敏捷开发

Scrum 作为敏捷方法论的重要组成部分，着重强调团队间的协作、灵活应对各种变化以及持续交付高品质的产品
Scrum 包括了3 个核心角色
    产品负责人、敏捷教练、开发团队
        产品负责人肩负着定义产品愿景以及管理待办事项列表的任务
        敏捷教练他作为流程的专家，确保Scrum 流程得到正确执行，保障团队遵循Scrum 的原则和规则开展工作
        开发团队则去完成具体的编码测试上线工作
Scrum 包含产品待办事项列表、冲刺待办事项列表以及增量
Scrum 的主要活动包含冲刺计划会议、每日站会、冲刺评审以及冲刺回顾
以用户故事为核心的需求拆分机制可有效应对软考阅卷系统的需求多变性
短周期迭代开发模式可提升系统的开发效率与交付质量
每个迭代严格完成“需求聚焦-设计简化-开发核心功能-测试验证”的闭环
    召开用户演示会，用实际可操作的系统代替文档沟通，把模糊的需求转化为具体的优化点，避免后期大规模返工
通过“小步交付+即时反馈”让开发始终对准用户真实需求，这是提升系统开发效率与交付质量的核心逻辑

DevOps

DevOps 在开发时期的主要活动包括持续集成、自动化测试、配置管理和交付准备，其核心目标是通过流程自动化与团队协作，提升效率与质量，缩短交付周期并快速响应需求
DevOps 注重跨团队协作与反馈，开发、测试和运维人员借助敏捷迭代、代码评审、看板和日志监控实现信息共享，提升协同效率
DevOps 自动化工具链和跨团队协作机制
    DevOps 的自动化工具链可提升软考阅卷系统的交付效率与质量保障能力
        Jenkins 搭建持续集成流水线，开发提交代码后自动触发编译、单元测试和集成测试
        部署则用Ansible 编写标准化剧本，包含环境预检查、配置文件模板化替换、服务灰度启停等步骤
    DevOps 的跨团队协作机制是保障系统稳定运行与需求响应能力的关键
        组建开发、运维、测试联合工作组，推行全生命周期参与
            需求分析阶段就拉通评
            每日站会同步进度
DevOps 的核心不是工具本身，而是通过流程自动化将“不可控的人工操作”转化为“可重复的标准步骤”

自动化测试

自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程，能显著提高测试效率和质量
其主要手段围绕系统核心需求展开
    单元测试:JUnit
    接口测试，通过Postman+Newman 构建自动化脚本
    UI 自动化测试，借助Selenium 模拟阅卷人员操作流程
    性能测试，用JMeter 模拟并发场景,验证系统响应速度与稳定性，避免高负载下卡顿
传统的人工测试方式已难以满足系统功能复杂和频繁迭代的需求。而自动化测试作为一种更高效、更精准的测试手段两个关键作用
    提升复杂功能测试的效率与准确性
        人工测试“耗时长、易出错”
        自动化测试脚本模拟
    测试数据生成工具批量测试数据 核心价值，不在于工具本身，而在于将人工测试中“重复性高、易受主观因素影响”的工作，转化为“可批量执行、结果可精准校验”的标准化流程，这才是提升系统复杂功能测试效率与准确性的关键。

以及保障系统迭代的稳定性与回归测试的效率

人工回归测试的“经验依赖”和“覆盖不全”是系统迭代的重大风险

梳理出核心业务流程的测试用例，将这些用例编写成标准化自动化脚本，统一存储　自动化测试的核心不是“自动化”本身，而是通过“用例固化”和“重复调用”，把人工测试中“不可控的记忆依赖”转化为“可追溯的标准流程”，这才是保障系统在频繁迭代中依然稳定运行的关键。

服务网格

采用服务网格来优化云原生架构
服务网格是一种专门处理服务间通信的基础设施层，它为云原生架构带来诸多关键作用
服务网格能够实现流量管理，提高系统处理效率
能增强安全性
可观测性也是其重要优势，能收集服务间的请求信息，生成详细的日志和监控指标，发现并解决潜在问题
的高并发场景下支持服务实例自动扩缩容，且能通过灰度发布让新功能先在小范围节点验证，再逐步全量推送，降低迭代风险
围绕服务网格在提升任务处理效率和保障数据传输安全这两个关键,云原生应用中的服务管理和消息传递始终是一个比较棘手的问题

服务网格通过精细化流量管理可提升系统的任务处理效率

高并发阅卷场景下的资源调度难题

基于任务特性设计智能路由策略 结合权重路由机制，实时监控各实例负载，动态调整任务分配权重，紧急任务优先流向负载低于阈值的节点

其实服务网格的核心价值不在于工具本身，而在于通过动态、精准的任务适配，将“被动应对资源瓶颈”转为“主动优化资源调度”，这才是提升系统任务处理效率的关键。
服务网格通过构建服务间安全通信机制可保障数据的传输安全性

配置SSL 证书方式

自动化脚本批量更新证书

跨环境同步困难

伪造风险

引入服务网格

服务网格的mTLS 功能自动对服务间通信流量进行端到端加密

采用SPIFFE ID 作为服务的唯一身份标识

加入请求签名与校验机制

其实服务网格的关键不是新增了安全功能，而是将分散在各服务中的安全逻辑剥离出来，形成统一的安全管控平面，这才是解决微服务规模扩大后通信安全问题的核心

Serverless

Serverless 并非意味着没有服务器，而是指开发者无需管理服务器基础设施，将精力聚焦于业务代码
云服务提供商负责服务器的运维、扩展等工作

主要包含函数即服务（FaaS）和后端即服务（BaaS）。

FaaS 允许开发者上传代码片段，云平台按需执行并收费

BaaS 则提供数据库、存储等后端服务

Serverless 可根据实际请求量自动弹性伸缩，能精准应对高并发场景，降低成本
无需花费时间管理服务器，能快速迭代系统功能，提高开发效
Serverless 架构天然具备高可用性和容错性，能保障系统稳定运行，为系统提供可靠支持
Serverless 架构在软考阅卷系统中的两个关键作用

一是提升系统的资源弹性与成本优化能力

资源分配的难题

引入Serverless 架构：将核心的试卷评分处理模块拆分为独立的处理函数，平时不占用资源,处理完成后

立即释放资源

弹性扩展的后端服务，根据实际数据量动态调整存储和计算能力

二是增强系统的开发效率与业务迭代速度

精力被服务器运维工作占据

采用FaaS 模式，将评分规则逻辑封装成独立函数，每次更新时只需上传代码，通过阿里云平台控制台快速完成测试和部署，省去了环境配置的繁琐步骤

借助BaaS服务集成云数据库和身份认证能力，无需再编写大量基础设施代码，让开发者能专注于评分规则的逻辑优化

Serverless 的核心不是"没有服务器"，而是将资源分配从"人为主观预估"转变为"系统按需响应"，这才是解决系统资源弹性与成本矛盾

Serverless 架构的价值不仅在于技术层面的简化，更在于通过解放开发者精力，让团队能更快速地响应业务变化，这对时间敏感的系统来说至关重要。

存储计算分离

存储计算分离模式是将数据存储和数据计算的功能分开处理

传统模式，存储和计算资源通常绑定，高并发，性能瓶颈

存储计算分离模式可以独立扩展存储和计算资源，提高系统的灵活性和可扩展性

存储节点可以专注于高效稳定地存储海量试卷数据，即使数据量不断增加，也能通过增加存储设备轻松应对

计算节点则可以根据阅卷的并发需求灵活调整计算资源

提高了系统的容错性，存储和计算部分不会相互影响，便于快速修复

系统能够更高效、稳定地运行

提升系统的资源弹性扩展能力与存储效率，以及增强系统的容错能力与故障隔离性

系统资源与业务需求的不匹配

峰谷式数据请求

存储层用分布式存储集群，需要扩容时直接加存储节点

计算层则用容器编排工具，平时保持少量节点运行

系统稳定性

定采用华为云MRS + OBS 存算分离架构

存储层基于华为云OBS 构建独立存储集群，替代传统本地存储，保障数据完整性

计算层依托MRS 搭建无状态计算集群，核心任务部署为MRS 的Spark 作业， 所有任务所需数据均从OBS 实时拉取， 不依赖本地存储

MRS的ResourceManager 组件实时监控计算节点状态，当计算节点故障宕机时，负载均衡机制会自动将任务重调度至其他健康节点，且任务进度可基于OBS 中的中间数据无缝衔接。

企业集成

作为整合企业内不同应用系统、数据源的解决方案

核心依托消息中间件、企业服务总线（ESB）、API 网关、数据标准化及安全控制等技术特性实现数据共享与业务流程协同

多部门业务流程，传统架构下各系统孤立运行

数据流通不畅、一致性差

业务接口不互通使得流程需人工衔接，效率低下

通过数据标准化统一各类数据的格式与编码规则，消除数据异构问题

借助API 网关实现各系统数据访问的统一管控，搭配安全控制机制保障数据安全

依托ESB 整合业务接口，构建跨部门协同链路

企业集成平台的数据集成技术和业务流程集成技术在系统中的应用

企业集成平台的数据集成技术可实现系统多源异构数据的整合与数据一致性保障

用ETL 工具定时抽取各系统数据,转换为关系型数据结构或用Python 脚本标准化字段映射
基于转换后的数据，构建统一数据模型
数据入库前设置校验机制
企业集成平台的业务流程集成技术可实现系统跨部门业务流程的自动化流转与协同效率提升
跨部门流程衔接的低效
借助企业集成平台其内置的BPMN 流程引擎对流程进行梳理与建模，将流程拆解为标准化节点
跨部门任务流转从过去的“人工传递+口头确认”转变为“系统自动推送+待办跟踪”
避免了漏传、迟滞等问题。原本需要人工介入的操作也大幅减少，全流程的协同效率得到显著提升。

基于架构的软件设计（ABSD）

基于架构的软件设计方法（ABSD）总共需要经历六个主要的开发阶段
架构需求阶段
设计阶段
架构文档化阶段
架构复审阶段
实现阶段
演化阶段
基于架构的软件设计方法通过组件解耦与接口标准化，能够有效提升系统的可维护性与可扩展性
统长期迭代后的模块耦合问题
拆分为独立组件
接口标准化
通过统一的访问协议屏蔽底层数据库差异
组件间仅通过接口交互，禁止直接调用内部方法或共享数据结构，避免了跨模块的故障风险
基于架构的软件设计方法通过架构先行策略规范系统开发流程并提升团队协作效率
得返工修改，进度严重滞后
项目初期，梳理系统整体流程，明确规定了各团队的开发边界
开发阶段，各团队就照着架构文档并行开发

软件可靠性

软件可靠性是指软件在规定条件和时间内，完成规定功能的能力
采用了容错设计
数据存储方面，设置了多副本机制
数据的完整性和可用性
采用了故障检测与恢复技术，系统实时监测自身运行状态
发现异常，能自动进行恢复操作，减少系统停机时间
软件架构设计上，采用模块化设计
降低模块间的耦合度，某个模块出现故障时，不会影响其他模块
提高了系统的稳定性和抗干扰能力，确保系统能够准确、高效地完成工作
多副本容错设计和模块化架构设计这两个关键方面，探讨它们在保障系统数据完整性、持续可用性以及提升故障隔离能力和系统稳定性等方面所发挥的重要作用 多副本容错设计可保障软考阅卷系统的数据完整性与持续可用性
"通过数据冗余和故障容错机制解决问题"
“1 主2 从”的多副本存储方案
加入了健康检测机制
"比对三个副本的哈希值，确保数据一致"
模块化架构设计可提升软考阅卷系统的故障隔离能力与系统稳定性
单体架构，功能高度耦合

从模块化划分、低耦合设计和接口标准化三方面重构
拆分独立模块
配独立数据库和环境
低耦合设计
禁止模块间直接访问内部数据
统一通过标准化REST 接口交互
"接口明确输入输出格式"
每个模块部署健康检查，异常时自动隔离

安全架构设计

数据的保密性、完整性和可用性至关重要，因此采用科学的系统安全架构设计方案
"系统安全架构设计是构建安全可靠系统的基础，它从整体层面规划和布局系统的安全策略、
技术和措施"
保障考生信息和答卷数据不被非法获取和篡改
采用访问控制技术
数据加密技术也是关键
存储和传输中数据进行加密处理
"入侵检测与
防范系统能实时监测系统中的异常活动，及时发现并阻止潜在的攻击行为"
将这些安全技术和策略有机结合，构建了多层次的安全防护体系，确保系统的稳定运行和数据安全
围绕访问控制与数据加密技术以及多层次安全防护体系这两个关键方面展开详细阐述
访问控制与数据加密技术是保障软考阅卷系统数据保密性与完整性的核心技术手段
等敏感信息泄露或被篡改引发风险
权限管理
加密防护上，存储层改用国密SM4 算法对考生敏感字段和答卷内容加密
多层次安全防护体系的构建是软考阅卷系统实现安全可靠运行的架构保障
整合网络流量监控与主机异常行为检测能力，实时采集系统日志、用户操作轨迹和网络访问特征
基于历史数据建立异常行为基线
结合最小权限原则的访问控制机制，严格限制不同角色的操作范围
保障了系统持续稳定运行，更通过可追溯的安全机制