



智慧幕墙：系统架构与服务器运维

Intelligent Curtain Wall: System Architecture and Server
Operations and Maintenance

计算机科学与技术学院

林继申、刘淑仪、中谷天音

2024年11月13日





Table of Contents

1 项目目标

► 项目目标

项目存在问题

用户需求与系统需求

项目核心目标

► 项目范围与约束

项目范围

项目约束

► 用户参与 / 过程模型

用户参与过程（系统开发流程）

敏捷 Scrum 框架

► 交流问题

► 获取方法



项目存在问题

Project Issues

1. 环境依赖问题

- 各开发环境不一致，导致代码在不同环境下表现不一。
- 开发人员常遇到“本地测试没问题，但在他人电脑或部署环境中出错”的问题。

2. 开发效率低下

- 缺乏规范的 Git 操作流程，代码冲突频繁。
- 没有使用分支、Pull Request 等最佳实践，导致代码管理混乱。

3. 手动部署耗时且易出错

- 部署过程依赖人工操作，需要手动拉取代码、配置环境并部署。
- 部署流程繁琐且容易出错，部署滞后，开发完成的代码无法及时上线。

项目必要性

- 解决环境依赖问题，提升开发稳定性
- 提升团队协作效率，减少代码冲突
- 部署自动化，缩短交付周期
- 支持快速迭代与反馈，及时响应需求变化



用户需求与系统需求

User Requirements and System Requirements

用户需求：

1. 环境一致性

- 用户痛点：环境不一致导致的运行错误。
- 示例问题：本地测试通过，但在其他环境中出现问题。

2. 高效协作

- 用户痛点：开发过程中频繁发生代码冲突，影响进度。
- 示例问题：缺乏规范化的 Git 操作流程和分支管理。

3. 快速部署与更新

- 用户痛点：手动部署耗时且易出错，影响功能上线时间。
- 需求期望：希望通过自动化部署实现快速上线。

4. 快速迭代与反馈

- 用户痛点：难以及时响应需求的变化，更新上线周期过长。
- 需求期望：希望通过敏捷方法快速迭代。

系统需求：

1. 技术需求

- 一致的环境配置
- 自动化部署

2. 代码管理需求

- 高效的版本控制

3. 系统集成需求

- 模块化系统架构

4. 性能需求

- 快速部署与更新

5. 团队协作需求

- 敏捷开发支持



项目核心目标

Project Core Objectives

1. 解决环境依赖问题
 - 引入 Docker 技术，确保开发、测试、生产环境一致。
 - 使用 Docker Compose 进行统一环境配置，消除“环境不一致”导致的问题。
2. 提升开发效率
 - 通过 Git 的 Submodule 机制和最佳实践，实现代码独立管理与高效集成。
 - 使用分支管理、PR 流程等，降低代码冲突，提高协作效率。
3. 实现自动化部署
 - 引入 CI/CD 流水线，自动化构建、测试、部署流程。
 - 服务器定时拉取最新镜像并自动部署，确保开发完成的功能能快速上线。
4. 加速项目迭代
 - 缩短从开发到上线的周期，快速验证和反馈。
 - 通过高效的迭代流程，保证产品质量和交付速度。



Table of Contents

2 项目范围与约束

► 项目目标

项目存在问题

用户需求与系统需求

项目核心目标

► 项目范围与约束

项目范围

项目约束

► 用户参与 / 过程模型

用户参与过程（系统开发流程）

敏捷 Scrum 框架

► 交流问题

► 获取方法



项目范围

Project Scope

1. 技术栈范围

- 各子系统的开发均基于 Docker 技术，提供一致的开发环境。
- 使用 Git 的 Submodule 机制，实现代码的独立管理与集成。
- 集成 CI/CD 工具（GitHub Actions）用于自动化构建与部署。

2. 团队管理

- 36 人的团队分为 10 组，各自负责不同子系统的开发。
- 每组拥有独立的 Git 仓库，并遵循最佳 git 实践（分支管理、PR 流程等）。

3. 子系统与集成

- 项目包含 11 个子系统，每个子系统由不同小组独立开发。
- 使用 Docker Compose 进行 11 个子系统的集成与统一部署。

4. 部署与生产环境

- 部署环境采用 Docker Compose，确保生产环境与开发环境一致。
- 服务器自动定时从 Docker Hub 拉取最新镜像，更新生产环境。



GitHub Actions



项目约束

Project Constraints

1. 技术约束

- 所有开发必须基于 Docker 进行，确保环境一致性。
- 代码管理必须严格遵循 Git 最佳实践，包括分支策略、PR 审核等。

2. 时间约束

- 每半小时服务器自动拉取最新镜像进行部署。
- 项目整体需在既定时间内完成，并且各小组的子系统需要定期交付版本。

3. 资源约束

- CI/CD 流水线和 Docker Compose 部署需要占用服务器资源，需合理配置服务器。
- 团队内的沟通与协作需要配备相应的工具（如微信、飞书、GitHub 等）以提高效率。

4. 人员约束

- 每组需要保证独立性，但必须定期将成果集成到主仓库。
- 需确保团队成员熟悉 Git、Docker、CI/CD 工具等技术，以避免因技术门槛导致的延误。



Table of Contents

3 用户参与 / 过程模型

► 项目目标

项目存在问题

用户需求与系统需求

项目核心目标

► 项目范围与约束

项目范围

项目约束

► 用户参与 / 过程模型

用户参与过程（系统开发流程）

敏捷 Scrum 框架

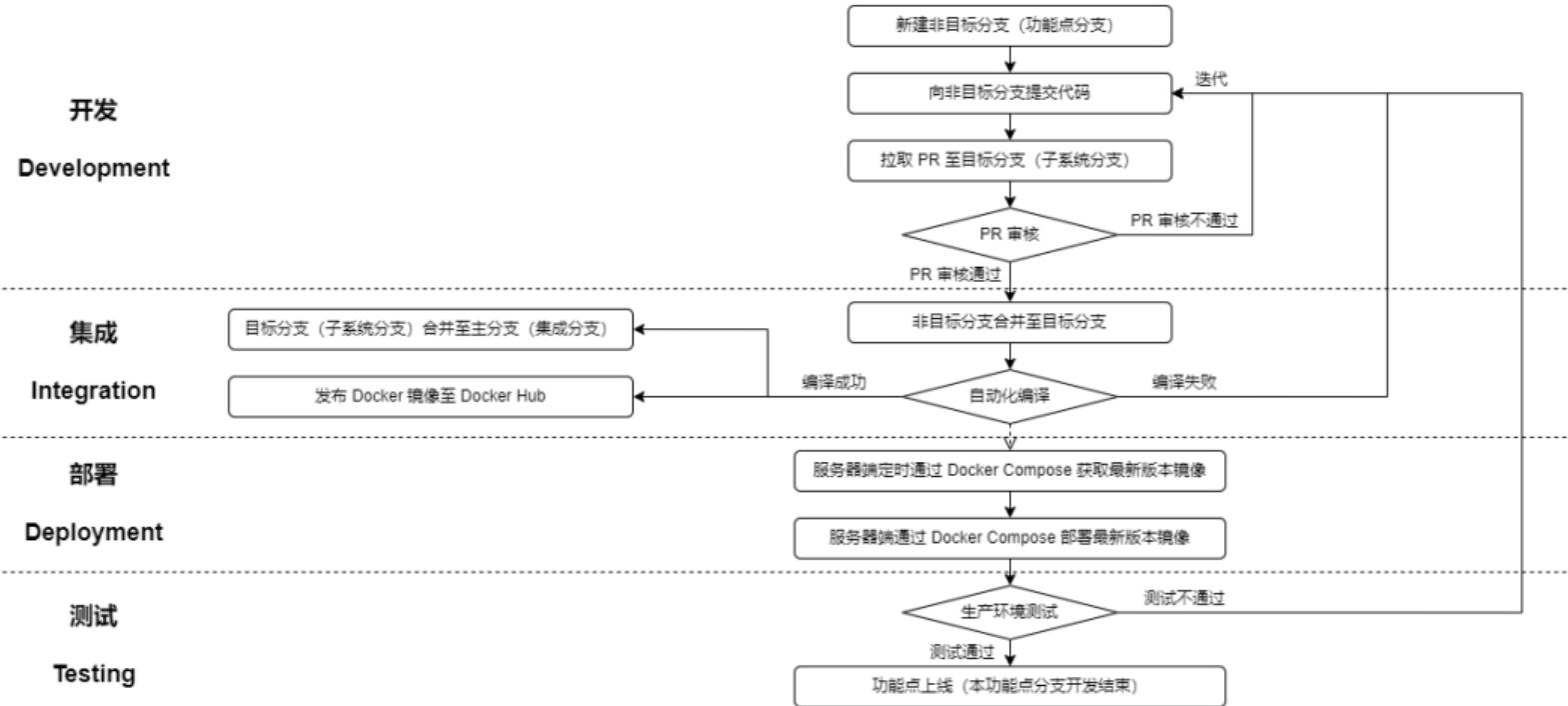
► 交流问题

► 获取方法



用户参与过程（系统开发流程）

User Involvement Process (System Development Lifecycle)





敏捷 Scrum 框架

Agile Scrum Framework

- 敏捷 Scrum 优势
- Scrum 开发流程
- Scrum 项目进展



同濟大學
TONGJI UNIVERSITY

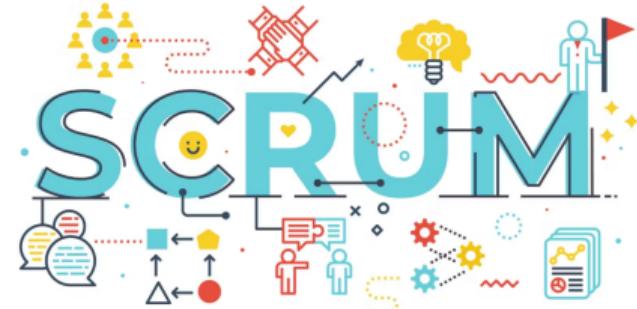


敏捷 Scrum 优势

Advantages of Agile Scrum

Scrum 是一种敏捷开发方法论，专注于通过小团队合作以及快速频繁的迭代来管理和完成复杂的软件和非软件项目。Scrum 的优势如下：

- 灵活性与适应性：Scrum 允许在项目开发过程中快速适应变化，这对于需求不断变化的项目环境尤为重要。
- 增强透明度：通过日常站会和冲刺回顾，项目进度和障碍对所有团队成员都是透明的，增强了团队的沟通和协作。
- 持续改进：每次冲刺结束时的回顾会帮助团队评估其效率和工作方式，不断寻找改进的机会。
- 快速迭代：短周期的工作冲刺使团队能够快速交付产品特性，及时响应市场和客户的反馈。





Scrum 开发流程

Scrum Development Process

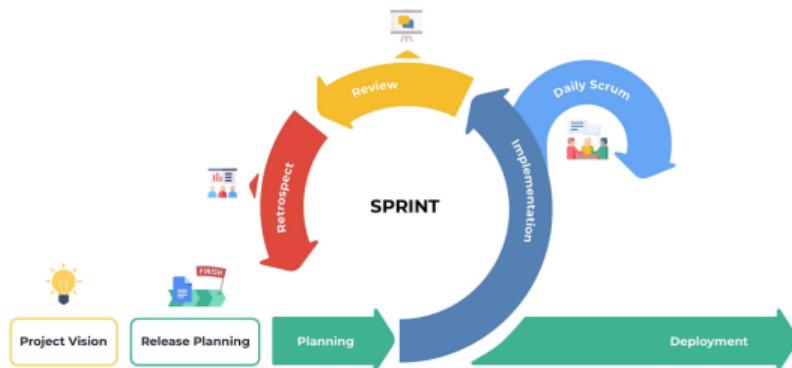
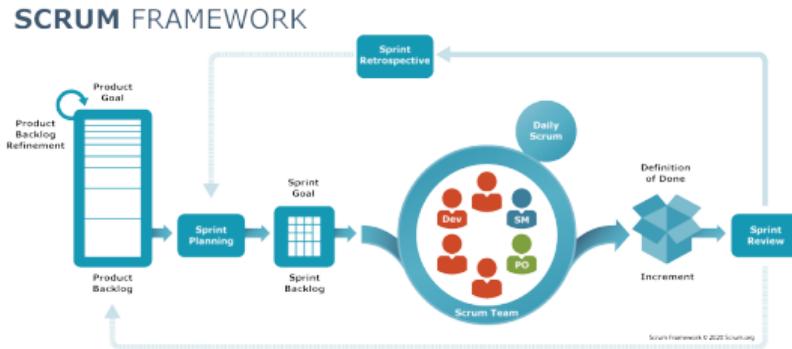
考虑到本项目涉及多个子系统和一个由 36 人组成的较大团队，使用 Scrum 框架可以有效地管理多元化的需求和快速变化的开发环境。

Scrum 实施步骤

Sprint 计划会议 → 每日站会 → 持续集成与部署
→ Sprint 评审会 → Sprint 回顾会

特定优化措施

1. 使用 Docker 和 CI/CD 可以极大地支持 Scrum 的快速迭代特性，通过自动化测试和部署，缩短开发周期，减少人工错误。
2. 通过 Git 的 Submodule 机制，各子系统独立但集成，使得 Scrum 团队能够更有效地协作，同时保持代码的整洁和管理的可控性。





Scrum 项目进展

Scrum Project Progress

已完成

环境依赖问题解决：使用 Docker 和 Docker Compose，确保开发、测试、生产环境一致。

代码管理优化：通过 Git Submodule 机制，建立各子系统独立的仓库管理，同时引入分支管理、Pull Request 等最佳实践。

自动化部署实现：配置 CI/CD 流水线，代码提交后自动触发构建、测试和部署。服务器定时更新镜像并重新部署。

进展中

快速迭代流程建立：每周进行 Sprint 规划、迭代开发和回顾，不断优化开发流程。

收集小组反馈，完善系统架构：持续收集各个小组的意见和建议，针对系统架构和开发流程进行优化，以适应项目的复杂性和团队协作需求。

待完成

项目中的图片数据集通过阿里云 OSS 对象存储进行管理和维护，确保数据存储高效、安全、可扩展。各小组可以便捷地访问和更新数据集，实现开发与测试资源的统一管理。



Table of Contents

4 交流问题

► 项目目标

项目存在问题

用户需求与系统需求

项目核心目标

► 项目范围与约束

项目范围

项目约束

► 用户参与 / 过程模型

用户参与过程（系统开发流程）

敏捷 Scrum 框架

► 交流问题

► 获取方法



交流问题

Communication Issues

- 确保团队成员之间的信息流通顺畅，尤其是在敏捷 Scrum 框架下，日常站会和 Sprint 回顾要确保及时共享进展与问题。
- 建立清晰的沟通渠道，如微信群或飞书等工具，避免信息滞后或误解。
- 处理反馈时要注重倾听和反馈的及时性，避免因沟通不畅导致项目进度或质量的影响。
- 随着项目的推进，技术架构、开发流程、CI/CD 配置等方面文档及时更新，确保每个团队成员都能访问到最新的工作信息和系统架构设计。这对于跨团队合作尤为重要。
 - 《后端技术选型——Docker Compose 技术栈》
 - 《Git 版本控制系统基本操作指南》
 - 《智慧幕墙项目后端应用程序集成 Git 开发规范与流程》



Table of Contents

5 获取方法

► 项目目标

项目存在问题

用户需求与系统需求

项目核心目标

► 项目范围与约束

项目范围

项目约束

► 用户参与 / 过程模型

用户参与过程（系统开发流程）

敏捷 Scrum 框架

► 交流问题

► 获取方法



获取方法

Acquisition Method

项目的获取方法如下：

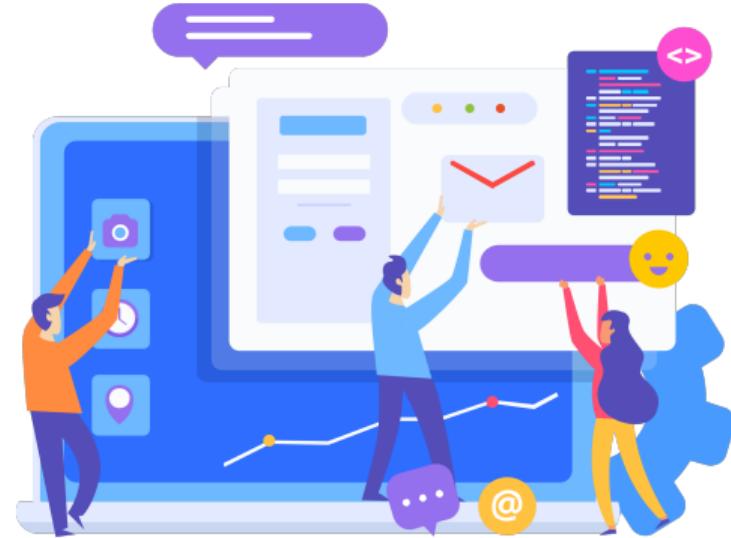
1. 科研项目需求

- 项目来源于科研任务或学术研究的需求。

2. 开发团队内部反馈

- 项目由开发团队在日常工作中发现的问题或改进点推动立项。

根据项目需求来源，可以灵活调整项目目标和实施策略，确保科研和开发目标都能高效实现。





智慧幕墙：系统架构与服务器运维

Intelligent Curtain Wall: System Architecture and Server Operations
and Maintenance

Thank you for your listening!

林继申、刘淑仪、中谷天音 2024年11月13日