软件工程中敏捷开发的实践

冯鲲
云南开放大学（云南国防工业职业技术学院）
DOI:10.12238/acair.v3i3.15559

[摘　要] 敏捷开发模型是以人为中心的开发模式,在当前需求不断变更的情况下能够快速产出可用的产品,并且对过程中的风险有很好的把控。本文就敏捷开发在软件工程中核心实践与应用作相关探讨,对关键技术实现、敏捷实施框架及典型项目案例进行分析,分析其在提高交付效率跟响应需求改变方面的价值.主要研究版本控制及自动化测试在C、C++、.NET与Python项目中的特定应用方案,同时纳入桌面应用、Web服务及企业系统的案例分析,探析迭代开发、架构进化及团队协作的理想模式,借助小步迈进与持续反馈,敏捷方法极大优化了软件开发流程。
[关键词] 敏捷开发；持续集成；自动化测试；版本控制；Scrum；DevOps
中图分类号：TP311.5　**文献标识码**：A

The practice of Agile development in Software Engineering

Kun Feng
Yunnan Open University (Yunnan Vocational and Technical College of National Defense Industry)
[Abstract] The agile development model is a human-centered development mode. It can quickly produce usable products under the current constantly changing requirements and has a good control over the risks in the process. This article conducts relevant discussions on the core practices and applications of agile development in software engineering, analyzes the realization of key technologies, agile implementation frameworks, and typical project cases, and explores its value in improving delivery efficiency and responding to changes in requirements. The main research focuses on the specific application solutions of version control and automated testing in C, C++,.NET and Python projects. Meanwhile, case analyses of desktop applications, Web services and enterprise systems are included to explore the ideal models of iterative development, architectural evolution and team collaboration. With the help of small steps and continuous feedback, the agile method has greatly optimized the software development process.
[Key words] Agile Development Continuous integration Automated testing Version control Scrum DevOps

引言

伴随着我国营商环境的改善和科技的进步,互联网和信息技术产业发展蓬勃,软件项目规模迅速扩大。在此背景下,软件的开发过程中出现了需求多变、项目工期紧张、对资源高度依赖等现象,对企业的软件开发项目管理水平提出了巨大的挑战。按时且高质量地完成软件项目的开发与交付能够帮助企业更快地抢占市场,把握先机。许多企业在资源受限的情况下,通过传统的软件项目进度管理方法难以很好地应对市场与用户多变的需求,致使项目实际进度落后于计划,在市场竞争中处于落后地位[1]。目前,许多软件企业放弃了传统瀑布式的开发模式,选择了轻量级的、适应变化的、迭代交付的敏捷开发模式。本文旨在系统分析敏捷开发在软件工程中的具体实践路径与关键技术实现,结合不同技术栈的实际案例,为高效实施敏捷提供参考。

1 敏捷开发在软件工程中的实施框架

1.1敏捷开发核心流程构建

敏捷开发的核心流程围绕迭代式增量交付和价值驱动展开。Scrum框架搭建起被广泛采用的基础架构,这个框架把开发作业拆解成固定时间的冲刺阶段[2]。各个冲刺均由冲刺规划会议起始,团队成员于此次会议承诺去完成从产品待办事项列表选出的高优先级用户故事。开发团队在冲刺阶段的每天都开站会,成员迅速同步进展状态、识别问题障碍,而后调整当日工作计划,冲刺收尾阶段召开评审会议,团队向利益相关者展示已完成的增量成果,再收集相关反馈,回顾会议让团队反思流程、技术实践以及协作,持续鉴别改进契机。

1.2主流编程语言的敏捷实践适配

不同编程语言的技术特性显著影响着团队开展敏捷实践的具体做法.强类型静态语言如Java和C,拥有强大的IDE支持以及编译时检查手段,有效强化了大规模代码重构的安全性及执行效能,这在敏捷流程频繁的代码调整方面意义重大.凭借这些语言成熟的框架及工具链,可顺利构建自动化构建与测试流水线,如Python和JavaScript这类动态语言,凭借其解释执行的特性与灵活的语法,自然契合快速原型及测试驱动开发,开发者可迅速完成小块功能的编写与验证。

2 敏捷开发关键技术实现

2.1版本控制与协同开发

2.1.1 Git分支策略在C++项目中的应用

作为分布式版本控制体系的Git,其灵活分支模型为C++项目实现并行开发与高效协作提供可靠基础。用于管理复杂C++桌面应用程序版本演进的常用策略是Gitflow工作流,这策略清晰地对分支类型及生命周期加以定义,`main`分支一贯代表生产环境的稳定局面,每一项新功能开发都于独立的`feature`分支实施。这些分支会从最新的`develop`分支分离得到,`develop`分支汇集所有完成并经测试合格的功能,充当起下一个发布版本的基础[3]。

2.1.2凭借Azure DevOps的C团队协作方案

AzureDevOps推出了一套集成化服务,完全支持C团队实施敏捷开发。AzureBoards担当工作项跟踪的核心角色,掌控产品待办事项、用户故事、任务与各类缺陷。为适应团队流程,工作项类型及状态可高度定制,工作流程借助看板视图实现可视化,以限制在制品数量优化流动效率。冲刺规划的会议当中,团队把用户故事切割成具体任务,再分配到AzureBoards的迭代路径,Azure Repos呈献强大的Git仓库托管服务资源,能支持前文提及的Git分支策略实施。

2.2自动化测试体系构建

2.2.1 Python单元测试框架的敏捷集成

Python生态系统拥有成熟的单元测试框架,如`unittest`和`pytest`,这些框架无缝集成到敏捷开发流程中。`pytest`借助简洁的语法、有力的夹具系统和丰富的插件生态,成了优选方案,开发人员编写单元测试的时间,既可能是在功能代码编写之前,也可能是同步期间,践行测试驱动开发策略,或采用测试伴行开发。`pytest`断言机制让测试编写变得高效且直观,共享夹具是通过`c.y`文件来进行定义的,诸如数据库连接还有模拟对象,维持测试的独立性及可重复性。

2.2.2持续集成环境下的测试覆盖率保障

在持续集成环境中保障高测试覆盖率是敏捷项目质量守护的关键环节[4]。实现这一目标依靠工具链的集成,构建过程里有一个专门步骤,执行运行测试套件操作并同步收集代码覆盖率数据。在既有.NET项目中,以流行之姿存在的覆盖率收集器Coverlet,同`dotnet test`命令完成集成,构建覆盖率综合报告。在Python项目里,常借助`pytest-cov`插件来收集覆盖率,所收集的原始覆盖率数据需处理后转化成可读报告,OpenCover或ReportGenerator在.NET中应用频繁,于Python中,`c.y`既能够单独使用,也可结合别的工具。

3 典型项目中的敏捷实践案例分析

3.1桌面应用开发项目(C++)

3.1.1迭代过程中架构演进路径

一个大型C++桌面图像处理软件的开发团队采用Scrum框架进行迭代开发,其架构经历了显著的演进。项目起步阶段选用较为简易的分层架构,以实现核心功能的快速交付,伴同迭代深化及功能数量增长,最初的架构逐步显现出扩展性、维护性方面的瓶颈。团队针对特定冲刺专门安排了架构改进相关任务。首次重大的演进是在要支持插件机制的时候发生的,团队采用模块化设计模式,制定清晰可辨的插件接口,把可扩展功能与核心功能加以分离,运用动态库实现插件热加载能力。

后续的迭代开发工作持续受益于这次关键的架构转型。开发团队在规划新的图像处理滤镜或文件格式支持功能时,能够将这些新增特性直接打包为独立的插件模块。插件开发人员遵循团队制定的统一接口规范,专注于实现特定功能逻辑,无需深入理解整个应用的核心代码结构。

3.1.2重构与性能优化的平衡策略

在C++桌面应用项目中,平衡新功能交付、重构技术债务和性能优化是持续的挑战。决策由团队采用的基于价值优先级策略指引,产品负责人把用户需求及商业价值作为依据来确定功能优先级。技术债务被登记录入专门的技术待办事项列表,技术负责人评判其带来的影响及相应修复成本,只要债务对系统稳定性、可维护性造成严重影响,或者阻碍新功能开发,便有高优先级。

团队在每次冲刺规划会议期间都会预留一小部分时间专门处理高优先级的技术债务或性能瓶颈问题。开发人员发现某个图像处理算法执行速度变慢时,会首先分析性能下降的根本原因并评估优化所需工作量。产品负责人结合用户反馈和业务目标,判断是否立即优化该算法还是将其放入技术待办事项稍后处理。对于需要重构的代码区域,开发小组通常会先编写或补充单元测试以保证重构行为不会破坏现有功能。

3.2 Web服务开发项目(Python+Django)

3.2.1用户故事拆解与API快速迭代

一个基于Python和Django框架的RESTfulAPI服务项目采用用户故事驱动开发。产品负责人与开发团队协同发力紧密配合,把业务需求细拆分,转变为符合INVEST原则的小型用户故事,各用户故事清晰地表达出特定角色所需功能及其具有的价值。后端API故事聚焦特定资源端点操作。团队拆分故事时采取“纵向切片”样式,保障单个故事可交付从数据库模型到API端点这一完整、具备可测试性的价值单元,“作为用户需查看个人资料详情”这一点由`GET/api/users/{id}`来实现,依托Django REST Framework的序列化器、视图集和路由器等组件可迅速构建API。

这种以小型、完整用户故事为中心的开发方式极大提升了团队的迭代速度和交付质量。开发人员在接到一个用户故事卡

片后,能够快速明确需要实现的数据库模型字段、序列化器逻辑、视图处理函数以及对应的URL路由配置。团队成员在实现故事过程中频繁进行代码审查和集成测试,确保新增的API端点不仅功能正确而且符合项目整体的代码规范和安全要求。

3.2.2微服务架构下的敏捷部署方案

随着业务复杂度提升,单体Django应用被分割为多个拥有松耦合关系的Python微服务[5]。微服务架构赋予团队独立开展开发、部署与扩展工作的便利,却也引入了运维工作的复杂性,通过采用容器化以及基础设施即代码,团队达成敏捷部署。各个微服务都有其独立的代码仓库以及CI/CD流水线,开发人员把服务代码修改好后提交,驱动专属流水线启动,进行流水线执行。主要步骤有：运行单元和集成测试用例、构建Docker镜体、把镜像推送到私有库。

每个微服务的专属CI/CD流水线在代码提交后自动触发执行流程,流水线成功完成所有步骤后会自动将新构建的Docker镜像部署到预发布或生产环境的对应容器集群中。这种自动化部署机制显著缩短了从代码提交到功能上线的周期,使得团队能够实现一天内多次安全可靠的部署。开发人员可以专注于自己负责的微服务功能开发,无需深入了解其他服务的部署细节。

3.3企业应用开发项目(C/.NET)

3.3.1领域驱动设计与敏捷开发结合

一个大型C/.NET企业级订单管理系统项目成功结合领域驱动设计和敏捷开发实践。项目启动的初始阶段,开发团队与领域专家借由事件风暴工作坊深度探索业务领域,识别核心域、子域、限界上下文、聚合根、实体、值对象以及领域事件。这些建模成就构建起领域模型的初步基础,限界上下文被团队用作微服务划分的自然边界。处在Scrum迭代里,用户故事的撰写紧紧围绕着领域概念与通用语言展开。例如,“订单提交”故事涉及`Order`聚合根的状态转换和领域事件发布。

在具体的开发迭代过程中,开发人员实现用户故事时会严格遵循前期建立的领域模型和通用语言。团队成员在处理“订单提交”故事时,清晰地知道需要修改Order聚合根相关的状态属性,并在状态变化满足业务规则时触发OrderSubmitted领域事件。这种基于领域模型的开发方式确保了代码实现与业务概念的高度一致性,极大减少了因理解偏差导致的功能缺陷。领域事件驱动架构的应用使得系统各部分能够松散耦合,例如库存管理微服务通过监听OrderSubmitted事件来执行相应的库存扣减操作。

3.3.2遗留系统改造中的敏捷过渡方案

将庞大的C/.NETWinForms遗留单体应用现代化是一项艰巨任务,团队采用“绞杀者模式”结合敏捷方法实现渐进式改造。改造目标是将应用逐步迁移至现代化的Web架构。团队首先在遗留系统外围构建新的API网关层。然后,识别出高价值、边界相对清晰的业务功能模块作为试点。每个选定的功能模块被重构成独立的领域微服务,使用ASP.NETCoreWebAPI实现,部署在容器中。遗留单体应用通过API网关与新服务通信,取代原内部调用。前端界面逐步从WinForms迁移到BlazorWebAssembly,通过网关调用后端服务。

团队在每次冲刺中选择一个或几个小型、独立的业务功能模块进行剥离和重构。被选中的功能模块在遗留单体应用中被标记为待替换区域,同时团队并行开发其对应的现代化微服务版本。新的微服务开发完成后,通过配置API网关将指向该功能的流量从遗留应用路由到新服务。遗留应用中原有的功能模块代码暂时保留但不再维护,最终在确认新服务稳定运行后将其移除。

4 总结

敏捷开发通过迭代交付、紧密协作和持续反馈,有效应对软件工程中的需求多变性与复杂性。其成功实践依赖于清晰的实施框架、自动化技术栈的支撑以及针对项目特点的灵活应用。版本控制、持续集成和自动化测试构成敏捷高效运作的技术基石。从C++桌面应用到PythonWeb服务再到C企业系统,敏捷原则指导架构演进、重构优化与遗留改造,持续提升软件价值交付能力与质量。敏捷开发已成为现代软件工程不可或缺的核心方法论。

[参考文献]

[1]连钰昕.敏捷开发模式在软件工程课程中的应用[J].现代职业教育,2025,(10):159-160.

[2]马艮娟,卜言彬.大数据背景下软件工程开发关键技术探析[J].数字技术与应用,2024,42(07):155-157.

[3]王斌.敏捷开发模式在软件工程项目中的应用[J].电子技术,2022,51(03):288-289.

[4]孙德刚.Scrum敏捷方法在软件工程实验框架设计开发中的应用研究[J].电子世界,2021,(09):172-173.

[5]姚立新,梁宏涛.敏捷开发软件模式初探[J].电子技术与软件工程,2013,(20):82-83.

作者简介：

冯鲲(1975--),男,汉族,云南宣威人,硕士研究生,工学硕士,讲师,软件开发、计算机教育、大数据技术,云南开放大学(云南国防工业职业技术学院),昆明市呈贡大学城云南开放大学传媒与信息工程学院。