# CCF-华为胡杨林基金

# 软件工程

# 2022 年度申报课题及说明

CCF-华为胡杨林基金技术委员会——软件工程领域组

二零二二 年 四 月

## 第一条 课题

本专题中课题主要为了持续提升相关软件工程技术的能力上界，针对业界典型问题和痛点，解决关键技术瓶颈或形成技术断裂点，大幅提升产品/平台/解决方案竞争力，创造产业价值，形成本研究计划的正循环。

## 第二条 基金池

2022 年度软件工程专题基金池共计 360-400 万人民币。

## 第三条 产业课题

单课题原则上资助额度为 40-60 万人民币，为期一年。提交成果中原则上需要包含源代码。

2022 年度拟资助 8 个课题：

1. 基于代码大数据分析的缺陷修复模式挖掘技术
2. 数据驱动的高效开源漏洞挖掘技术
3. Flaky 测试的根因分类定位方法
4. 基于多态失败日志的根因定位算法
5. 业务特性感知的资源调度技术
6. 面向研发数据的知识图谱实体关系识别和融合技术
7. 开放应用平台中代码缺陷检测技术
8. 面向联盟区块链的新型智能合约语言

# 课题一：基于代码大数据分析的缺陷修复模式挖掘技术

## 项目背景

静态检查工具在实际大规模代码上可能产生数以万计的告警，人工修复成本巨大。此外，在实际代码修复过程中，开发人员可能缺乏必要经验和知识，存在分析困难和不知如何解决的情况。英国剑桥大学 2013 年的研究表明，每年全球在修复代码缺陷和调试代码逻辑上要花费 1560 亿美金，占到编程人员所有工作时间的 50%。因此，需要探索具有一定自动化程度的缺陷修复技术方案。可以从代码、文档记录等软件数据中智能识别并挖掘各类漏洞和缺陷的修复模式，提供尽可能准确的修复建议甚至自动修复，提高软件系统的质量和安全性。

## 研究内容

面向 C/C++/Java 等常用编程语言，综合利用代码分析、数据挖掘和机器学习等技术，从漏洞或缺陷代码及历史修复数据中挖掘提炼各类缺陷修复模式。本项目拟探索通用的缺陷修复模式挖掘方法和技术，但在研究阶段，可以在某几种缺陷/漏洞类型上进行验证。具体缺陷或漏洞类型包括但不限于：内存泄漏、释放后使用、空指针解引用、开源三方库 API 使用不当、不可信数据校验缺失、函数返回值检查缺失等。具体可选合作方向有：

方向一：自动判别代码提交是否为缺陷修复，并识别修复的缺陷类型；在此基础上，从大量缺陷修复历史挖掘缺陷修复模式，构建缺陷修复模式库。

方向二：基于缺陷修复模式，孵化并开发缺陷修复技术或工具。

## 项目目标

方向一：缺陷模式挖掘

1. 缺陷模式挖掘算法的准确率不低于 70%，回收率不做强制要求；
2. 针对给定的缺陷类型，从开源工程中挖掘出不小于 200 个有效缺陷修复模式；

方向二：缺陷修复工具

1. 定义一种缺陷修复模式描述语言，并开发相应的缺陷修复执行引擎，能够自动匹配缺陷上下文并进行代码转换，生成修复代码；
2. 针对给定缺陷类型，可以人工归纳部分修复模式，对实际缺陷进行修复，准确率不低于 80%，召回率不低于 50%。

# 课题二：数据驱动的高效开源漏洞挖掘技术

**项目背景：**

现代软件开发大量复用开源软件，软件开发效率得到了极大提升，但与此同时，也会面临严峻的开源风险攻击。据《2021 年开源安全与风险分析报告》统计,平均每个代码仓含 158 个安全漏洞，84%的代码仓都至少存在 1 个安全漏洞。相比 20 年，21 年高风险漏洞环比增长了 11%。围绕开源三方库，快速感知开源安全漏洞并进行精准漏洞影响范围通知及安全响应，将成为未来商业产品的有力安全保障。

为支撑上述场景，本课题拟聚焦探索数据驱动的高效开源漏洞挖掘技术，从而快速、全面、精准地识别开源漏洞信息，从而帮助到漏洞的精准定位、通知、响应。本课题关注的漏洞范围包括：0-day 安全漏洞（尚未公开披露）的自动化挖掘、1-day 安全漏洞（已公开披露）的智能分析（含数据收集、影响范围定位等）。

**研究内容：**

面向 Java，Python 项目，利用软件分析和机器学习相结合的方式，完成 0-day 漏洞的自动挖掘；对于 1-day 漏洞，综合利用软件分析、自然语言处理、深度学习、数据挖掘等多种技术，自动化或半自动化方式智能定位出该漏洞所影响的具体开源软件版本。

0-day 漏洞自动挖掘技术可基于但不限于如下技术思路：

1. 基于历史提交代码的漏洞代码挖掘；
2. 基于 1-day 漏洞修复补丁的同源分析。

1-day 漏洞影响范围自动定位技术可基于但不限于如下技术思路：

1. 基于漏洞描述、软件描述等相关的文本信息，利用 NLP 的技术，实现所影响软件以及具体版本的自动推荐；
2. 基于修复补丁或修复代码片段，使用软件分析、深度学习等多种技术，定位出该漏洞所影响的软件以及具体版本。

**项目目标**

0-day 漏洞自动挖掘：

1. 漏洞挖掘算法的准确率 > 70%，回收率不做强制要求；
2. 针对给定的 top 1000 Java 开源项目和 top 1000 Python 项目，定向挖掘出的 0-day 漏洞数量不少于 100 个。

1-day 漏洞影响范围的自动定位：

在给定的测试集（已人工确认的漏洞及包版本映射关系数据）上进行验证，针对相关漏洞定位出的三方库的准确率高于 80%，覆盖率高于 80%；针对相关漏洞定位出的三方库具体版本的准确率高于 70%，覆盖率高于 70%。

# 课题三：Flaky 测试的根因分类定位方法

## 项目背景

Flaky Test 是指在同样的软件代码和环境配置下，得不到确定(有时成功、有时失败)的测试结果。

系统测试是软件开发与运维中的重要环节。然而，在实际生产中，存在 flaky 测试的问题，即测试结果发生失败-成功的跳转。Flaky 测试的结果难以复现，会耗费开发人员大量时间来定位失败原因，却不能对提升产品质量产生增益，也会对生产系统引入风险。定位系统级 flaky 测试的常用方法是重跑。然而相较于单元测试，系统测试(黑盒)重跑会耗费大量计算资源及时间。因此，需要开发针对系统级 flaky 测试的自动化检测、分类、定位技术，以及 flaky 修复技术，以降低其负面影响。

## 研究内容

根据调研,业务开发过程中存在大量的系统级 flaky 测试日志,并且目前业务场景中尚无自动探测系统级 flaky 测试的方案。当 flaky 测试发生时，测试人员检测 flaky 测试的常用方法是重跑验证，然而重跑会耗费大量计算资源以及执行时间。造成 flaky 的根因主要是用例执行顺序、并发同步、环境原因等常见问题。针对上述业务场景痛点,本课题旨在通过综合利用测试用例日志、测试脚本代码、用例执行结果、被测环境信息等多维数据，研究和开发 flaky 测试的检测及根因分类定位方法。

## 项目目标

目标一：研发一套基于机器学习,而非重跑验证的 flaky 自动化探测技术(auto-detection).可结合测试日志脚本代码、测试坏味道、代码修改记录和用例执行记录等信息做出评估。

目标二：针对上述常见的 flaky 根因，研发 flaky 测试的根因分类检测技术(重点投入，准确率达到 90%以上)。

挑战目标：在完成目标二的基础上，根据 flaky 根因生成测试用例的修复建议或方案，提交算法设计文档，最好能在相关领域发表学术论文。

# 课题四：基于多态失败日志的根因定位算法

## 项目背景

现有软件失败日志定界分析方法通常都是通过专家预定义的经验模式或者将特征提取和深度学习分类器相结合进行失败原因的定界与定位。随着软件规模的持续增长以及产品迭代周期的不断加快，失败日志模式越来越复杂多样，类别树频繁变动，标签数据质量不佳，专家经验难以通过人工方式预先总结归纳，而现有分类器也有着鲁棒性不强的问题。

因此，亟需探索具有一定自动化程度的技术方案，通过测试日志、运行日志、环境日志、测试脚本、被测对象等软件失败相关数据的基础上探索改进的失败日志定界分析方法，通过自动化的手段提升失败定界与定位算法的鲁棒性、可解释性以及新问题的发现效率。

## 研究内容

每个测试用例在每次执行时会产生一组配套的测试日志、产品日志、trace 日志、环境日志等半结构化信息；这些日志信息是测试用例失败分类分析和定位分析的重要数据源，不同类型的日志中包含了各种不同阶段、不同形态的失败信息，如何对多态日志进行关联，从而实现全面、准确、有效地多态失败日志联合分析是当前的一个关键问题。

## 项目目标

目标一：如何有效提取多态失败日志特征

本问题旨在基于多态日志及其他软件数据（业务配置、环境配置、测试脚本等），利用并完善多态失败数据间的关联关系，提取同用例历史执行轮次多源信息的特征，提升测试失败分类定位算法系统准确性。

目标二：如何有效提取失败原因规则

在实现目标一的基础上，完成多用例同类失败原因特征的正则或字符串规则的提取，提升测试失败分类定位算法系统可解释性。

目标三：有监督算法失败日志分析鲁棒性提升

标签数据是有监督日志分析的前提，由于海量失败日志数据标注工作量巨大且失败日志模式多变，日志标注过程中会引入部分数据标注质量不佳以及类别标签变动等问题，进一步影响了失败分类分析和定位分析准确性。本问题旨在基于海量日志数据与标签信息，需要建立鲁棒性评价模型；在鲁棒性提升的前提下，提高多变日志的普适性，减轻标注质量不佳问题并细粒度地提升智能分析准确率 10%。

# 课题五：业务特性感知的资源调度技术

## 项目背景

随着云计算、微服务、容器等技术的蓬勃发展，如何应对云上业务快速变化，提供稳定可靠的运维保障，计量计费成为业界热点话题，而“合理调度资源，实现最优供给”则是解决这一问题的根本。

基于业务感知的资源调度算法，在产品规划，开发编码，构建，测试，发布，部署和维护等方面，使得发布软件能够更加快捷、频繁和可靠。同时对提高资源利用率，降低成本等多方面起着重要作用。

目前资源平均空闲时间长，利用率整体偏低，主要原因如下：

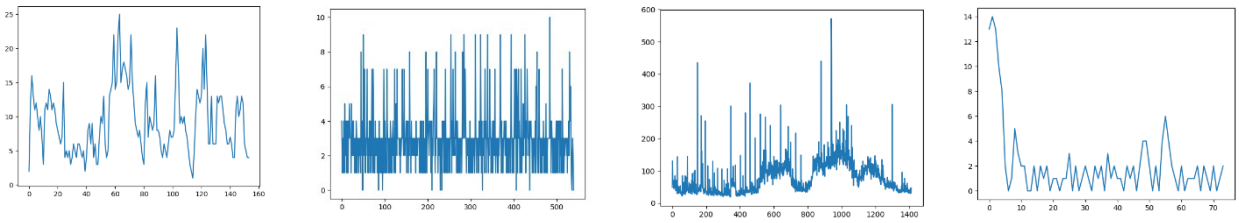
1. 虚拟机数目以峰值任务做静态配备，未按需做动态调整，容易导致大量机器在非峰值时段空闲。
2. 受到机器规格、镜像、系统架构、地域等因素影响机器共享受限。

## 研究内容

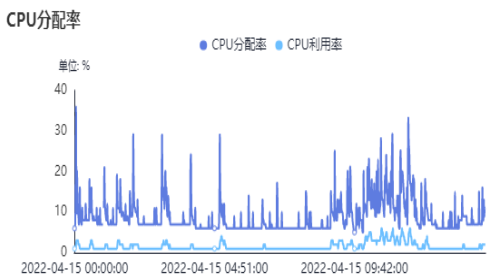
1. 虚拟机弹性调度，充分利用各产品线构建任务的时序错峰特性，在初始静态配额数的基础上实现资源池大小的弹性伸缩，提供机器学习驱动的动态伸缩、定时伸缩、混合伸缩等多种模式，赋予各产品灵活自选的伸缩方案。
2. 容器弹性调度：获取容器集群历史任务数据，分析预测一天内集群宿主机节点的水位，生成策略通知环境服务增加/释放宿主机节点。

可选合作方向：

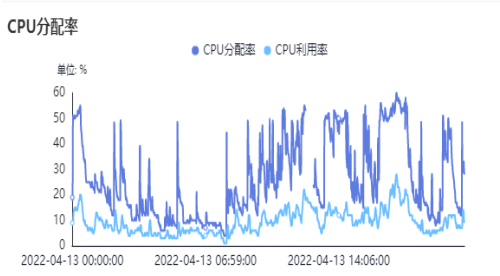
方向一：业务特性感知的资源调度技术实现成本和性能优化建议、业务 SLA 保障算法。目前云上资源任务时间序列波形的多样性（平稳型、突发型、随机型、条形码型等），任务规格多样性，业务周期不断变化，周期性不明确，如何提升预测准确率，对提升资源利用率有显著作用，如图所示：



方向二：CPU 和内存资源有效利用率（10-20%）提升现有 CPU 利用率远低于分配率（80-90%）

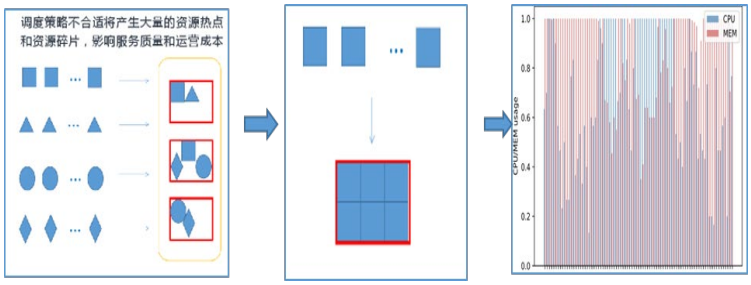



CPU 分配率低




CPU 分配高的情况下 CPU 利用率低

方向三：调度过程中产生大量资源碎片问题，如何合理分配，提升资源利用率




## 项目目标

1. 针对多种多样业务形态实现精准调度平台，提供自动化寻优模型，优化调度策略；
2. 针对应用场景实现算法创新研究，达到提升资源利用率低>30%；
3. 针对资源碎片问题，实现合理调度算法完成动态资源分配，减少资源碎片，实现资源利用率提升 20%。

# 课题六：面向研发数据的知识图谱实体关系识别和融合技术

## 项目背景

公司内软件设计、开发、构建、测试等研发过程沉淀了大量的数据，这些数据来源于不同的服务系统，呈现多源异构的特点。通过提炼研发过程的数据，构建研发知识图谱，可以更好的组织和表示数据。知识图谱具有多维度、结构化特点，常用于查询和推荐等场景，相比机器学习方法，更具有可解释性。在研发过程中可用于用例推荐、根因定界、代码分析等，有助于研发效率的提升。如何将研发数据中结构化、半结构化、非结构化数据抽取成为准确的实体和关系，需要解决研发数据场景中多源异构数据的抽象、实体和关系识别及融合问题，完成高效、准确的知识图谱构建，支撑研发相关应用。

## 研究内容

针对研发数据同时存在结构化、半结构化、非结构化等形式的特点，构建知识图谱，需要将需求、功能、代码、用例、脚本、执行结果、日志、配置、硬件等数据进行实体识别和抽象，还需要解决实体消歧并通过分析其关系模式进行关系补全，完成研发知识图谱的融合。可选合作方向：

方向一：研发知识图谱实体和关系的识别

使用规则挖掘、机器学习、深度学习等方法，对半结构化、非结构化数据，进行实体抽象、识别和关系抽取，如日志中的信息抽取表示、代码元素抽取表示、执行时序关系表示等。构建研发知识图谱，支持研发过程所需的查询和分析。

方向二：研发知识图谱实体消歧

在实体和关系的识别中可能会产生较多相同实体的不同表示，需要对这些实体进行消岐处理，以消除知识图谱的二义性，提升知识图谱的质量，从而达到提升基于知识图谱的研发过程查询和分析的效果。

方向三：研发知识图谱关系补全

基于符号推理和统计推理等技术，根据研发图谱中已有的事实或关系推断出未知的事实或关系，对知识图谱的关系修正和补全，进一步完善研发知识图谱。

## 项目目标

方向一：研发知识图谱实体和关系的识别：实体及关系抽取算法在研发数据上准确率>=90%，主要需要解决研发数据中的结构化、非结构化的实体和关系识别问题，抽取的实体、关系具有可解释性，可以通过人工标注验证。

方向二：研发知识图谱实体消歧：实体消歧算法在研发数据上准确率>=90%，通过研发实体在不同系统表示，整理相关重复实体数据，验证算法有效性。

方向三：研发知识图谱关系补全：目前主要通过知识图谱链接预测 hits@k 衡量，要求该指标在测试数据集上达到 state of the art 水平。能对研发数据集有较好效果，研发数据集通过人工标记验证。

# 课题七：开放应用平台中代码缺陷检测技术

## 项目背景

开放应用平台为一个逻辑多租的开发平台，提供了多租户、多场景的开发能力以及应用全生命周期管理的能力。平台能力强大与否，直接关系到对租户复杂业务的支撑程度。因此，开放应用平台在提升自身性能与安全方面面临着较为急迫的任务，例如在平台自身的 TPS、QPS、吞吐量的提升，可以支撑平台在秒杀、推广、问卷等场景的使用；事务、读写一致性的提升，可以丰富平台在交易、进销存等场景的使用；页面内嵌代码、脚本等技术提升，可以提升业务的柔软、自由度。

## 研究内容（可选合作方向如下）

方向一：页面嵌入脚本的实时安全检测

在开放应用平台，用户自定义的 H5 或者小程序页面中会嵌入用户的程序段、脚本段、图片以及附件。因此对于用户上传的程序段和脚本，需要实时检测是否有安全漏洞、合法合规、数据泄露与窃取等危险操作。对页面嵌入脚本的实时安全检测的要求如下：

1. 识别出用户脚本的高危级别。
2. 识别出用户上传图、片附件等的高危级别与是否合法合规。
3. 满足实时检测，时延性低

方向二：对于代码循环依赖的事前检测与实时探测。

在开放应用平台，存在用户自定义的业务逻辑与外部方法调用。这些程序、脚本在运行期间，会出现单一程序段内的循环调用，也会出现内外部调用链上的循环调用。形成循环调用之后，对平台的计算资源与存储资源会形成极大的浪费，也会因为单一用户的行为造成平台的不可用。因此对于解除循环依赖的要求如下：

1. 事前检测，分析用户的自定义逻辑，识别关键字与关键语义，智能分析提示。
2. 实时探测，识别循环调用的特征，识别循环依赖资源消耗阈值，给出警报分析。

## 项目目标

方向一：页面嵌入脚本的实时安全检测：

1. 扫描脚本并给出高危等级评分，识别率不低于 80%，误报率不高于 10%。
2. 实时扫描速度不低于 500 行/秒，全量扫描速度不低于 100 行/秒。

方向二：对于代码循环依赖的事前检测与实时探测：

1. 事前检测识别率不低于 80%，误报率不高于 10%，检测速度不低于 500 行/秒。
2. 实时探测识别率不低于 95%，检测时延不超过 3MIN。

# 课题八：面向联盟区块链的新型智能合约语言

## 项目背景

区块链正在发展成为新一代数字经济基础设施，支持可信数字经济和数据要素流转，承载海量应用与大规模数据交换，与云计算融合形成云链一体化的服务体系。

随着元宇宙等技术的不断推动，互联网将走向更加开放。每一个人和组织都可以变成独立的数字实体接入，共享和进行价值交换，由此更加凸显出区块链智能合约的重要性。虽然当前的区块链智能合约虽然提供了图灵完备的表达能力、对象化编程语言和高效的执行方法，但是在构建去中心化场景应用依然存在隐私安全、监管和易用性等问题。

## 研究内容

目的：研发适配去中心化环境的语言范式和编译器技术，解决 DAPP 安全隐私、性能、监管等问题，提高智能合约语言自身的执行效率，保障数据安全与逻辑安全，根据需求方便地定制语法和实现，让开发者拥有友好的开发体验。可选合作方向如下：

方向一：增强智能合约数据及逻辑安全；

方向二：增强智能合约执行效率及多场景支持。

## 项目目标

方向一：智能合约数据及逻辑安全

1. 提供相关加密原语和语法，合约内部提供数据隐匿能力，支持对数据确权；
2. 提供形式化验证工具在开发合约阶段进行规约验证，在编译阶段通过定理求解器进行验证，增强逻辑安全；
3. 支持安全资产模型和其他安全类模型。

方向二：智能合约执行效率及多场景支持

1. 支持将合约编译为可移植、体积小、加载快的字节码格式；
2. 支持通过数据流分析技术对合约代码进行数据依赖分析，提高并发效率为现有执行效率 2 倍以上；
3. 根据需求快速推出定制化编程模型，例如可编程分布式协作模型、跨链协同和外部合约调用的异步编程模型；
4. 配套的开发工具，例如单元测试框架、模糊测试工具、依赖包管理器等。