智能软件工程中的人工智能应用：软件测试的现状、挑战与未来趋势

2022112831　蔡志宇

摘　要：本文系统化总结了人工智能技术在软件工程，特别是软件测试中的应用现状。通过分析大规模语言模型（LLMs）、模糊测试、组件化测试等智能技术的最新研究，探讨其在自动化测试、程序修复、漏洞检测中的优势和挑战。本文还评估了应用商店对软件测试实践的影响，并介绍了智能算法的多维度评估框架，涵盖准确性、效率和鲁棒性等非功能需求。本文为未来的研究方向提供了见解，建议进一步探索自适应测试技术和智能化教育工具的应用。

关键词：人工智能；软件测试；自动化测试；大规模语言模型；智能算法评估

1 引言

1.1 研究背景

在当今软件工程的快速发展中，人工智能（AI）技术的应用逐渐成为一个核心趋势，极大地提升了软件开发的各个环节，尤其是在软件测试中的应用尤为显著。随着软件系统规模和复杂性的增加，传统的软件测试方法逐渐显露出局限性，如人工测试的高成本和自动化测试覆盖不足等问题[1]。因此，如何有效利用智能技术进行软件测试，确保软件产品的质量和可靠性，成为软件工程领域的重要研究课题。

特别是近年来，大规模语言模型（LLMs）作为自然语言处理和人工智能领域的突破性技术，展示了在处理大规模数据集和执行复杂任务上的非凡能力。这些模型在软件测试中的应用，例如测试用例生成、程序修复等，展现出显著的潜力。与此同时，模糊测试、基于组件的测试等智能化技术的逐渐成熟，也在推动着软件测试的发展。

1.2 综述目的

本文的目的是系统化总结人工智能技术在软件工程，尤其是软件测试中的应用现状。通过分析近年来的相关研究成果，本文将探讨当前 AI 驱动的软件测试方法的优势与挑战，并指出未来可能的研究方向。这些分析不仅有助于了解现有技术的局限性，还能为未来研究提供借鉴与参考。

研究范围

本文主要聚焦以下几个方面：

1. 人工智能技术在软件工程中的应用，包括模型驱动和组件化的软件测试方法。
2. 大规模语言模型（LLMs）在软件测试中的使用，涵盖测试用例生成、代码修复等任务。
3. 模糊测试在自动驾驶和网络连接系统中的应用，尤其是面向漏洞的测试框架。
4. 应用商店对软件工程实践的影响，特别是在发布管理和测试环节的作用。
5. 针对智能算法的多维度评估方法，以及 AI 驱动软件的非功能性需求测试。

2 智能软件工程的当前研究

2.1 人工智能技术在软件工程中的应用

系统化映射研究

近年来，人工智能技术已深入应用于软件工程的各个阶段，包括需求分析、设计、开发、

测试和维护等。通过系统化映射研究，可以发现 AI 技术在软件工程生命周期内的不同贡献。例如，机器学习、深度学习和数据挖掘等技术在需求挖掘、代码生成、错误检测和性能优化等方面得到了广泛的应用[2]。AI 的引入显著提高了软件开发过程的自动化程度，并极大地增强了项目的成功率和软件产品的质量。

特别是 AI 技术在软件测试中的应用，能够有效帮助软件工程师识别出开发过程中的潜在漏洞，并通过智能算法优先处理这些问题。这种技术不仅减少了发现漏洞的时间，还能优化测试和检查成本。通过 AI 驱动的自动化测试，尤其是在测试覆盖率和精确性方面，AI 展示了其显著优势。

关键 AI 技术

在软件工程中，常见的 AI 技术包括机器学习、深度学习、数据挖掘、自然语言处理和大预言模型等。这些技术能够支持自动化任务，例如需求预测、代码审查和测试案例生成等。例如，基于大预言模型的代码生成和错误检测算法在实际应用中展现了极大的潜力。这类技术不仅提高了软件开发效率，还能在复杂的项目中保证较高的质量控制。

此外，AI 技术也帮助软件工程师更好地处理大规模数据集，从中提取有用的信息来辅助项目决策。例如，基于数据挖掘的需求预测工具，能够根据历史数据为软件团队提供未来可能发生问题的警示。

2.2 组件化和模型驱动的软件测试

组件化测试方法

组件化和模型驱动的软件测试技术在现代软件开发中占据重要地位，尤其是在复杂系统（如机器人软件生态系统）中。例如，SmartTS 方法是一种基于组件和模型驱动的测试框架，用于验证系统中各个组件之间的相互作用。组件化测试的一个关键优势是，它允许开发者为每个独立的组件生成特定的测试套件，并且在不影响组件封装的情况下进行验证和模拟[3]。

通过使用组件化方法，系统构建者可以轻松验证功能性和非功能性需求是否得到满足，从而提高系统的可靠性和安全性。尤其在开放环境下的机器人软件系统中，SmartTS 通过与组件模型紧密绑定的测试套件，提供了可实证验证的测试记录，有助于组件选择和系统集成。

模型驱动测试的优势和挑战

模型驱动测试（MDT）方法是一种基于系统模型来生成测试用例的技术[3]。在 MDT 中，测试套件直接从系统模型中推导出来，这不仅减少了测试用例的开发时间，还能够确保测试的全面性和覆盖率。然而，模型驱动测试也存在一定的挑战，例如模型的质量对测试的效果有着直接影响。如果系统模型不完整或不准确，生成的测试套件可能无法有效覆盖所有潜在问题。

通过组件化和模型驱动的测试方法，开发者能够在不影响组件独立性的情况下进行全面的

测试验证，这对于复杂系统的开发和维护具有重要意义。

3. 软件测试中的人工智能技术

3.1 大规模语言模型在软件测试中的应用

LLMs 的应用场景

大规模语言模型（LLMs）近年来成为自然语言处理领域的突破性技术，这些模型在生成测试用例和程序修复等软件测试任务中展现了显著的应用潜力。LLMs 通过从大规模数据集进行训练，能够自动生成符合特定条件的测试用例[4]，这在过去依赖手工编写测试用例的情况下是极其耗时的。此外，LLMs 还可以用于代码的自动修复，如通过分析错误日志或调试信息，生成修复补丁。

常见的应用包括：

1. 测试用例生成：通过利用预训练的语言模型，可以自动生成覆盖广泛测试场景的测试用例，从而提高测试覆盖率和效率。
2. 程序修复：LLMs 可以根据代码片段的语义分析来自动修复程序错误，这极大地简化了软件开发中的错误修复过程。

挑战与机遇

尽管 LLMs 在软件测试中的应用前景广阔，但仍存在一些关键挑战。例如，生成的测试用例在某些情况下可能无法有效覆盖复杂系统的所有功能，尤其是涉及到系统特定语义的部分。此外，虽然 LLMs 能够生成大量的测试用例，但如何确保这些用例的质量和准确性仍是一个研究难题。

不过，LLMs 为软件测试带来的机遇也不可忽视。例如，通过先进的提示工程（prompt engineering），可以进一步提高模型的输出精度，使其在复杂任务中表现更为优越。未来研究可着重于如何改进 LLMs 的提示设计，以更好地适应软件测试中的特定场景。

3.2 基于漏洞的模糊测试

模糊测试技术

模糊测试（Fuzz Testing）是一种广泛应用于软件安全和漏洞发现中的测试技术，尤其在自动驾驶和连接车辆系统等领域具有重要应用[5]。该技术通过向系统输入随机数据，试图触发系统中的潜在漏洞，以发现未曾预见的错误。传统的模糊测试依赖于随机化输入，因此在测试覆盖率和效率上存在局限。

为了提高测试的效率和针对性，近年来提出了面向漏洞的模糊测试框架（如 VulFuzz），该

框架通过安全漏洞度量来指导和优先处理最易受攻击的组件。这种方法利用基于灰盒的模糊测试技术，在确保代码覆盖率的同时，能够更加有效地检测出系统中的漏洞。

输入结构感知的变异技术

VulFuzz 在传统模糊测试的基础上，增加了输入结构感知的变异技术，以此绕过系统中可能存在的输入格式限制。这种方法通过模拟输入结构，使得测试用例能够更深入地探测系统中的潜在漏洞，避免了传统模糊测试中因输入格式错误导致的测试失败。通过这种方式，VulFuzz 在较短时间内显著提高了系统漏洞的发现效率，特别是在自动驾驶系统中取得了优异的测试效果。

4. 应用商店对软件工程实践的影响

开发者与用户之间的桥梁

应用商店不仅改变了软件分发的模式，还对软件开发者与用户之间的互动产生了深远影响。在传统的软件开发模式中，开发者和用户之间的联系较为间接，而应用商店的出现使得这种联系更加紧密[6]。通过应用商店，用户可以直接提供反馈、打分和评论，这些信息对开发者的开发决策产生了重要影响，尤其是在测试和功能改进方面。

开发者能够利用来自用户反馈的信息，快速识别出应用程序中的问题，并进行有针对性的测试和修复。应用商店的评价机制还推动了测试策略的调整，开发者更加重视用户在应用商店中提出的反馈，这使得测试不再仅仅依赖传统的开发周期，而是需要根据用户反馈不断进行调整和改进。

市场透明度的提升

应用商店增加了软件市场的透明度，使得用户和开发者能够更清晰地了解应用程序的表现和市场需求。特别是在移动应用开发中，应用商店的发布策略对测试管理的影响尤为显著。例如，开发者会根据应用商店的发布周期和用户行为分析结果，调整测试的重点，以确保在发布前发现并修复潜在的问题。

通过这种方式，应用商店的存在不仅影响了开发流程中的测试实践，还对整个软件生命周期的管理提出了新的要求。未来研究可以进一步探讨如何将用户反馈与自动化测试技术相结合，提升软件质量和用户满意度。

5. 智能算法评估技术

多角度算法评估框架

随着人工智能（AI）软件在多个领域（如军事、工业和智能系统）的广泛应用，对智能算法的评估需求越来越高。传统的软件测试方法通常集中于功能性需求，然而，AI 软件不仅仅

需要满足功能性要求，还涉及到准确性、效率、鲁棒性和覆盖率等非功能性需求。为了应对 AI 软件的特殊性，提出了基于多角度的算法评估框架[9]，主要从以下四个方面对算法进行评估：

1. 准确性评估：衡量 AI 算法在给定数据集上的预测或分类性能。高精度的算法能够在不同条件下保持一致的输出结果。

2. 效率评估：该指标关注算法在计算资源消耗和响应时间方面的表现，尤其是在大规模数据处理场景下，AI 算法的效率尤为关键[7]。

3. 数据鲁棒性评估：考察算法在面对不完美或有噪声的数据时的表现，确保 AI 软件在非理想输入条件下仍能做出合理的决策[8]。

4. 算法覆盖率评估：该指标评估算法在不同场景下的适用性，确保算法能够处理多种任务和输入场景。

该评估框架通过综合多维度的评估指标，能够更加全面、准确地判断 AI 算法的质量，从而为 AI 软件的未来发展提供支持。此外，AI 算法的评估不仅限于传统的软件质量控制，还需要考虑其自适应学习和进化特性。

非功能需求的覆盖

传统的软件测试方法难以全面覆盖 AI 软件的非功能性需求，特别是在测试数据获取和注释方面存在较大挑战。AI 软件的自学习和自适应特性使其表现出更大的随机性和不确定性，因此，测试用例的生成和评估需要考虑这些复杂性。

为了解决这些问题，智能算法评估技术可以利用大数据和机器学习技术，自动化地生成评估数据集，并通过对评估指标的动态调整，优化测试过程。这一方法不仅可以显著减少手动生成测试数据的时间，还能更好地应对复杂场景中的非功能性需求。未来的研究方向包括如何进一步自动化 AI 软件测试数据的生成，以及如何改进评估指标的多样性和精准性。

6. 教育领域的软件测试研究

软件测试教育中的挑战

随着软件系统的规模和复杂性不断增加，培养具备高技能的软件测试工程师成为了一个迫切的需求[10]。近年来，许多大学已将软件测试课程纳入计算机科学或软件工程课程体系，但测试教育仍然面临诸多挑战。这些挑战主要集中在以下几个方面：

1. 教学资源的不足：尽管许多高校已经意识到软件测试的重要性，但实际的教学资源（如教材、案例库和工具）仍然匮乏。许多课程中使用的工具和技术较为陈旧，难以跟上快速发展的测试技术。

2. 理论与实践的脱节：在很多课程中，软件测试的教学往往偏重于理论，而忽略了实际操

作能力的培养。学生在课堂上学习了测试技术，但由于缺乏实际的项目经验，往往难以将理论应用于真实的开发环境。

3. 跨学科的需求：现代软件系统的测试要求学生不仅要具备软件工程知识，还需要了解其他领域的知识（如人工智能、数据科学等）。然而，现有的测试课程往往只关注软件开发本身，缺乏跨学科的整合。

教学工具与方法

为了应对这些挑战，许多高校和研究机构正在探索如何改进软件测试教育的教学工具与方法。例如，一些研究建议将测试技术与其他软件工程课程相结合，通过项目驱动的方式进行教学，让学生在真实的开发场景中应用测试技术。此外，教学工具方面的创新也在不断发展，许多学校开始引入自动化测试平台、云计算环境和开源测试工具，以帮助学生更好地掌握测试技术。

在教学方法上，许多教育者正在尝试混合教学模式，即通过线上和线下结合的方式进行教学，利用在线测试平台让学生进行自我测试和模拟实践。这种方式不仅能够提高学生的学习积极性，还能提供个性化的学习路径，使得不同水平的学生都能得到有效的学习支持。

总之，软件测试教育的未来发展需要更多地关注如何将理论与实践相结合，并通过跨学科的方式培养适应现代软件系统需求的测试人才。

7. 结论与未来研究方向

研究总结

通过本文的综述，我们系统回顾了智能软件工程和软件测试领域的最新进展，重点讨论了人工智能技术在软件测试中的应用前景、挑战以及教育实践中的发展需求。在软件测试中，尤其是大规模语言模型（LLMs）的应用已显示出巨大的潜力，不仅在自动化测试生成、程序修复等方面表现优异，还为未来的软件工程工作流程带来了新的视角。此外，面向漏洞的模糊测试技术在连接自动驾驶车辆系统等复杂场景中的应用，进一步拓展了智能软件测试的边界。

针对智能算法的多维度评估框架，为 AI 软件的测试提供了全面的非功能性评估工具，提升了 AI 系统的鲁棒性、效率和覆盖率。与此同时，应用商店对软件工程实践的影响表明，用户反馈对测试流程的调整至关重要，开发者应充分利用来自应用商店的数据进行迭代优化。

在教育领域，虽然软件测试的课程设计和实践应用仍面临诸多挑战，但新的教学工具和方法的引入，为培养具备综合技能的软件测试工程师提供了重要的支持。

未来研究建议

尽管目前的研究已经取得了显著进展，但在未来的研究中，仍有几个关键领域值得进一步探索：

1. LLMs 与自动化测试的进一步集成：尽管 LLMs 已经在测试用例生成和程序修复中展现了良好效果，但未来的研究可以进一步探讨如何更有效地将 LLMs 与自动化测试工具相结合，提高复杂系统的测试覆盖率和测试用例的质量。此外，还应研究如何改进提示工程，使得 LLMs 能够在更广泛的测试场景中表现出更好的适应性。

2. 面向智能系统的自适应测试技术：随着自动驾驶、智能设备等新兴技术的发展，针对这些系统的测试技术亟需进一步改进。特别是如何应对复杂系统中的实时反馈和动态行为，是当前面临的重大挑战。未来的研究应重点探索自适应测试技术的开发，以应对这些智能系统的特殊需求。

3. 跨学科的智能算法测试框架：智能算法的测试不仅需要考虑软件系统的传统功能，还应覆盖数据处理、机器学习和深度学习等多学科领域。未来的研究可以探索如何开发统一的测试框架，整合不同学科的测试需求，尤其是在大规模数据集和高动态场景下的应用。

4. 教育和工具的进一步优化：随着软件测试领域的不断发展，教育中的工具和方法也需要与时俱进。未来的研究可以探索如何进一步利用自动化测试平台、开源工具和云计算环境，来改进软件测试教育的质量。此外，如何培养学生在实际项目中的测试应用能力，以及如何提升跨学科测试技能，也是未来教育领域的重要研究方向。

未来的研究应继续聚焦如何提升 AI 技术在软件测试中的应用深度与广度，解决当前存在的测试覆盖不足和自动化程度不高等问题，同时积极探索跨学科、跨领域的解决方案。通过智能化技术与传统软件工程方法的结合，未来的软件测试将更加高效、智能化。

参 考 文 献

[1] Boukhlif, Mohamed, Mohamed, Hanine, Nassim, Kharmoum. "A Decade of Intelligent Software Testing Research: A Bibliometric Analysis". Electronics 12. 9(2023).

[2] Alshammari F. H. (2022). Trends in Intelligent and AI-Based Software Engineering Processes: A Deep Learning-Based Software Process Model Recommendation Method. Computational intelligence and neuroscience, 2022, 1960684. https://doi.org/10.1155/2022/1960684

[3] Vineet Nagrath, , Christian Schlegel. "SmartTS: A Component-Based and Model-Driven Approach to Software Testing in Robotic Software Ecosystem". International Journal of Advanced Computer Science and Applications 12. 7(2021).

[4] Junjie Wang, Yuchao Huang, Chunyang Chen, Zhe Liu, Song Wang, & Qing Wang. (2024). Software Testing with Large Language Models: Survey, Landscape, and Vision.

[5] L. J. Moukahal, M. Zulkernine and M. Soukup, "Vulnerability-Oriented Fuzz Testing for Connected Autonomous Vehicle Systems," in IEEE Transactions on Reliability, vol. 70, no. 4, pp. 1422-1437, Dec. 2021, doi: 10.1109/TR.2021.3112538.

[6] A. A. Al-Subaihin, F. Sarro, S. Black, L. Capra and M. Harman, "App Store Effects on Software Engineering Practices," in IEEE Transactions on Software Engineering, vol. 47, no. 2, pp. 300-319, 1 Feb. 2021, doi: 10.1109/TSE.2019.2891715.

[7] Genheden, S., Thakkar, A., Chadimová, V., Reymond, J. L., Engkvist, O., & Bjerrum, E. (2020). AiZynthFinder: a fast, robust and

flexible open-source software for retrosynthetic planning. Journal of cheminformatics, 12(1), 70. https://doi.org/10.1186/s13321-020-00472-1

[8] C. Pan, J. You and Y. Gao, "Survey on Reliability Engineering for AI Software Systems: An Extension Based on the IEEE 1633 Standard," 2023 3rd International Symposium on Artificial Intelligence and Intelligent Manufacturing (AIIM), Chengdu, China, 2023, pp. 116-121, doi: 10.1109/AIIM60438.2023.10441228.

[9] Y. Zhang, X. Shen, J. Ding, L. Wu and L. Tang, "Research on Intelligent Algorithm Evaluation Technology of AI Software," 2024 IEEE 3rd International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA), Changchun, China, 2024, pp. 354-358, doi: 10.1109/EEBDA60612.2024.10485944.

[10] Vahid Garousi, , Austen Rainer, Per Lauvås jr au2, Andrea Arcuri. "Software-testing education: A systematic literature mapping." (2020).