**一、系统设计规约：**

应用（43）、数据（29）、技术（15）：

均分为总体原则、系统集成原则、系统设计原则、非功能设计原则、安全设计原则

**1.1 应用**

**1.1.1 总体原则（9条）**

1）我行应用构架规划

2）应用分层、高内聚低耦合、组件化和参数化

3）关键功能、非功能需求和设计约束决定技术方案

4）避免过度设计，解决根本问题，先保证解决方案简单可用，在考虑通用性和复杂性

5）重大方案设计需要主备，并有方案对比

6）前后端分离，业务逻辑和数据数据访问隔离，通用和专用功能分离

7）系统内OLTP和OLAP分离

8）海外和子公司系统设计遵循多法人、多语言、多时区

9）过渡方案须支持向目标演进

**1.1.2 系统集成原则 （8条）**

1）行内系统间联机交互通过交换平台

2）交换平台不负责交易组合

3）行外系统接入通过外联平台

4）分行特色外部接入在总行外联分行专区上实现，分行特色应基于快捷协作平台

5）联机批文件传输通过通用文件传输平台，批量数据文件传输通过fex，跨网络区域文件通过跨网文件传输系统

6）批处理调度交易类系统使用Entegor，非交易类使用ETLPLUS

7）行内用户通过统一身份认证平台进行身份认证

8）保证金额域上送、转换和处理正确

**1.1.3 系统设计原则（9条）**

1）客户编号由核心统一生产

2）业务系统作为签约主系统，保留签约主体，核心系统存放签约索引并控制签约护持

3）发起方系统生产发起方流水号，并作为全局唯一流水号

4）涉及账务的冲正由发起方系统发起，冲正服务方须支持正交易和反交易

5）重发时，发起方使用原交易流水号，发起方和服务方都应具备防重机制

6）大数据量查询交易有分页和结果集条数控制机制

7）批处理设计有并发、断点续做和重跑机制

8）交易类系统需按7\*24小时服务设计

9）新建系统字符集需utf-8

**1.1.4 非功能设计原则（8条）**

1）错误信息分级分类，错误信息提示正确完整，不得误导客户

2）高可用和可扩张性采用集群方式

3）有容错机制

4）考虑流量控制和故障隔离

5）交易有超时设计且符合漏斗原则

6）可维护性具备应用监控、日志和巡检自动化设计

7）容量设计支持3年以上业务发展需要

8）有版本回退策略

**1.1.5 安全设计原则（9条）**

1）最小授权原则

2）所有客户交易需进行权限检查

3）各渠道相同交易安全认证强度须保持一致

4）复杂流程交易进行前后手检查

5）外部产生输入输出进行合法性校验

6）源代码需进行安全代码扫描

7）程序发布前删除调试文件

8）移动客户端加固后发布

9）动帐、开户、签约等重要交易进行实时风险控制

**1.2 数据**

**1.2.1 总体原则 （7条）**

1）我行数据架构规划

2）我行数据架构分层、集成、整合和共享

3）制定数据全生命周期管理策略

4）我行数据库、大数据设计和开发规范

5）数据标准须通过数据字典贯标

6）相同数据在各个系统中定义须保持一致

7）数据加工和访问分离、冷热数据分离、动静数据分离

**1.2.2 系统集成原则（5条）**

1）明确数据主题，主本向副本单向同步原则

2）同一份数据一次采集多次使用

3）系统间批量文件交互时，由上游系统负责准备数据，主动通知并提供给下游

4）全行共用报表在同一报表平台加工

5）减少固定报表开发，鼓励业务自主定义报表

**1.2.3 系统设计原则 （8条）**

1）数据模型需经过逻辑模型和物理模型两个阶段

2）所有表有主键，并有合理索引

3）表结构设计包含银行号和时间戳字段

4）禁止物理删除数据记录

5）单张表数据量超过500万行可考虑分区或分表，超过1000万时必须分区或分表

6）数据库设计须支持3年以上存储空间和表空间容量需求

7）尽量减少表关联，多表关联原则上不超过3张

8）有当前数据转移到历史数据机制

**1.2.4 非功能设计原则（5条）**

1）数据链路设计考虑端到端时效性

2）数据服务方保证数据质量

3）数据消费方提出数据服务水平要求

4）数据消费方有数据监测和异常处理设计

5）服务方日常对外提供批量数据时已采取增量方式

**1.2.5 安全设计原则 （4条）**

1）开发和测试环境数据脱敏

2）敏感数据加密存储和传输

3）敏感数据禁止下发到客户端

4）敏感信息展现时须屏蔽

**1.3 技术**

**1.3.1 总体原则（4条）**

1）我行技术架构规范

2）新建系统使用我行应用开发平台

3）根据系统重要性定级，确定灾备策略

4）从准入清单中选择商用和开源软件产品，并选择主推版本

**1.3.2 系统集成原则（2条）**

1）系统间连结使用同步短连接

2）发布订阅和时间流场景须采用事件驱动处理机制（如kafka）

**1.3.3 系统设计原则（3条）**

1）通用功能须使用我行已有企业级公共技术组件（验密和OCR识别）

2）新建系统须支持win10和IE 11

3）积极引入新技术解决传统技术无法解决的问题

**1.3.4 非功能设计原则（2条）**

1）应用开发平台固化非功能技术组件

2）新建系统接入日志中心

**1.3.5 安全设计原则（4条）**

1）优先采用国密算法

2）交易日志打印审计要素

3）交易日志中禁止打印敏感数据

4）客户端禁止打印交易日志

**二、架构设计：**

**2.0 总体设计：**

包括上下文关系、应用架构、数据架构、技术架构、安全架构（数据、应用安全、交易认证、业务安全）、部署架构

**2.1 思路，原则，假设/前提：**

**2.1.1 思路**

1）从业务目标、it规划挖掘前瞻性、扩张性需求考虑

2）形成方案设计指导策略，给出定位、范围等

3）新建应用系统，理由应充分，定位清晰合理

4）新建应用系统与行内现有系统存在功能或数据交叠，应考虑整合（无法整合，说明原因）

5）已有相关应用系统的改造，功能分布应清晰合理

6）组件化、服务化、前后端分离的设计原则

**2.1.2 原则**

给出本次设计遵循的基本原则，可引用已发布架构决策

**2.1.3 前提/假设**

方案实施的前提条件

**2.2 上下文关系**

1）从全局和概览角度展示业务需求涉及的所有系统

2）主要描述系统与系统之间的关系，展示系统全景图

3）为分析人员、系统设计人员进行需求分析、功能拆分、方案设计打下基础

**2.3 应用架构**

**2.3.1 本期建设范围**

1）主办系统视角描述系统的业务功能和基础支撑功能

2）系统功能进行组件化设计（内部高内聚，之间松耦合）

3）通过应用架构设计，为主办系统的设计和开发打下坚实基础

**2.3.2 目标功能模块成熟度**

1）已成熟稳定、未成熟优化（框色标识）

2）包含本次立项新增功能模块

**2.3.3 目标功能模块自主实施度**

1）合作研发：厂商和行内员工

2）自主研发：只有行内员工

3）厂商研发：只有厂商

4）外购模块：厂商提供、我行购买、没有源码

**2.4 数据架构**

1）从全局角度描述业务需求涉及的信息在我行系统上数据分布和流向，通过数据架构图表示

2）画法：

主办系统位于中心

数据提供系统在ppt上方，消费系统在下方

标明系统间数据流向

数据分类采用抽象方式

**2.5 技术构架**

1）描述方案主要系统使用的技术产品：

操作系统的类型、版本

数据库类型、版本；中间件类型、版本

开发工具类型、版本

浏览器类型、版本

**2.6 部署架构：**

描述主要系统的物理部署形态：

1）网络区域：标识分区部署，按照现有网络划分（网络DMA区、行内交易区、行内MIS区、行内核心区）摆布物理设备

2）物理设备：设备类型、内存、cpu核数、存储容量、标出高可用性的部署实施策略（APP, WEB Server, DB）

3）新建系统，画出系统灾备方式和技术，涵盖同城异地

**2.7 安全架构**

**2.7.1 数据保护：**

1）数据安全：敏感数据使用软键盘加密、传输使用算法加密、敏感数据屏蔽展现、密码使用总行密码配置工具，加密存储、用户密码采用加盐哈希方式（SM3）

2）隐私保护（app）：有隐私保护协议、明示客户、客户可选择同意

**2.7.2 应用安全：**

1）服务端安全：权限检查、SQL注入防范、前后手校验、重放

2）客户端安全：客户端混淆、客户端加固

3）API安全：消费方系统认证和权限控制

**2.7.3 交易认证：**

1）登录方式认证

2）交易方式认证

3）其他重要操作

**2.7.4 业务安全：**

1）锁定：密码错误超限锁定

2）限额：交易限额

3）权限：

4）风控：

**三、 工厂化软件设计：**

应用系统分为3层：前端、后端、数据库及后台服务

**3.1 问题：**

1）前端页面跳转及组件调用冗余重复，后端业务流程实现质量参差不齐

2）前后端应用开发效率不高，缺陷率影响功能快速交付

3）数据建模缺乏统一的流程及规范

4）设计及开发管控方面缺乏有效的机制，管控的效果差

5）各方人员组织的层面粗放，无法发挥团队合力

**3.2 主体思想**

1）前后端分离

2）根据前后端不同特点，采用不同的抽象思路和方法抽离公共的技术组件和业务组件以及前后端流程模版

3）组建模版组建车间，提前在概要设计把公共组件和模板予以实现

4）系统其他功能设计，引入模板和公共组件，分析差异后进行自定义组件设计和开发，最终实现完整的前后端功能

5）模块化、组件化、模板化：开发方面，实现了降低耦合度、便于多开发协作、降低缺陷率、降低开发和维护成本；产品方面，对业务产品的功能逻辑具有指导性，有利于保持交互一致性、客户体验一致性。

**3.3 方法框架**

1）场景抽象和功能装配：

对前端页面流程和后端功能进行场景归纳总结，形成通用场景模型，分离公共和差异部分，抽象和提炼形成适合多类功能的前端页面流程模板和后端功能流程模板。（流程模版：类比生产线，前端开发的功能，基于同一套模板，页面流程相同，公共组件保留，引入适合自己功能需求的自定义组件的个别的公共组件）

2）前后端开发方法：

i）根据开发特点和功能建立模板组件车间，基于对功能需求的分析抽象，形成技术组建，业务组件，基于对场景流程的分析归类，形成流程模板

ii）建立多套功能开发车间

iii）功能开发生产线式产出

iv）严格的质量检查

3）设计与开发管控

4）人员组织管理

5）数据建模方法

**3.4 内容适用项目类型**

1）新建、重构的系统项目，不适合存量系统日常优化改造

2）中大型项目的联机交易实现，不适合小型项目及批量交易实现

3）工厂化研发方法思想主要体现在设计阶段和开发阶段

**3.5 前端设计：**

**3.5.1 分层：**

1）基础平台：选用行内推荐的开发平台

2）应用框架：采用比较流行的前端框架

3）页面组件/页面流程：用于展现前端业务逻辑（每页的业务内容，各页面间跳转流程）

**3.5.2 设计原则：**

1）一致性：与现实生活一致、在界面中一致

2）反馈性：控制反馈、页面反馈

3）效率：简化流程、帮助客户识别

4）可控：用户决策、结果可控

**3.5.3 前端页面流程模板提取方法**

1）思路：见3.3 1）注：页面由页面要素组成，页面要素由公共或自定义组件实现

2）方法：

i）业务功能视角:

业务功能为分类维度，如支付转账、理财产品、定期存款、通知存款、账户管理等

提取每类功能特点，提取共性

ii）页面操作视角：

前端每一个页面的展现或者用户操作动作，都是为了完成一个操作

基于该操作，对功能进行二次归类

iii）页面流程视角：

归纳总结，形成页面流程模板

iv）补充公共机制：

签名、蒙板、防重、随机数、结果页、提示页、通讯等公共机制需含在流程模板中

**3.5.4 前端组件抽象方法**

1）技术组件：基础组件、数据展示、数据输入、数据校验、数据交互、安全相关

2）业务组件：客户类、账户类、金额类、日期类、银行选择类、摘要类、其他

**3.6 后端设计：**

**3.6.1 分层：**

1）基础平台

2）应用框架

3）技术组件：与业务无关的通用类组件（工具类、通讯类等）

4）业务组件：完成一个独立完整的最小粒度的业务功能

5）业务流程

**3.6.2 设计原则：**

1）统一

2）先进

3）高可靠/安全

4）标准化

5）成熟型

6）适用性

7）可扩展性

**3.6.3 后端业务流程模板抽象方法**

1）思想：见3.3 1）注：该流程模板固化，由公共子流程、可扩子流程、公共组件、自定义组件组成。

2）方法：

i）业务功能场景视角：同3.5.2 i）

ii）业务处理场景视角：与前端页面流程共同完成对应的业务功能

iii）业务处理流程视角：后端提交类功能通常有操作权限校验、账户权限校验、操作流水记录、日志记录、入库数据准备、入库操作、接口数据准备、发送接口、防重发处理、接口返回处理等；查询类通常有操作权限校验、账户权限校验、操作流水记录、日志记录、组织查询条件数据、数据库查询、后台系统接口查询等

iv）补充公共机制：操作权限校验、防重发、数据通讯等相关操作

**3.6.4 后端组件抽象方法**

1）技术组件：数据类操作、数据库操作类、数据通讯类、文件操作类、日期时间类、日志类、安全类、编码转换、线程调度及流程执行类、异常处理类

2）业务组件：客户类/用户类、账户类、限额类、收款人类、功能类

**四、应用性能设计与应用高可用性设计**

**4.1 应用性能设计**

**4.1.1 容量需求评估：**

1）存量系统：

i）渠道服务、客户服务、流程层系统，以用户数（

ii）产品服务系统，以客户数、账户数与产品种类乘积

iii）信息管理类系统，以整个数据库空间

注：i）ii）至少按照前2年最大增长率的两倍预估

2）新建系统：

i）已有类系统，采用类比法

ii）没有类似系统，考虑业务规划与经验值相结合

**4.1.2 通讯调度层性能设计**

1）通讯协议（TCP/UDP）:

i）行内应用系统的通讯程序不使用网络层及以下两层

ii）交易数据一般用TCP

iii）局域网环境，建议将UDP数据控制在1472字节以下

iv）Internet编程时，建议将UDP数据控制在548字节以下。

注：OSI七层模型：应用层、表示层、会话层、传输层、网络层、数据链路层、物理层

2）长短连接：

i）长连接：交互频繁、点对点、连接数不太多、网络状况较好

ii）短连接：交互不频繁、client连接server、网络状况不好

3）同步异步：

i）同步：（短连接）单笔金融交易、查询交易面向客户（但是，前置机之后的中间环节到核心系统采用异步）、后台的进程池/线程池的多个进程/线程

ii）异步：（长连接）批量交易

4）单进程与多进程/多线程：

i）服务端程序设计不采用单进程

ii）与第三方长连接通讯，双方可能都采用固定的收发处理进程

iii）TCP短连接通讯，采用多进程或多线程的处理方式

iv）交易量小、并发不大的场景，采用多进程方式（C语言）

v）交易量大、并发大的场景，采用多线程方式（Java语言）

5）本机进程间通讯（IPC）：

i）定义：在不同进程间传播或交换信息

ii）低级通讯：控制信息的通讯

包括互斥锁、条件变量、读写锁、文件和记录锁、信号量

iii）高级通讯：大批量数据的通讯

包括消息传递类、共享内存区

**4.1.3 数据缓存**

1）针对数据库中变化不大但频繁访问的数据

2）提高系统的TPS

3）主流共享缓存：Redis

4）读写分离机制（DB2考虑HARD\CDC等机制）

5）静态和动态库

i）静态库：程序体积会膨胀（以空间换时间）

ii）动态库：只保留库文件名字和函数名，体积基本不变（以时间换空间）

7）限制查询大小：避免循环内进行不必要的sql查询；大量数据查询，采用分页查询，避免内存溢出

**4.2 高可用性设计**

**4.2.1 影响因素：**

1）服务停用：

i）计划内：日常操作、系统维护、升级

ii）计划外：系统错误、数据或磁盘问题、停电

2）网络服务、数据中心基础服务、数据存储服务、操作系统和基础软件服务、中间件等应用软件和应用依赖服务、应用程序自身

**4.2.2 高可用设计原则与策略：**

1）原则：

i）前瞻性

ii）适用性：不存在统一业界标准

iii）渐进性

2）策略：

i）故障设计：避免故障、降低影响、故障检查、快速恢复

ii）思想：冗余备份、减少依赖、分层分治、自动化智能化

4.2.3 常用方法：

1）分层（横向）分级（纵向）：

i）分层：接入层、应用层、服务层、数据层

ii）分级：按功能、重要性分隔为独立应用或者独立服务；

通过服务调用组合实现系统整体功能；

针对应用和服务的重要性设计不同的高可用方案

2）冗余设计：自动侦测、自动切换/故障转换、自动恢复、动态扩容、灰度发布

3）故障监控：基础服务、应用资源、交易处理、业务统计

服务端监控实施方案：

i）基于网络报文

ii) 基于应用日志

iii）基于数据库数据

4）故障隔离：物理部署、应用程序、应用资源

5）服务降级：

i）常见：超时降级、失败次数降级、故障降级、限流降级

ii）手段：功能禁用、仅提供基础服务、使用本地缓存代替远程调用服务

应用监控：应用资源、交易（报文、日志、监控数据库数据）

**4.3 冗余设计**

**4.3.1 主从模式（主备）**

i）应用场景：数据服务、服务主从模式、分布式模式中数据结点的冗余设计

ii）需要完善的切换策略

**4.3.2 双主模式**

i）应用场景：数据库服务、基于Keepalived的双主模式

ii）优点：不需要切换、客户端无需区分不同角色的主机

iii）缺点：必须保证数据双向复制

**4.3.3 集群模式**

i）应用场景：应用服务集群

**4.3.4 分布式**

i）应用场景：基于Zookeeper的分布式服务管理、使用Raft算法的分布式服务管理

**五、生产事件警示：**

**5.1 需求阶段：流量控制**

1）目标：局部故障不发散、防止系统瘫痪

2）原则：

i）保证自身安全、隔离故障影响范围的义务

ii）“各司其职”

iii）外联系统或因特殊原因无法实现部分或全部流量控制机制，在其直接发送节点根据需要实现对应的控制

iv）应用层面的流量控制粒度分为：控制系统、端口、单一交易流量

**5.2 总体方案设计阶段**

1）传输过载：排队等待、快速失败、预热模式

2）数据移植：重要数据清理、转换与检核设计

3）交易链路：漏斗设计原则

4）数据存储分离：在线存储、近线存储、离线存储

**5.3 概设及详细设计阶段**

关注性能、高可用性等非功能设计