1. **系统说明(需求及目标)**

系统需要满足哪些功能项，每项功能的详细描述，设计出UserCase。分为几个子系统或组件，每个子系统/组件的功能和产出。对系统性能的评估及依据。

构建一个适用于风控、反欺诈等各种业务场景通用的决策引擎平台。

系统主要包含三个大模块

1. 策略设计器：主要提供用户配置策略的页面，可对配置的策略进行单元测试。
2. 策略执行core包，主要负责各个组件的执行逻辑。
3. 策略运行时环境，主要提供上线策略的执行环境，对外提供调用的API接口。

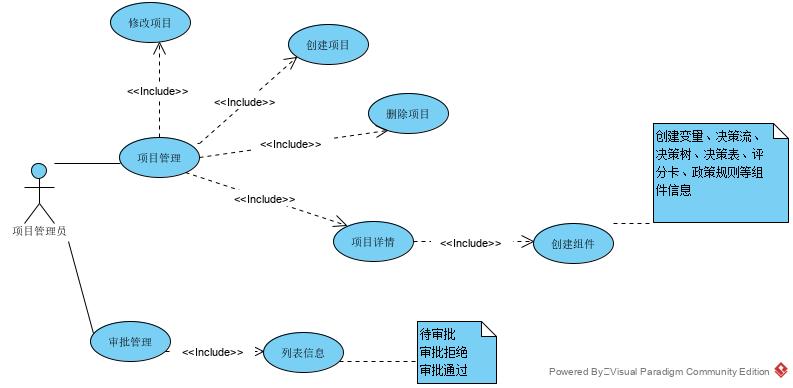
Git地址：

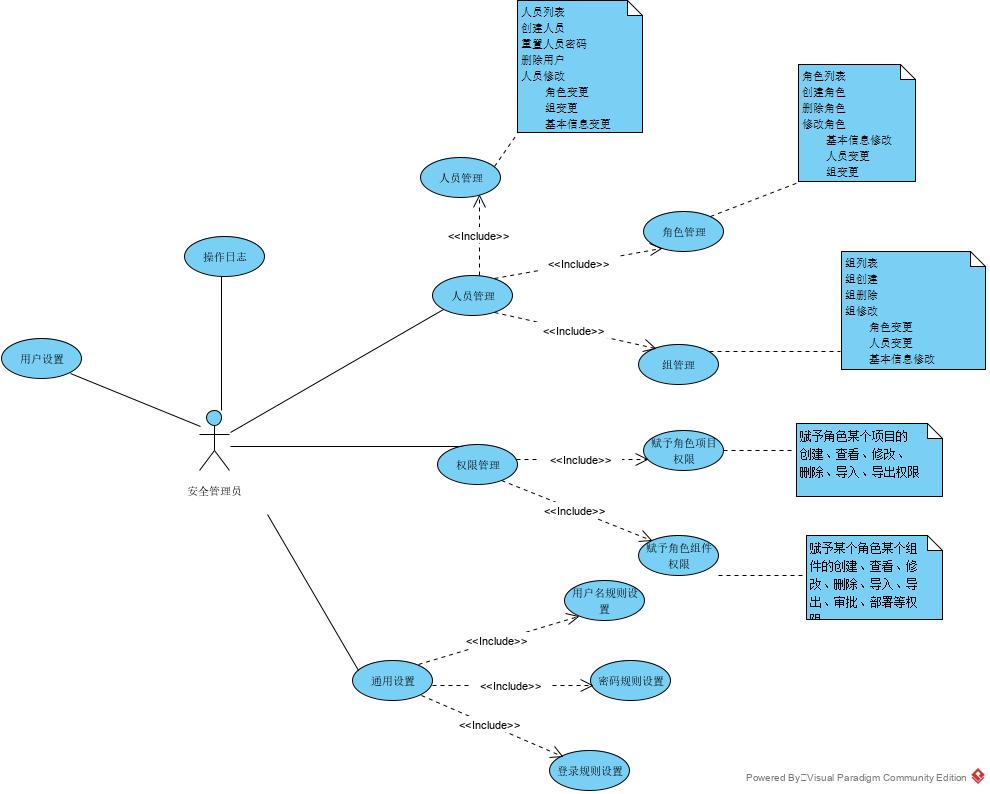
策略设计器：<http://jcode.cbpmgt.com/git/jrs-designer.git>

策略执行核心jar包：<http://jcode.cbpmgt.com/git/jrs-engine-core.git>

策略运行时环境：<http://jcode.cbpmgt.com/git/jrs-engine-runtime.git>

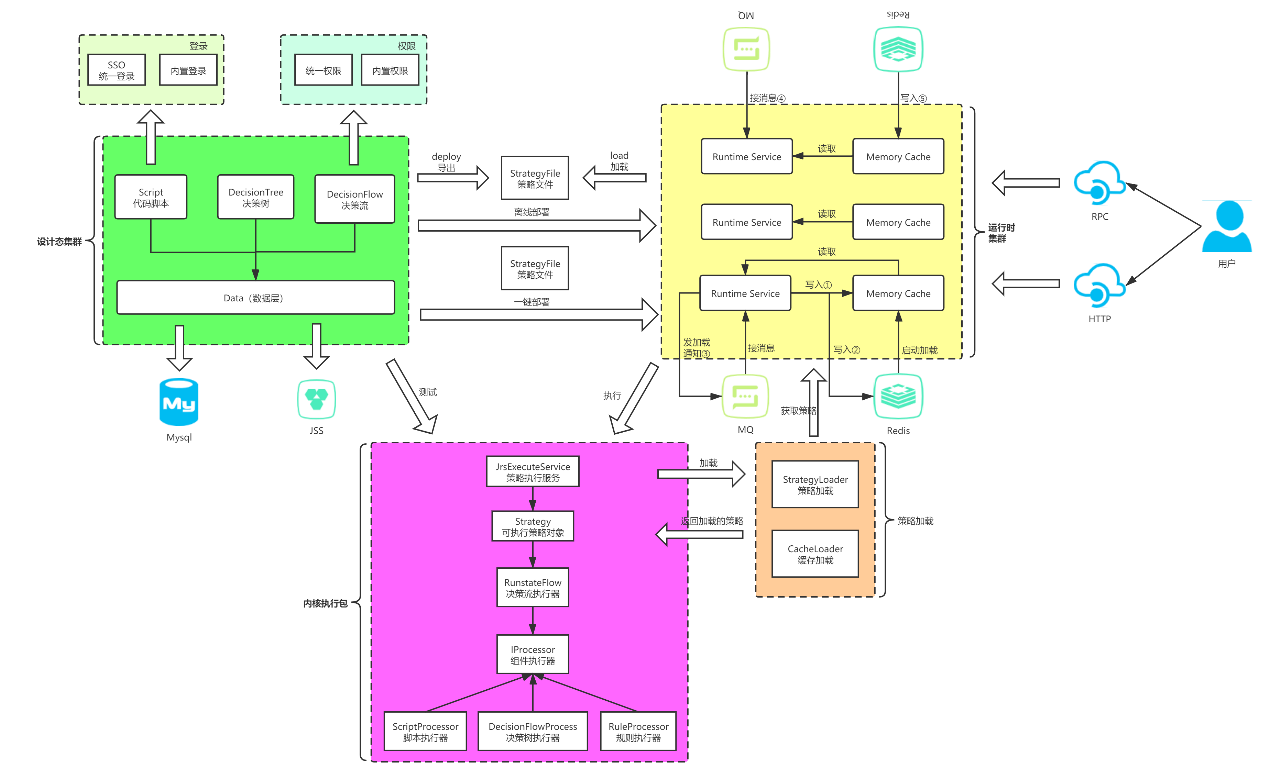
用例图：



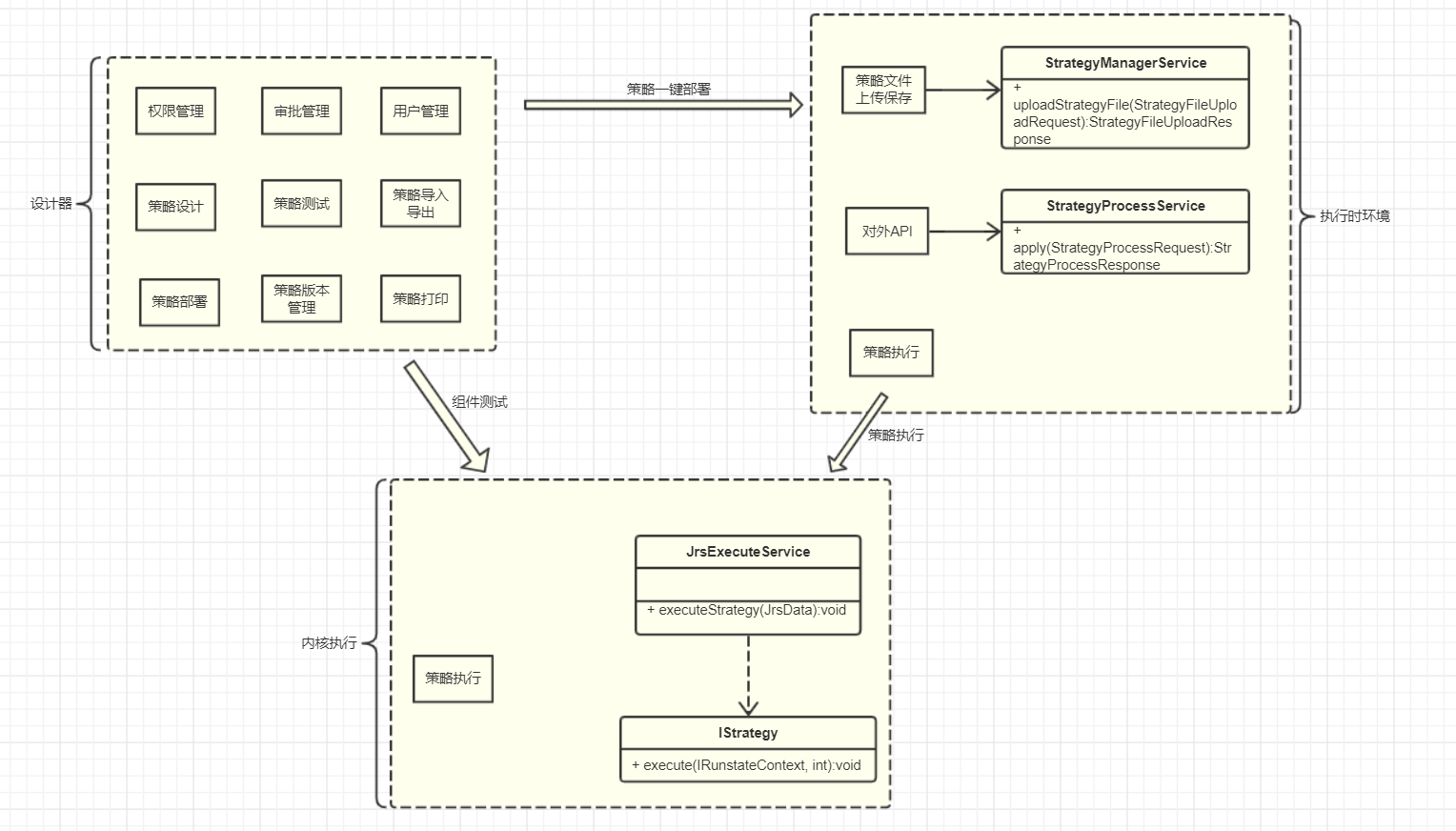


**2、系统设计**

2.1系统架构(总体架构)：子系统/组件等模块和基础设施等抽象出整体系统架构图，做到所在位置准确、角色明确、功能完备、职责清晰。

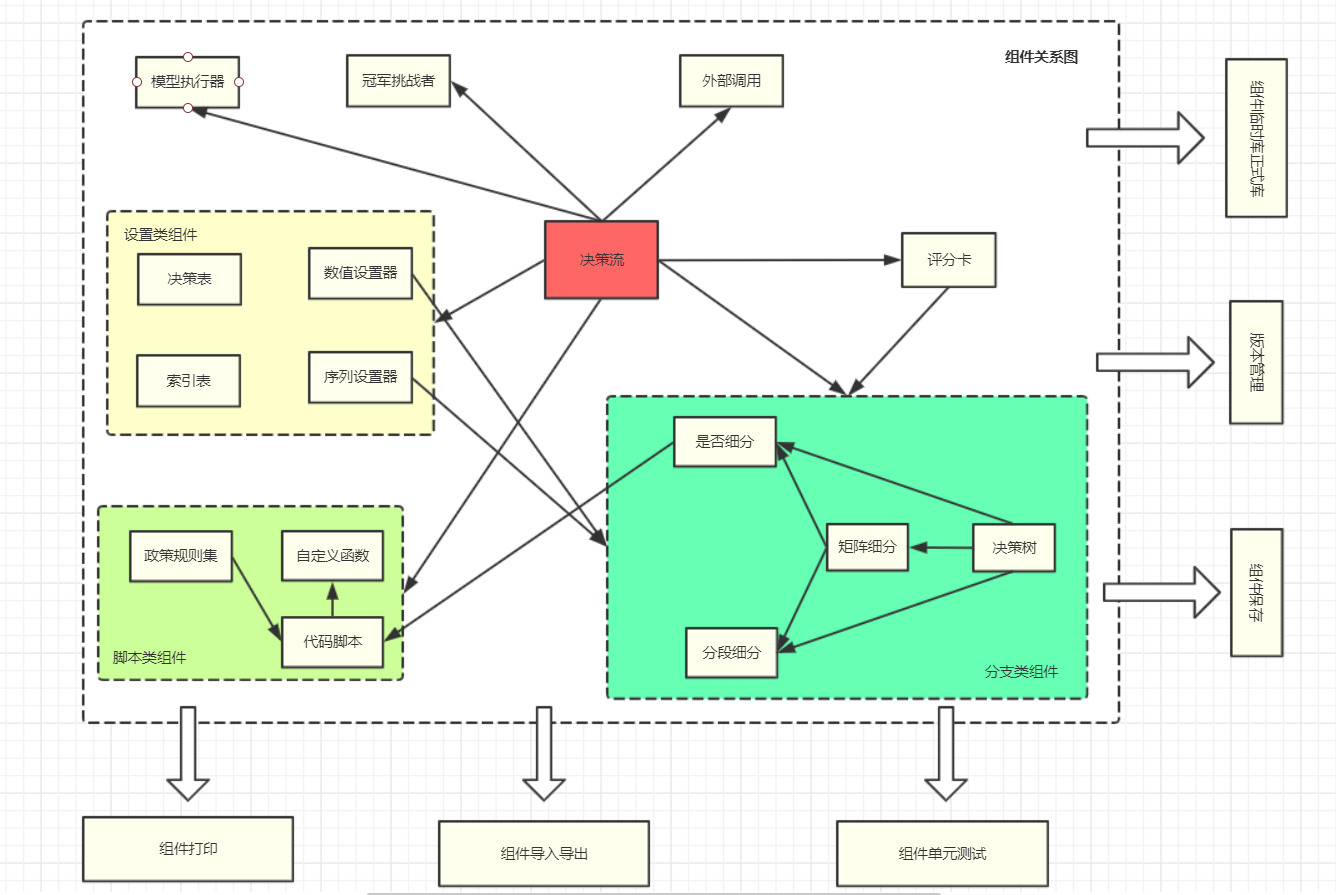


2.2子系统/组件关系：为满足设计出的userCase，使用组件图描述出每个组件核心职责，对其他组件提供的接口和依赖接口，组件核心属性等。

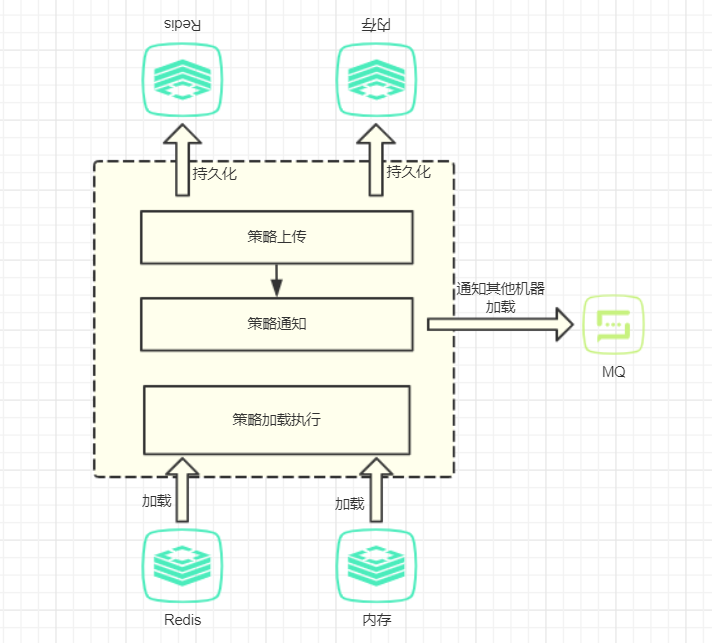


2.3 各子系统/组件设计：各个子系统/组件内部的结构及逻辑

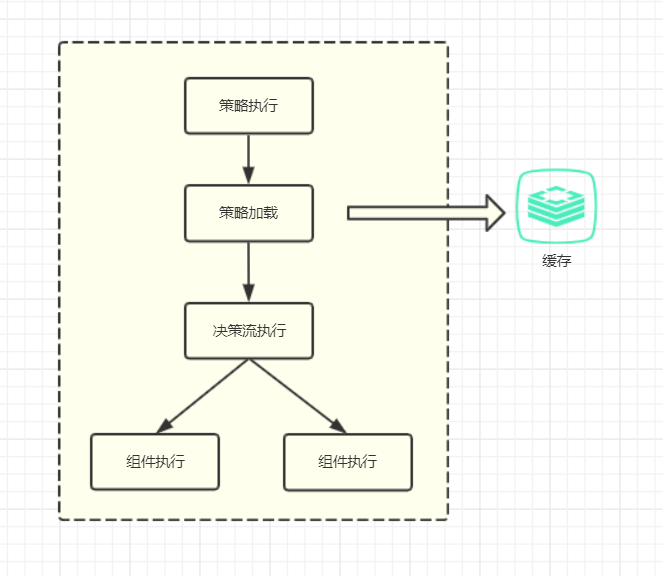
**设计器：**



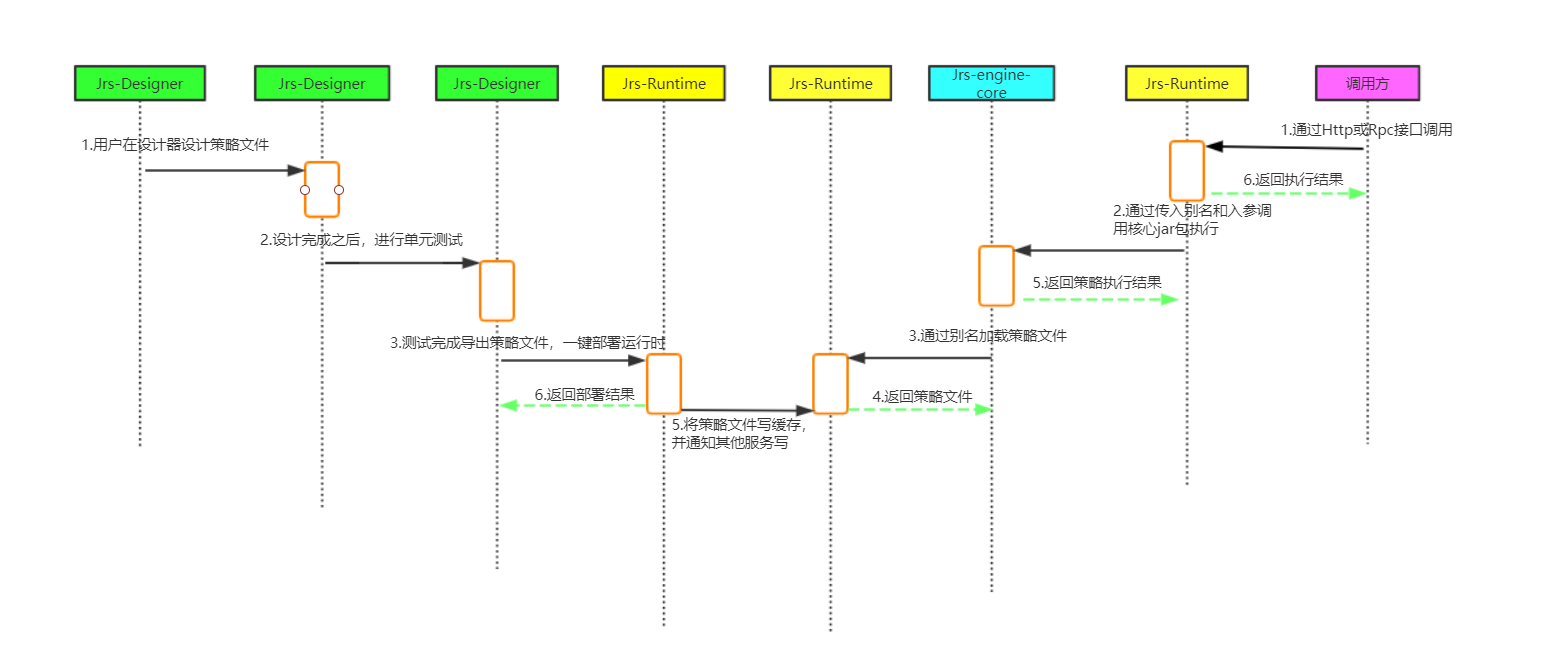
**运行时：**



**内核执行：**

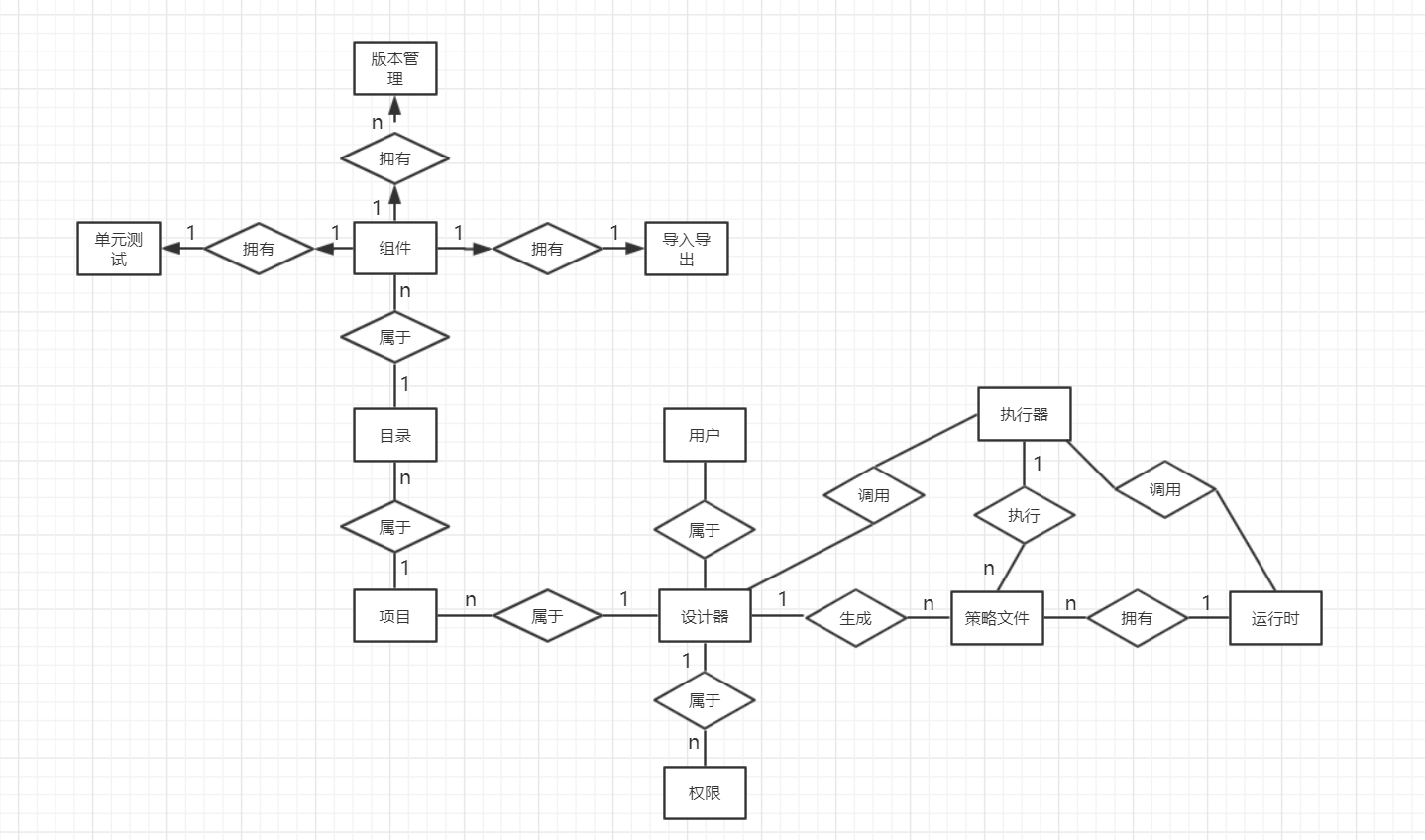


2.4核心流程：使用时序图说明子系统/组件间的交互逻辑，使用泳道图描述出业务流程（可选，在依赖外部业务时可用）。



**概念设计**

对实体做出E-R图，描述清楚内部关联。



**存储设计**

策略的定义存储在mysql，运行时环境策略文件主要缓存在JVM内存、redis中。



**可靠性保障与可扩展性**

设计态和运行时都部署了集群，支持横向扩展。 整体架构采用组件形式，可快速、简单的进行功能扩展。

**弹性设计**

暂无弹性设计

**容错处理**

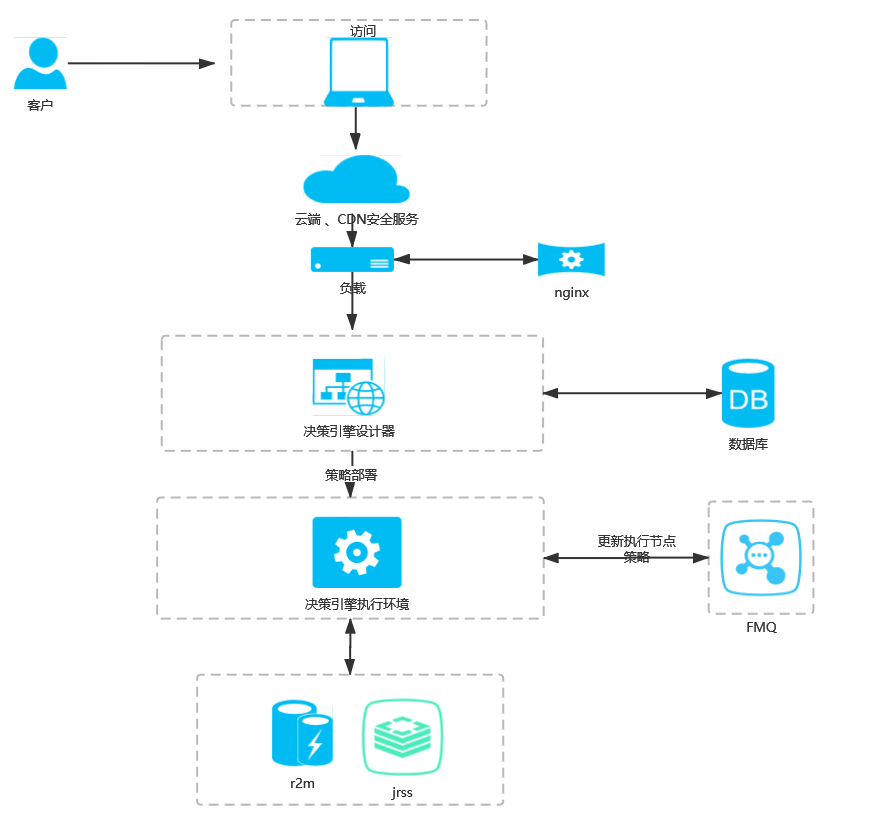
设计态保存有对组件正确性进行校验，导出时有对整个决策流进行校验，并且提供单元测试保证配置策略的逻辑正确性。运行时有对策略包、入参类型等校验，并且根据传参可打印不同级别的日志。

**高性能设计**

总结出系统哪些点上是可以做性能优化的，时效性和大吞吐的平衡策略，对缓存的使用策略和数据压缩，哪些点可以做异步化，提供具体的思路或算法。

系统设计为预编译模式，执行时已经是可执行对象，不需要再进行解析、数据获取、编译等比较耗时的操作，只进行一些逻辑运算。单机8C16G可达到2wtps，tp999在5ms以内。需要更高吞吐率时进行横向扩展，做好负载均衡既可。支持异步获取传入的参数值，在执行过程中需要的时候在获取值不必等到所有需要的入参都获取到了在执行。可以大大提高整体执行性能。

**系统部署**



设计器：4C8G既可，为了达到高可用可以部署2台以上的集群。

运行时：建议8c16G, 根据业务需要可增加机器，建议部署2台以上的集群。

依赖中间件：Redis、MQ、JSS

**可测试性**

对核心功能的原子化原则，提供低成本的测试方案。

页面提供对组件的单元测试。方便产品进行业务逻辑测试