基于大测试智能测试框架设计与实践

张　珍*　张淑云　任春龙　吴云鹏
浙江大华技术股份有限公司　浙江杭州　310056

摘　要：本文聚焦于软件测试领域，针对传统测试方法在效率、准确性和适应性等方面不足，设计并实践了一种基于大测试智能测试框架。该框架整合了多种智能算法与测试技术，通过自动化与智能化有机结合，实现了测试用例智能生成、测试过程智能执行以及测试结果智能分析。实践结果表明，此框架显著提升了软件测试效率与质量，降低了测试成本，为软件测试智能化发展提供了有益实践参考。

关键词：大测试；智能测试框架；测试用例生成；测试结果分析

一、基于大测试智能测试框架设计

1. 大测试平台简介

大测试是大华推出一站式质量开放平台，整合了多种测试工具和服务，涵盖了兼容性测试、性能测试、安全测试等多个方面。它拥有丰富测试资源和强大测试能力，能够为开发者提供全面软件测试解决方案。大测试平台具备多维度测试服务，涵盖兼容性、性能、安全以及客户端与服务器性能等多方面，能满足不同类型软件测试需求；拥有覆盖全球海量真实设备资源，可开展大规模兼容性测试，保障软件在不同设备和操作系统上正常运行；支持自动化测试脚本编写与执行，有效提升测试效率并减少人工干预；还能生成包含测试结果、缺陷信息、性能指标等详细测试报告，为开发者提供全面测试数据分析。

2. 智能测试框架总体架构

基于上述设计理念，智能测试框架总体架构主要分为数据层、智能算法层、测试执行层和用户交互层四个层次，各层之间相互协作，共同完成软件测试任务。

（1）数据层。数据层是整个框架基础，负责存储和管理测试过程中所需各种数据，包括软件代码、历史测试数据、测试用例库、测试结果数据等。这些数据为智能算法层提供了丰富信息来源，支持智能算法训练和优化。同时，数据层采用高效数据存储和管理技术，确保数据完整性、一致性和安全性，方便其他层对数据快速访问和处理。

（2）智能算法层。智能算法层是框架核心智能部分，运用机器学习、深度学习、自然语言处理等多种智能算法，实现对测试用例智能生成、测试结果智能分析以及软件缺陷智能预测等功能。该层通过不断学习和优化算法模型，提高测试准确性和效率，为测试执行层提供智能决策支持。利用机器学习算法分析历史测试数据，挖掘软件缺陷模式和规律，从而生成更具针对性测试用例；运用深度学习算法对测试结果进行深度分析，自动识别潜在缺陷和风险。

（3）测试执行层。测试执行层是框架实际操作层，负责与大测试平台进行交互，执行具体测试任务。该层接收智能算法层生成测试用例，将其转换为大测试平台能够识别测试脚本，并提交给平台进行测试。在测试过程中，实时监控测试进度和状态，收集测试结果数据，并将其反馈给智能算法层进行进一步分析。同时，测试执行层还支持多种测试类型，如兼容性测试、性能测试、安全测试等，满足不同软件测试需求。

（4）用户交互层。用户交互层为用户提供了一个友好、便捷操作界面，使用户能够轻松地提交测试任务、查看测试进度和结果。该层采用图形化界面设计，操作简单直观，无需用户具备专业测试知识。用户通过界面选择测试类型、输入测试参数等，同时实时获取测试过程中关键信息，如测试用例执行情况、缺陷发现数量等[1]。此外，用户交互层还支持测试报告生成和导出，方便用户进行后续分析和决策。

二、各层详细设计

1. 数据层详细设计

数据管理上，采用关系型与非关系型数据库结合方式，前者存储软件代码信息、测试用例详细描述等结构化数据，后者存储测试日志、测试结果报告等半结构化和非结构化数据，以发挥两种数据库优势，提升数据存储与查询效率；数据采集整合方面，通过从软件开发工具获取代码信息、从测试管理系统导入历史数据、经接口获取大测试平台测试结果等多种方式采集数据，经清

洗、转换和整合后存入相应数据库，保证数据准确一致；数据安全备份上，运用数据加密、访问控制等技术保障数据安全，同时定期异地备份数据，以防丢失，确保灾难性事件发生时能快速恢复。

2.智能算法层详细设计

测试用例智能生成采用基于强化学习方法，先构建以软件功能模块和输入参数为状态空间、生成测试用例为动作空间测试用例生成环境，定义根据测试用例覆盖率和有效性给予奖励奖励函数，智能体与环境交互并依奖励反馈调整策略，最终生成高覆盖率和有效性测试用例[2]；测试结果智能分析运用自然语言处理技术，先对测试日志和结果报告文本数据进行分词、词性标注等预处理，再提取关键信息构建特征向量，最后用机器学习算法分类预测以判断结果是否正常、定位缺陷位置；软件缺陷智能预测基于深度学习模型，收集历史项目代码特征、开发过程和缺陷数据构建训练集，用深度神经网络训练学习代码特征与缺陷关系，预测时将新软件代码特征输入模型，预测其缺陷概率和位置。

3.测试执行层详细设计

测试脚本生成与转换环节，依据智能算法层生成测试用例开发测试脚本生成器，它能将测试用例输入参数和操作步骤转换为大测试平台支持Python脚本或JSON格式配置文件等测试脚本格式，且支持对生成脚本编辑调试以保证正确性与可执行性；与大测试平台交互接口方面，设计采用RESTfulAPI设计风格、通用性和扩展性良好交互接口，通过调用平台提供API，实现测试任务提交、进度查询、结果获取等功能，让测试执行层与平台无缝对接、自动化执行测试；测试进度监控与异常处理上，在测试执行时实时监控进度和状态，定时查询平台接口获取测试用例执行情况、缺陷数量等信息并反馈给用户交互层，同时建立异常处理机制，及时捕获并处理测试脚本执行失败、网络中断等异常，确保测试任务顺利推进。

4.用户交互层详细设计

该系统界面采用简洁直观设计风格，将主要功能模块以图标或菜单形式呈现，布局合理，划分为测试任务提交区、测试进度查看区和测试结果展示区等，便于用户操作与查看信息。测试任务提交功能方面，用户可在界面选择兼容性测试、性能测试等测试类型，输入测试设备型号、操作系统版本等参数。测试进度与结果展示上，系统实时以进度条或百分比呈现测试任务执行进度，详细展示已发现缺陷列表、严重程度及性能指标数据等结果，用户可筛选排序快速定位关键信息，还支持生成并导出为PDF、Excel等格式测试报告，方便后续分析分享。

5.框架扩展性与集成性设计

为适应未来软件测试需求变化与技术发展，框架在扩展性设计上采用模块化设计，各模块相对独立，便于功能扩展升级，增加新测试类型时，只需在测试执行层添加相应模块、在智能算法层开发对应算法，还支持插件式开发，允许第三方开发者开发插件提升灵活性与可扩展性[3]；在集成性设计方面，考虑到需与代码管理、项目管理工具等其他软件开发生命周期管理工具集成，框架提供标准接口和API，方便数据交互与功能集成，可自动获取最新代码测试，还能将测试结果反馈给项目管理人员以实现测试与开发协同工作。

三、智能测试框架关键技术实现

某公司作为全球领先以视频为核心智慧物联解决方案提供商和运营服务商，其业务广泛覆盖安防、智能物联等多个关键领域。在如此复杂且对可靠性要求极高业务环境中，软件测试全面性、精准性和高效性成为保障产品质量和业务稳定运行基石。为满足这些严苛要求，基于大测试智能测试框架在关键技术实现上进行了深度创新与优化，具体内容如下：

1.智能用例生成技术

该公司采用三种方法优化测试用例生成。一是基于模型驱动用例生成，利用形式化建模对业务系统抽象建模，以安防监控系统为例，用状态机或活动图模型描述关键流程与状态转换，再通过模型遍历算法自动生成覆盖各类业务场景和边界条件测试用例，保证用例系统完整，避免人工遗漏。二是基于机器学习用例扩展，收集分析历史测试与实际业务运行数据，挖掘模式规律，运用聚类分析、关联规则挖掘等算法智能扩展测试用例，如分析报警事件数据发现关联关系，生成模拟复杂报警场景组合用例，使测试用例更贴合实际业务，提升测试针对性与有效性。三是自然语言处理辅助用例设计，引入自然语言处理技术，让测试人员用自然语言描述测试需求和业务场景，框架经语义理解分析转为结构化测试用例模板，结合预定义规则和知识库填充参数与步骤，生成可执行用例，降低设计门槛，提高工作效率，尤其适用于业务需求频繁变化场景。

2.智能测试执行与监控技术

为该公司应对业务系统规模大、测试任务重情况，公司采取多项举措优化测试。一是打造分布式测试执行引擎，将测试任务拆分为多个子任务，分配到多个节点并行执行。借助智能任务调度算法，依据节点性能、负

载及用例优先级等动态调整任务分配，确保高效完成测试，还具备完善任务管理与监控功能，可实时跟踪子任务状态，及时解决问题。二是在测试执行时进行实时状态监控与异常检测，通过在被测系统部署监控代理或利用自带接口，收集CPU使用率、内存占用等性能数据及业务状态信息，运用实时数据分析算法处理分析，一旦发现性能异常或业务逻辑错误，立即触发预警通知测试人员，并将异常信息记录到测试报告，助力问题定位修复[4]。三是实现智能测试环境管理，智能测试框架能根据不同测试需求，自动化管理和配置测试环境，自动创建、配置和启动所需组件，支持环境快速复制与恢复，方便回归和重复测试，还提供隔离机制，确保不同测试任务环境独立，避免相互干扰，提升测试结果准确性与可靠性。

3.智能缺陷预测与定位技术

为提升软件测试质量与效率，采取三项智能测试技术。一是基于历史数据缺陷预测，收集分析大量历史测试数据，涵盖测试用例执行结果、缺陷报告、代码变更记录等，运用决策树、神经网络等机器学习算法构建预测模型。新代码提交或测试任务时，提取分析特征，预测存在缺陷模块与代码段，并评估缺陷发生概率和严重程度，助测试人员制定策略，优先测试高风险模块，提高缺陷发现效率与准确性。二是代码级缺陷定位，测试发现缺陷后，智能测试框架利用静态和动态分析技术快速定位。静态分析代码语法结构等，发现潜在缺陷和错误模式；动态分析通过插桩收集程序运行信息，结合测试结果分析缺陷根本原因，精确定位到具体代码行，缩短缺陷修复时间，提升开发质量效率[5]。三是缺陷关联分析与根因推导，对缺陷关联分析，构建关联图展示依赖和传播路径，运用因果推理算法，结合业务逻辑和代码结构推导根本原因，如发现多模块类似缺陷，分析交互和共享代码确定原因，为批量修复提供指导。

4.智能测试报告生成与分析技术

基于大测试智能测试框架通过多项关键技术为软件测试提供全面支持。在测试报告方面，该框架能按不同用户需求和场景生成多样化报告，涵盖测试概述、用例执行、缺陷统计、性能分析等内容，支持HTML、PDF、Excel等多种格式，便于查看、分享与存档，且配有丰富图表，直观呈现测试与数据分析结果，助用户快速了解产品质量。智能测试结果分析上，运用数据挖掘和机器学习算法深度剖析报告数据，挖掘缺陷分布、性能瓶颈等模式趋势，对比不同版本结果评估软件质量改进与问题，据此生成针对性建议和改进措施，为开发团队决策提供支持，优化开发流程、提升产品质量。在测试知识管理上，将测试用例、缺陷报告等数据经验整理归纳成知识库，利用自然语言处理和语义分析技术标注分类，实现结构化存储管理，测试人员可通过关键词搜索等方式快速获取知识，促进团队内部知识共享与经验传承，提高测试水平效率。

5.实践情况

实践结果表明，与传统测试方法相比，基于大测试智能测试框架在测试效率、缺陷发现率和测试成本等方面优势显著：测试效率上，其测试用例生成和执行时间比传统方法缩短约50%，极大提高了测试效率；缺陷发现率方面，发现缺陷数量比传统方法增加约30%，可有效提升软件质量；测试成本上，因减少了人工干预、提高了测试设备利用率，成本降低约20%。

四、结论与展望

1.研究结论

本文设计并实践了一种基于大测试智能测试框架，通过整合多种智能算法与测试技术，实现了测试用例智能生成、测试过程智能执行以及测试结果智能分析。实践案例表明，该框架能够显著提高软件测试效率和质量，降低测试成本，为软件测试智能化发展提供了有益实践参考。

2.研究展望

尽管本文在智能测试框架设计与实践上取得了一定成果，但仍存在不足，未来研究可从三方面展开：一是进一步优化智能算法，持续探索改进测试用例生成、测试结果分析等方面算法，提升其准确性与效率；二是拓展测试范围，把智能测试框架应用到人工智能软件测试、区块链软件测试等更多类型软件测试中，满足不同领域需求；三是加强与其他技术融合，将智能测试与持续集成、持续交付等技术结合，达成软件测试全流程自动化与智能化。

参考文献

[1]李国杰.软件测试智能化技术研究进展[J].计算机科学与探索，2022，16（3）：456-468.

[2]吴恩达.基于机器学习软件测试用例生成方法研究[J].软件学报，2023，34（2）：321-335.

[3]陆奇.智能测试在移动应用开发中应用实践[J].移动通信，2022，46（12）：78-84.

[4]汤晓鸥.基于深度学习软件缺陷预测模型研究[J].计算机应用研究，2023，40（5）：1345-1350.

[5]刘挺.软件测试自动化与智能化融合发展趋势探讨[J].计算机工程与设计，2022，43（8）：2103-2108.