行业中台设计文档

战法引擎

2019年06月12日

修订、签收记录

变更记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 作者 | 版本 | 变更参考 |
| 2019-06-12 | 陈远超 | v.01 | 初始化版本 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

评审记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 评审人 | 版本 | 评审/审批备注 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[1 概述 3](#_Toc10712324)

[1.1 目的 3](#_Toc10712325)

[1.2 范围 3](#_Toc10712326)

[1.3 提供方式 3](#_Toc10712327)

[2 功能规格 3](#_Toc10712328)

[3 概要设计 3](#_Toc10712329)

[3.1 0层架构 3](#_Toc10712330)

[3.1.1 上下文定义 3](#_Toc10712331)

[3.1.2 设计思路 4](#_Toc10712332)

[3.1.3 设计可选方案 4](#_Toc10712333)

[3.1.4 设计原则与约束 4](#_Toc10712334)

[3.2 逻辑设计 4](#_Toc10712335)

[3.2.1 领域模型 4](#_Toc10712336)

[3.2.2 模块设计 4](#_Toc10712337)

[3.2.3 逻辑架构 5](#_Toc10712338)

[3.2.4 业务流程 5](#_Toc10712339)

[3.3 数据库设计 7](#_Toc10712340)

[3.3.1 数据定义 7](#_Toc10712341)

[3.3.2 实体关系图（E-R图） 7](#_Toc10712342)

[3.4 接口设计 8](#_Toc10712343)

[3.4.1 南向接口 8](#_Toc10712344)

[3.4.2 北向接口 8](#_Toc10712345)

[3.5 原型设计 9](#_Toc10712346)

[4 非功能性需求 9](#_Toc10712347)

[4.1 设计目标 9](#_Toc10712348)

[4.2 设计方案 10](#_Toc10712349)

[4.3 集成需求 10](#_Toc10712350)

[5 假设与依赖 10](#_Toc10712351)

[图表 1：模块分解示意 5](#_Toc10712291)

[图表 2：逻辑架构示意 5](#_Toc10712292)

[图表 3：涉驾查稽时序图示例 6](#_Toc10712293)

[图表 4：E-R图示意 7](#_Toc10712294)

[表格 1：技术选型列表 4](#_Toc10712286)

[表格 2：设计目标示例 10](#_Toc10712287)

1. 概述

复杂的问题会在任何领域都出现，但是解决它们的总体策略通常是一样的：分而治之。我们会将问题拆分为更容易解决的子问题，然后这些方案再按照与分解相反的方式组合在一起形成整体的解决方案。通过观察会发现这样的问题是经常发生的；借助于经验，能够识别出最优的方案。这种集成模式同样适用于安防行业。

我们面临多行业多场景使用多战法协同工作以解决特性问题的业务需求，传统方式是各行业为固化的流程采用硬编码的方式串联所需的技战法，这并不利于提升产品开发效率和客户实战价值，战法引擎赋予的战法管理、战法路由和数据驱动会契合这个问题的解决方案。

* 1. 目的

战法引擎的设计目标是为前台提供平台化的底层支持，所谓平台化，就是对业务开发能以扩展，隔离的方式推进，驱动业务快速支持，根据业务需要灵活配置技战法，快速响应行业前台业务需要，加速行业产品研发效能，优化业务流程，提高用户业务效率。战法引擎达到以下设计目标：

1. 业务和引擎分离：战法引擎将不关心具体的业务，只通过抽象统一的模型去完成业务逻辑的串联和数据驱动。
2. 变化和实现分离：在复杂前台系统中，业务变动需求频繁，因此我们需要把业务变化通过战法引擎管理起来，实现变化与实现分离，通过引擎的规则配置来快速响应需求变化而不是硬编码实现，从而提高业务服务能力和系统稳定性。
   1. 范围

战法引擎适用于相对固化的、自动化的业务流程串联，不适用于需要人工干预较多的场景。

* 1. 提供方式

战法引擎以独立运行的平台（工具）提供。

1. 功能规格

描述该服务/组件的功能规格，要能与用户故事相对应。Todo: 把需求规格补充进来

1. 概要设计
   1. 总体架构



图表 1：战法引擎总体架构

* + 1. 上下文定义

描述服务/组件与周边服务/组件的功能定位、接口。

* + 1. 设计思路

描述设计思路技术选型等。

战法引擎设计思路借鉴了流程引擎（BPM）和企业服务总线（ESB）。

|  |  |  |
| --- | --- | --- |
| 组件 | 选型 | 选择原因说明 |
| 路由 | Camel | TBD |
|  |  |  |

表格 1：技术选型列表

* + 1. 设计可选方案

备选方案分析。

* + 1. 设计原则与约束

描述服务/组件设计过程的遵循标准、硬件限制、技术限制等。

* 1. 逻辑设计

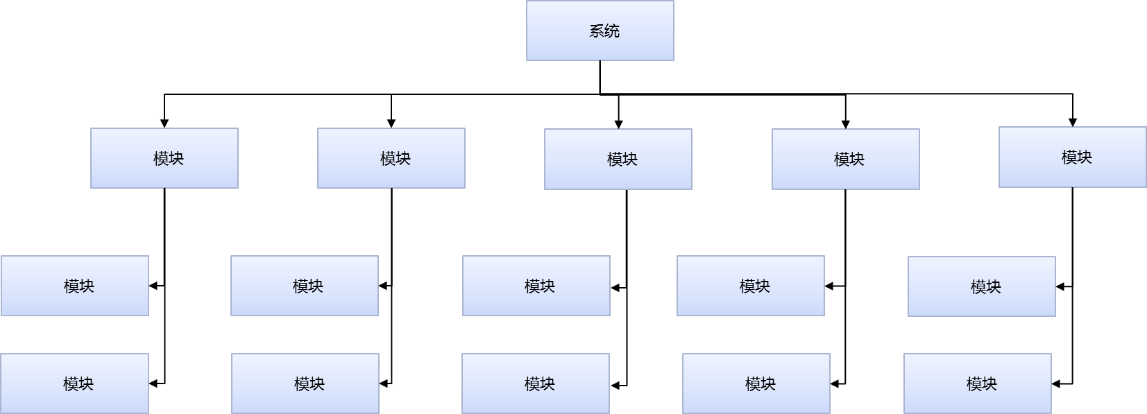
从逻辑视图描述服务/组件领域模型，模块分解，逻辑架构和流程。

* + 1. 领域模型

说明系统的核心（顶级）业务流程和核心领域对象。

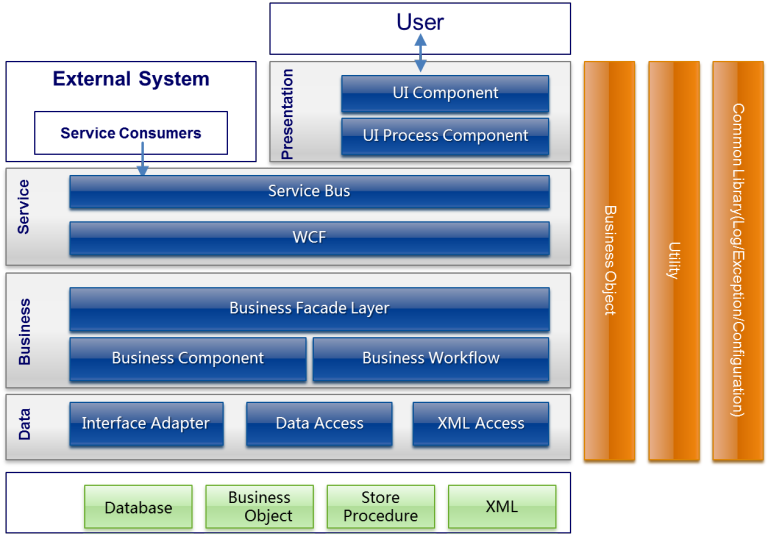
* + 1. 模块设计

说明模块分解设计。

**

图表 2：模块分解示意

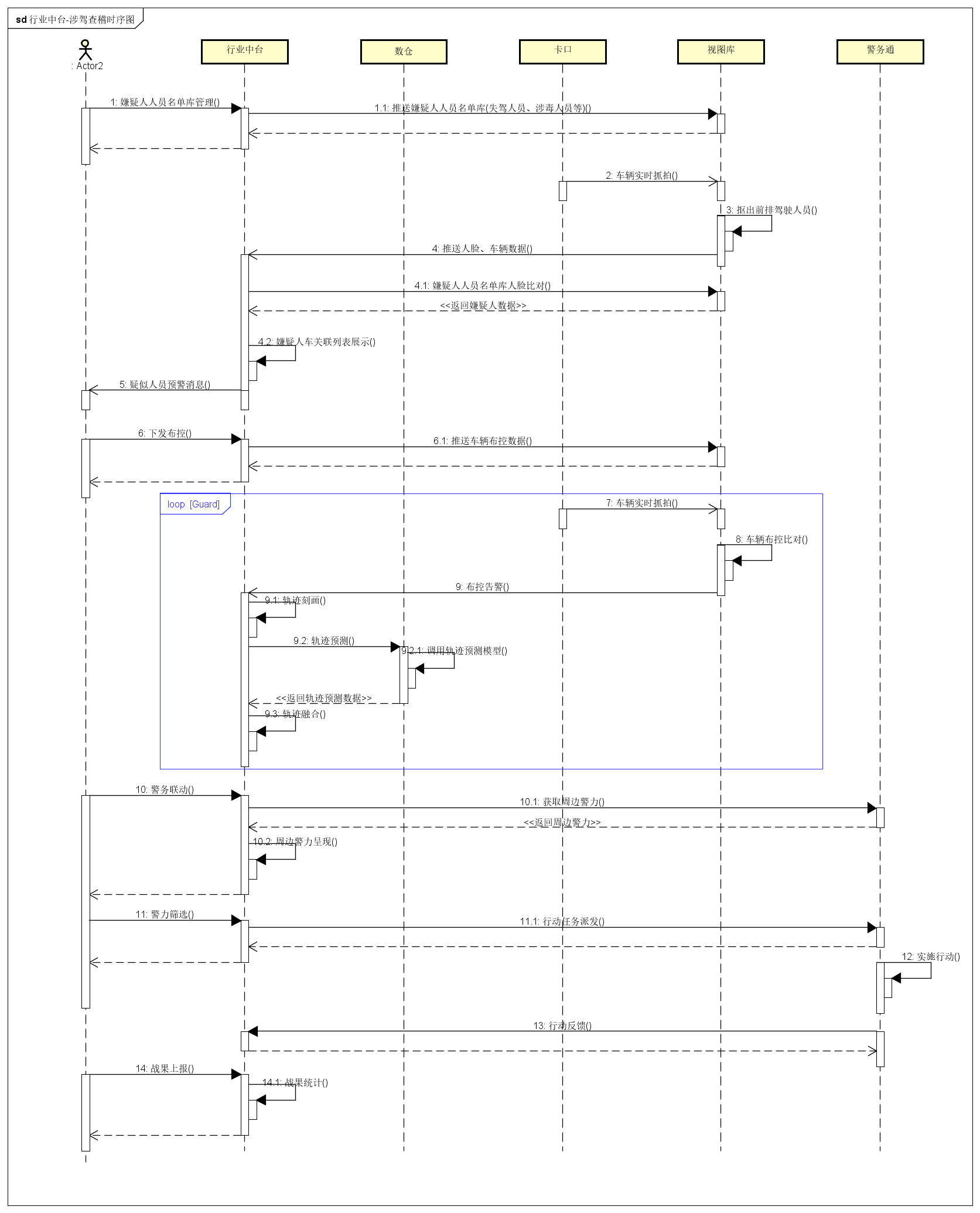
* + 1. 逻辑架构



图表 3：逻辑架构示意

* + 1. 业务流程

表现形式可以是流程图和/或时序图。



图表 4：涉驾查稽时序图示例

* 1. 数据库设计

识别出该组件/服务拥有的数据，并通过E-R图等方式对数据模型设计进行说明，如果用到其他No-SQL数据库，也要对其进行说明。

* + 1. 数据定义

列出该组件/服务相关的所有的数据存储类的实体，详细描述实体的内容并列出全部属性。对每个属性，详细描述其数据库、数据大小、特定约束。实体的所有约束及实体间的关系也要注明。

* + 1. 实体关系图（E-R图）

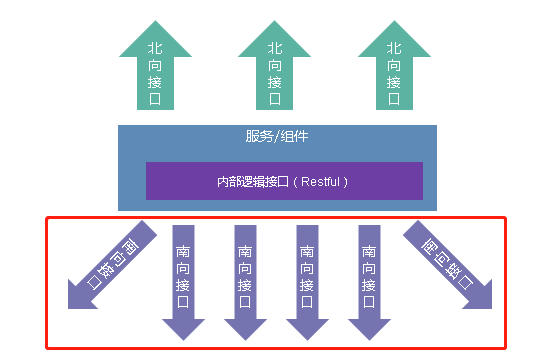
描述关于上节提到的所有实体的E-R图。



图表 5：E-R图示意

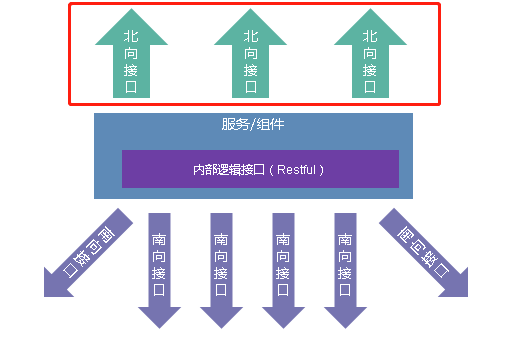
* 1. 接口设计
     1. 南向接口

描述服务/组件向下依赖的接口，南向接口的定义影响服务/组件的可扩展性。例如一种新的是数据引入的时候，要考虑如何通过南向接口去支撑它。



* + 1. 北向接口

描述服务/组件往上提供的接口，北向接口并不等同于组件/服务为实现内部逻辑而定义的Restful接口，北向接口是该服务/组件对外提供的业务能力的封装。



* 1. 原型设计

提供服务/组件的原型设计效果图，例如多维轨迹展示形式可以是2D或3D切换可选。此处原型设计稿和高保真设计稿都可以接受。

1. 非功能性需求

详细非功能性需求可参考《华智应用非功能需求规格表0321.xlsx》

* 1. 设计目标

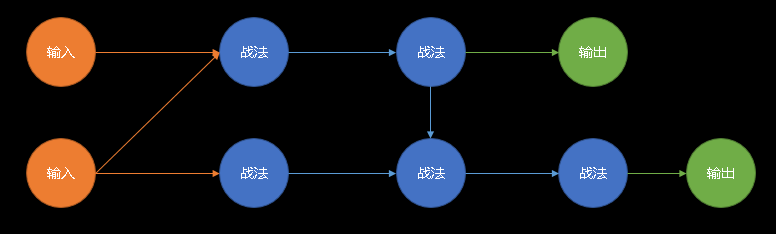
|  |  |
| --- | --- |
| 质量属性 | 要求 |
| 性能和容量 | • QPS/TPS指标  • 高峰并发用户数  • 系统响应时间要求  • 最大数据容量规格 |
| 安全 | • 使用环境（内外网），用户群体  • 安全合规要求  • 数据加密要求等 |
| 可扩展性 | • 高可用指标(例如99.9%） |
| 互操作 | • 需要支持的系统间集成协议  • 需要交互的外部系统 |
| 兼容性 | • 需要支持的浏览器版本，  • 需要兼容的特殊软件版本 |
| 数据保留与归档 | • 数据线上保留时间，业务量等  • 归档策略 |
| 备份与恢复 | • 数据如何备份恢复 |
| 可靠性 | • 业务可靠性  • 数据可靠性  • 传输可靠性等 |

表格 2：设计目标示例

* 1. 设计方案

针对如何达到上述目标的设计策略和方案，可分为多个章节。

* 1. 集成需求



图表 6：战法引擎输入输出示意图

战法引擎核心的意义在于各种技战法的串联和驱动，战法引擎的输出是一条完整业务流程的配置规则。业务系统集成战法引擎只需关注该完整业务流程的输入和输出。而

1. 集成输入：
   1. 如果该战法流程的输入需要用户界面进行，由于战法本身会提供UI界面（如果需要），因为业务系统只需通过多产品融合的方式访问启动战法的界面即可；
   2. 如果该战法流程的输入无需UI交互，业务系统集成过程则遵循传统的战法调用过程。
2. 集成输出：
   1. 如果该战法流程的输出具有UI界面，则通过多产品融合的方式直接由战法本身提供输出展现；
   2. 如果该战法流程的输出不需UI界面，业务系统集成过程则遵循传统的战法调用过程。
3. 假设与依赖
4. 战法引擎实现数据驱动依赖于消息队列，目前支持通过Kafka和Restful接口调用的方式来流转数据，因此对技战法有以下约束条件：

* 每个技战法的输入必须以Restful接口或者Kafka的方式；
* 每个技战法的输出必须输出到Kafka；

1. 战法引擎数据驱动过程中如果某一战法发生异常，整个流程链路对该异常的感知和事务回滚机制依赖对每个技战法有以下依赖：

* 每个技战法支持异常回调接口；
* 每个技战法提供事务回滚接口；