插件开发核心是善于在业务中抽取出不变和变化点，再将变化进行封装。

目的：物联网平台转换业务语言，支撑业务平台能力。简单来说简化它里面繁琐的配置，拓展它支撑业务的能力。所以就有了非侵入式业务插件的想法出现，物联网平台插件解决的问题包括但不限于：

- 增强物联网平台的交互性
- 业务的处理能力（时序数据的处理，查询，统计）
    - 数据处理
    - 数据查询
    - 数据多维度统计
- 拓展物联网平台能力

1 思路

1.1 数据交互

在插件的实现中，首先要解决的的是与宿主系统进行通信的问题，解决的办法是通过接口调用，而不是RPC进行调用，原因有二：

1. 绝大多数的系统都是提供接口而实现数据交互
2. RPC调用虽然性能较好，但是对宿主系统改动量很大（如果不是RPC方法的话），这样就不符合非侵入式插件的目的。

采用接口调用最大的问题就是"认证"，“认证”即通过给定系统访问的"token"即可以解决，但是宿主系统提供的"token"存在时效性，所以需要宿主系统提供能刷新token的地址。实现OAuth2规范。

除了主动的进行数据交互，对于某些时序数据的处理，更需要被动接收数据的能力，所以必然通过消息队列做"数据暂存"与"择机处理"。

1.2 模型

模型是对宿主系统数据的解释，通过模型的定义，能对宿主系统做到数据查询，处理与多维度统计。

1.3 亟待解决的问题

1.3.1 报警模型

构建**报警模型**，抽象的物联网告警规则数据，与物联网报警进行数据交互的主要入口，它支持的功能是：

1. 业务报警规则设置转换为物联网告警设置
2. 物联网告警设置转换为业务系统语言

1.3.2 时序模型

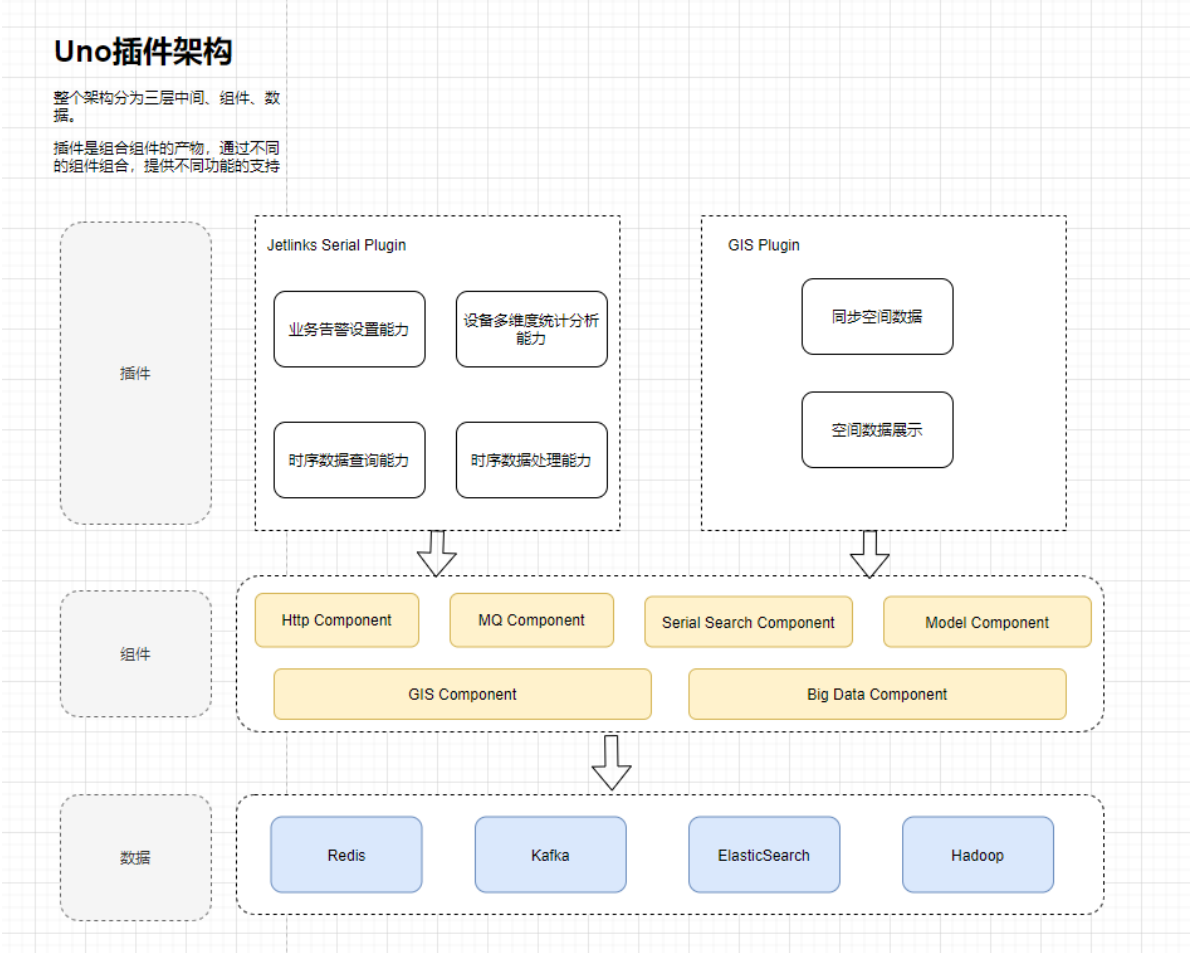
时序模型，同样是物联网设备物模型的抽象，它可以提供很方便的数据查询的入口，以及数据计算的模型定义。所以，它的职责是：

1. 直接与时序数据库进行交互
2. 提供数据计算的模型并转换为物联网规则链

1.3.3 统计模型

多维度统计，需要多个时序模型进行组合。提供复合的时序数据查询与处理的能力，复合查询与处理的结果是是统计数据的入口，在根据设置的统计计算模型得到统计结果。

2 架构



插件的架构分为三层，分别是插件、组件、数据。

- **数据**：提供基本数据操作支持。
- **组件**：提供基本的功能支持。
  - Http Component：拓展了HTTP调用，比如说：根据`OpenApi规范文档`生成具体的远程调用对象。
  - MQ Component：拓展MQ功能，提供简单的发布、订阅操作
  - Serial Search Component：对于时序数据库，提供的简单的api就可以查询到需要的时序数据
  - Model Component：对抽线模型进行定义，提供基本的模型。文件模型
  - ...
- **插件**：组合不同的组件，对他们的提供的功能进行使用与拓展，最终形成可用的业务插件系统

2.1 技术选型

插件采取异步非阻塞编程实现，所以它的技术选型如下：

- `Project Reactor`
- `Spring Webflux`
- `Spring Boot`
- `Redis`

- `Kafka`
- `ElsaticSearch`
- ...

2.2 规划

插件的实现初步分为三个阶段：

1. 基础组件的编写：`Http Component`与`Model Compoent`是实现一个插件必不可少的组件。消耗时间：预计0.5月/人
2. `Jetlinks`时序插件：通过编写完成的基础组件后，下一步就可以进行具体插件的开发，开发过程中将会编写新的组件。消耗时间：预计1月/人
3. 维护与业务插件开发：经过`Jetlinks`插件的开发，不可避免存在一些问题，所以在这个阶段解决设计不合理地方与继续开发新的插件。消耗时间：待规划

2.3 插件平台

当插件越来越多，不同项目需要的插件不同，对某个项目需要的插件打包发布过于繁琐，并且批量管理这些项目中的所有插件。所以插件平台的想法就随之产生。

插件平台管理着所有插件的生命周期，包括创建、版本升级、使用、弃用。并且可以组合这些插件来提供项目需要的功能。除此之外还可以存在插件监控、报警、链路追踪等插件应用性能管理。