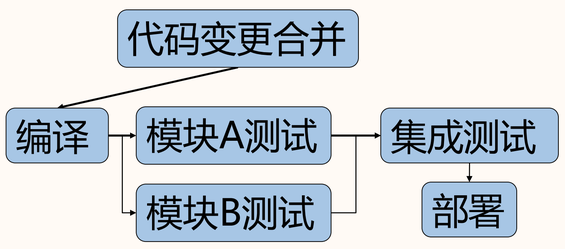
智能化持续集成与持续部署流水线

杨丰 朱家鑫 高楚舒 陈伟 魏峻
联系人：朱家鑫 zhujiaxin@otcaix.iscas.ac.cn

背景与问题

持续集成与持续部署（CI/CD）加速了软件系统与服务的演进速度，改善了用户体验。而快速迭代意味着更多的资源开销和对开发工作的干扰与影响，如何降低持续集成与部署的开销代价，同时达到预期的效果？

CI/CD流水线示例：



目标：快速推向用户

系统解决方案

模式驱动的流水线建模
CI/CD任务预规划

→

数据驱动的流水线调优

流水线度量
特征度量/集成

流水线决策
模型训练/预测

执行任务的动态选取与配置

→

流水线使能
流水线执行和动态调整

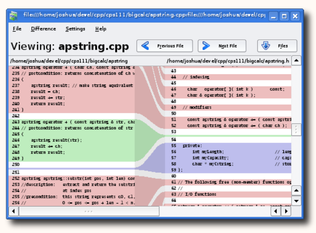
→

高效流水线：适时完成更新推送、及时发现冲突与bug、最小资源开销

关键技术—多维CI/CD流水线度量框架

基于开发逻辑、需求及相关数据理解、验证

开发/提交者
1. 开发协作经验
2. 开发协作能力



代码提交
1. 规模
2. 类型
3. 复杂度
4. 上下文

CI/CD任务
1. 执行的平均资源开销
2. 执行结果
3. 执行效果

关键技术—模式驱动的流水线建模

多源数据实证研究

Travis CI
Jenkins
circleci
CI/CD配置文件

TravisCI	658,000个
Jenkins	2,500个
CircleCI	1,400个

保留字使用及赋值分析：
1. 系统特性提取
2. 使用频率分析
3. 使用模式挖掘
4. 使用效果评估

流水线建模语言　流水线模板　流水线推荐

关键技术—数据驱动的流水线执行决策

基于回归分析发现关键特征

开发者特征
代码提交特征
任务特征

决策模型(分类器: 执行与否）
编译　静态检查
测试(多粒度)　部署

关键技术—全方位流水线使能技术

任务启停、中断与恢复　制品缓存
工作空间管理　执行单元管理

实施效果预估

对Ruby on Rails项目的模拟实验（基于历史CI/CD数据）显示：
上述决策技术可以减少50%以上非必要的集成任务，减少25%以上的资源消耗