**1 功能实现/发布版本**

1.1发布版本的标准制定

目的：定义清晰、可度量的成功发布的标准。考虑功能完成度、bug容忍阈值、性能基准和设计规范的符合度等方面。

流程：建立一个涉及关键利益相关者的审核过程，确保所有标准都得到满足。

1.2发布版本的测试过程/测试结果

测试策略：描述整体测试策略，包括单元测试、集成测试和系统测试。

结果报告：提供详尽的测试结果，包括通过率、发现的问题以及可能的风险。

1.3版本代码管理

版本控制：使用适当的版本控制系统（Git），确保代码的版本和分支管理符合最佳实践。

文档化：确保所有代码更改都有清晰的记录和适当的文档支持。

1.4发布版本的演示

演示目的：展示新功能、修复的问题以及可能的性能改进。

参与者：确保所有相关利益相关者（仓库经理，仓库管理员，员工等）都参与演示，以便收集反馈和建议。

**2 协作**

2.1 Git 开发分支的活动

分支策略：定义分支管理策略，如功能分支、修复分支等，确保代码整洁并易于管理。

合并规则：设立严格的代码审查和合并流程，确保主分支的稳定性。

2.2 自动编译打包-自动测试

自动化流程：实施持续集成（CI）流程，自动化编译、打包和测试。

监控与报告：使用工具监控构建状态和自动测试结果，及时反馈问题。

**3 项目管理**

3.1 Sprint 燃尽跟踪

监控工具：使用燃尽图等工具来追踪Sprint的进度，确保团队能按时完成任务。

调整策略：根据燃尽图的反馈，适时调整开发策略和资源分配。

3.2 小组绩效统计

绩效指标：确定关键绩效指标（KPI），如完成任务的数量、代码质量、团队合作等。

反馈与改进：定期回顾绩效数据，与团队讨论改进方案。

**4 功能实现和版本发布的细化**

4.1 发布版本的标准制定

功能完整性：确保所有预定功能，如自动货物分类、存储、检索和库存管理，都已完全实现并通过初步测试。

性能标准：设定系统响应时间、处理速度和错误率的具体标准。例如，系统响应时间不超过2秒，错误率低于0.1%等。

用户接口：界面应清晰易用，支持多语言，特别是操作界面要符合仓库作业人员的实际操作习惯。

安全性和合规性：确保系统符合行业安全标准和数据保护法规，如GDPR或PCI DSS。

4.2 发布版本的测试过程/测试结果

测试计划：制定详细的测试计划，包括模块测试、集成测试和系统测试，以及特定的性能和压力测试。

自动化测试：开发自动化测试脚本，模拟不同操作场景，验证系统功能和性能。

实地测试：在实际仓库环境中进行测试，确保系统与现场设备如自动叉车、传送带等无缝集成。

用户验收测试：邀请最终用户参与测试，收集反馈，确保系统满足实际操作需求。

4.3 版本代码管理

代码版本控制：使用Git等工具管理所有代码和文档，确保每个版本都可追溯和管理。

分支策略：实施严格的分支管理策略，如功能分支、发布分支和修复分支，保证开发流程的清晰和高效。

代码审查：定期进行代码审查，提高代码质量，减少bug率。

4.4 发布版本的演示

内部演示：在内部团队中进行全面演示，验证所有功能，性能指标和用户界面。

客户演示：向客户展示系统的核心功能和操作流程，收集客户的意见和建议。

演示反馈整合：将演示中收集到的反馈整合进系统的优化和调整中。