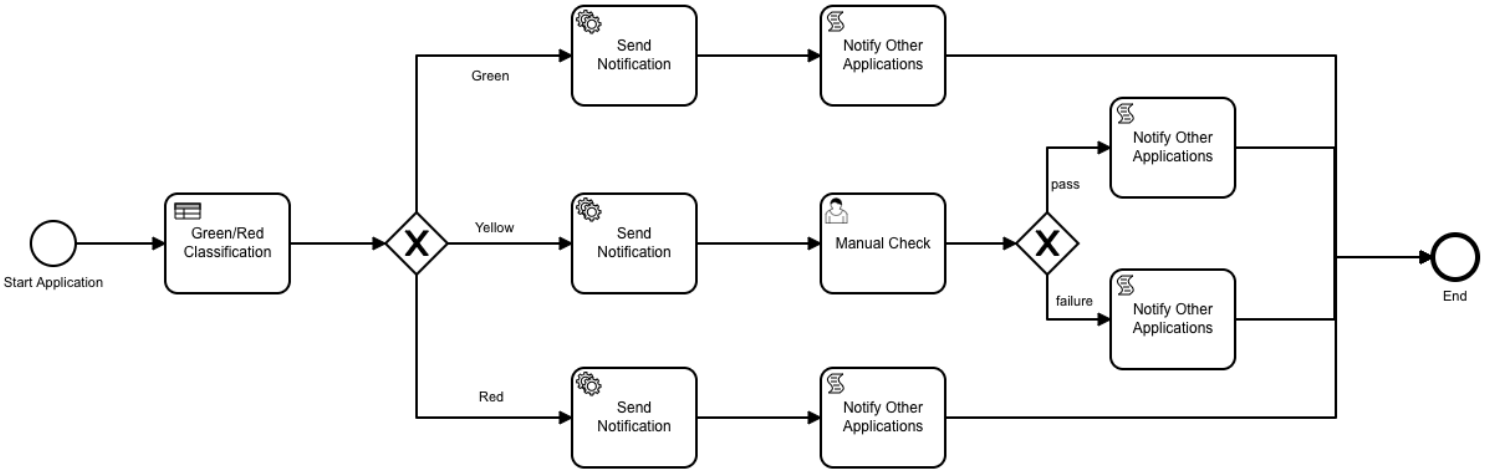
Case Study

eorion

案例学习1

信用卡申请流程

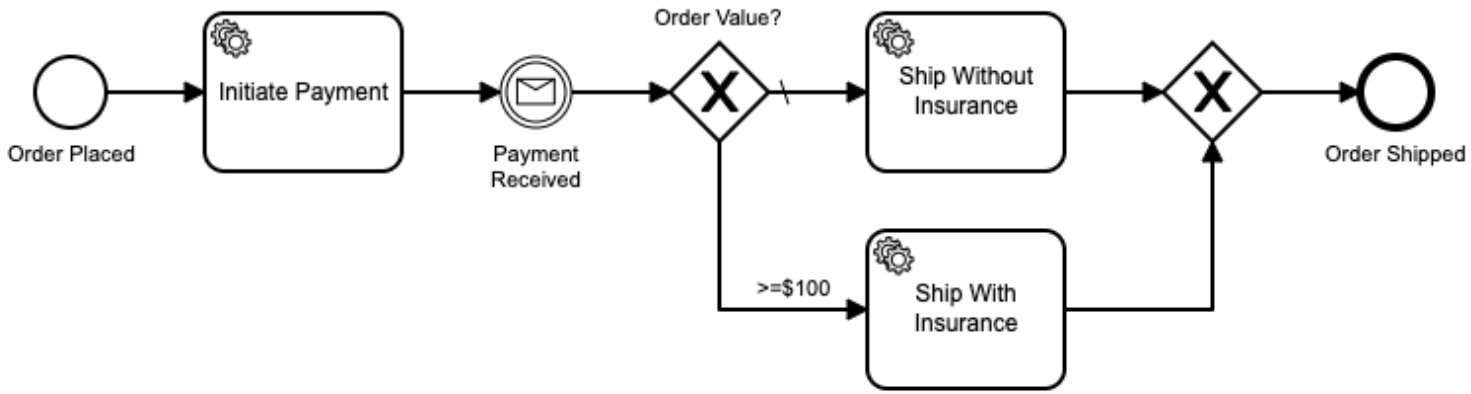
- ✓多终端支持；
- ✓基于DMN的业务逻辑分离；
- ✓基于同步的RESTful API集成；
- ✓基于Spring Boot构建的微服务。



案例学习2

在线销售流程

- ✓ 使用BPMN协同微服务；
- ✓ 展现微服务所表现的业务流程；
- ✓ 基于异步的消息集成；
- ✓ 业务抽象能力。



附加信息

Process Mining

www.camunda.com

tonnychen@eorionsolution.com

eorion

Thanks

www.eorionsolution.com

eorion