大模型下的软件工程：机遇与挑战
——第九期CCF秀湖会议报告

整理：金 芝 夏 鑫 高翠芸

编者按：2024年1月19~21日，第九期CCF秀湖会议在苏州CCF业务总部&学术交流中心举行。在为期三天的会议中，来自学术界与工业界的20余位专家围绕“大模型下的软件工程：机遇与挑战”这一主题进行了深入交流和研讨，形成如下报告。

背景与意义

软件是人类制造的最复杂的一类制品，软件工程（SE）是指将工程化原则和方法应用于软件开发、运行、维护和管理的过程中，实现软件的高质量、高效率、可靠性和可维护性。当前，ChatGPT等大语言模型（LLM）在需求分析、代码生成、测试生成、程序修复、代码优化等众多软件工程任务中取得了突破性进展，为软件工程的研究和实践带来了全新的机遇和挑战。在大模型时代，亟须探索从数字化、知识化到高度智能化的人机协同软件开发，构建任务驱动、数据驱动、模型驱动、可信融合的软件工程技术。

基于以上问题，本次会议从“大模型支持软件工程（LLM4SE）”和“软件工程支持大模型（SE4LLM）”两个角度进行集中研讨，邀请了20余位来自软件工程、系统软件、人工智能等不同专业领域的学术界与企业界的专家学者，围绕“大模型下的软件工程：机遇与挑战”这一主题开展观点分享和思想碰撞。会议按照六个专题进行组织：“大模型下的软件工程”专题邀请领域专家总结大模型下的软件工程的现状及主要问题，为后续交流与研讨提供背景知识，并突出需求设计工程的重要性；“大模型与代码智能”专题聚焦当前最主要的一个智能化软件工程应用子领域——代码辅助与生成；“大模型与软件质量保障”专题关注软件测试及代码分析、检视和运维领域的发展趋势；“代码大模型的数据、评估与验证”专题关注代码大模型的数据工程以及评估和验证环节；“大模型与开源工程”专题关注供应链生态和大模型的开源生态构建；“LLM4SE的未来与挑战”专题关注大模型与软件工程交叉融合的前景和机遇。

大模型下的软件工程

以ChatGPT为代表的大模型在文本理解和生成等方面展示出了惊艳的效果，在很多领域引发了广泛关注和讨论。通过构建软件工程领域的专用大模型解决相关软件工程问题成为研究热点。

实践分享

20世纪60年代，软件工程被正式提出，各领域学者开始注重程序结构的研究，使程序设计语言和编译系统得到广泛应用。在软件工程发展的早期阶段，人工智能和机器学习技术尚未得到广泛应用，

软件工程主要关注结构化编程、模块化设计和数据结构。伴随着结构化编程、面向对象编程和云计算的快速发展，人工智能和机器学习技术不断成熟，智能化软件工程（AI4SE）逐渐兴起。一些智能化技术被引入软件开发、测试和维护等过程中，如智能化需求分析与设计、智能化代码生成与修复、智能化项目管理等，以提高软件开发的效率和质量。

2021年末，OpenAI发布了首个代码大模型Codex，用于代码生成。Codex与GPT-3的架构类似，使用从GitHub上收集的159 GB公开代码数据进行预训练。基于Codex的Copilot插件目前已成为代码生成辅助工具的标杆，Codex论文中提出的HumanEval数据集也成为后续代码生成常用的评估数据集之一。截至2024年1月，针对软件工程领域的大模型已经有20余个，发布机构包括企业界的OpenAI、DeepMind、Salesforce、华为、微软等和学术界的伊利诺伊大学香槟分校、清华大学等，参数量也从早期的几十亿发展到几千亿。

2023年人工智能生成内容（AIGC）技术的飞速跃进，为软件开发模式的革新铺垫了道路。以大模型为核心的软件工程新体系，因其潜力巨大，成为了学术界与工业界瞩目的焦点。GitLab Duo、GitHub Copilot X、Tabnine、Codota等工具的成功部署，进一步拓宽了AI在软件工程领域的应用疆域。高德纳发布的《2024年十大顶级战略技术趋势》包含了平台工程、AI增强开发、行业云平台、智能应用等，凸显出未来软件工程与AI结合的前景和优势，以及对新的软件开发模式的迫切需求。

在大模型的引领下，一场聚焦于高效利用AI的软件工程革新正在兴起：构建一个协同进化、开放共享的软件知识平台，使大模型洞悉复杂系统实现的全貌及其业务与技术背景，以自然的方式促进软件开发知识的高效流通与应用；借助大模型卓越的知识处理能力，促进研发流程数字化，减少知识冗余和重复劳动，提升文档与知识工作的价值；通过强化数据治理与数字化项目实施，奠定坚实的数据基础，结合软件开发累积的数字化知识，利用大模型的记忆力、理解力和关联分析优势，增强智能化开发的效能与可信度。各方学者与实践者的积极探索，不仅为软件工程开辟了新的增长点，也带来了前所未有的挑战与机遇。

观点争鸣

南京大学吕建院士强调大模型的成功带来的鲶鱼效应，几乎为所有领域打开了新的探索空间。他指出人工智能是模仿人类行为并与之交叉产生的智能行为，不存在人工智能完全替代人类的情况，大模型时代仍然需要使用知识、推理和因果关系，应用大模型要扬长避短，将其应用到合适的领域，实现大模型的持续健康发展。在条件约束、场景适配、系统化适配等前提下，应用领域要积极拥抱大模型带来的变化，不断自我锻炼，实现逐步提升。吕建强调，我们当前处于一个“大观”世界，大模型链接世界所有知识，要积极寻找新的科学基础，促进“大观”世界与传统科学有机结合。

清华大学胡事民院士分析了大模型在软件工程中的应用，特别是自动代码生成技术的发展历程和当前面临的挑战。他指出，尽管自动代码生成技术对于提高软件开发效率至关重要，但代码生成大模型在数据需求、骨干网络设计、训练与推理效率以及致幻现象等方面仍面临诸多难题。他介绍了计图（Jittor）深度学习框架在支持大模型训练和推理方面的创新，包括元算子概念、统一计算图思想等。基于计图开发的Fitten Code编程助手在代码生成的速度和准确度上超越了OpenAI Copilot和CodeGeeX等。胡事民对大模型在软件工程领域的未来发展方向进行了展望，包括提升训练数据集的规模与质量、制定AI代码生成规范、完善自动化生成代码的版本管理机制，以及加强软件缺陷检测等，以期提高开发效率和处理复杂系统方面的能力。

南京大学教授李宣东指出，大语言模型在软件开发领域打开新空间的同时，也带来了可信性判断的挑战。软件开发是一个将自然语言需求转化为程序代码的复杂决策性任务，而大语言模型在软件开发的多个阶段，如需求分析、设计、编码等，展现出强大的预测性内容生成能力，这为软件工程带来

新的机遇。然而，这些模型生成的代码存在可信度缺乏的问题，需进一步由人工审核和优化。由此便催生了人机协同编程的模式，要求开发者引导大模型生成预测性的代码，随后进行细致的分析、理解和必要的修改，以确保代码的质量。同时，通过执行静态分析、动态测试和形式化验证等可信保障活动，进一步提升软件的可靠性。李宣东指出，人机协同编程模式的兴起对开发者提出了更高的专业能力要求，包括对大模型生成内容的可信性判断能力。为了有效利用大模型工具并推动软件工程的未来发展，需要在技术发展、教育创新和专业能力培养这三个关键领域实现全方位的提升。

华为2012软件工程应用技术实验室主任夏鑫指出，大模型驱动软件工程的发展正处于3.0时代，它不仅革新了传统的开发流程，还带来了新的挑战与机遇。借助ChatGPT、Codex等大模型，能显著提升代码自动生成与测试用例生成的准确率，优化软件测试与开发效率。然而，模型在复杂逻辑推理、小语种支持、遗留系统现代化、测试脚本与修复等方面尚存在局限性。硬件与软件栈的精度问题，如模型训练损失对齐与端到端收敛，及多机集群下的调试难题，是待解的关键技术障碍。此外，高质量数据集构建、模型评估体系、测试需求与指令调优成为当前研究热点。

大模型下的需求分析和编程

需求和设计是软件开发的关键阶段，目前仍然是软件开发过程中极具挑战的过程。软件编程将抽象的设计理念、需求规格说明书以及业务逻辑转换为机器可理解和执行的形式，在软件开发过程中扮演着至关重要的角色。随着大模型在软件工程领域的应用，未来将有效辅助生成需求工程阶段的高质量需求规格说明书和代码原型，进一步提升后续编程环节的效率和质量。

实践分享

软件需求分析是软件开发过程的基石，是指收集、分析、规范及管理系统、产品或服务需求的过程。该过程有助于帮助项目团队和利益相关者明确软件的目标、功能和性能要求，确保开发方向与用户期望一致。通过早期识别和明确需求，可以减少后期因需求变更带来的返工，降低开发成本和项目延期的风险。良好的需求分析有助于构建模块化、可扩展和易于维护的软件，降低故障率。

智能编程技术被应用于多种场景，包括代码补全、代码搜索等方面。这些应用旨在提高软件开发的效率，减少人为错误，以及辅助开发者快速实现功能。近年来，大语言模型在代码生成任务中得到了广泛的应用。这些模型采用了各种不同的架构，包括CodeBert、PLBART和CodeGPT等代码大模型。这些模型在代码语料库上进行预训练，以深入理解代码的语法、语义和惯用结构。为了增强模型对代码复杂性的理解，一些创新的方法整合了结构化表示。GraphCodeBert在CodeBert的基础上整合了基于图的表征，而CodeT5将编码器－解码器范式与代码的结构本质相结合。

开源社区为代码生成研究提供了丰富的资源，包括开源代码库、数据集和工具。例如，GitHub和Stack Overflow等平台提供了大量的代码样本，这些样本被用于训练和评估大模型。基于大规模数据，Codex和CodeGen构建了具有数十亿参数的大规模模型，这些模型在代码生成任务中展示出先进的性能，能够辅助开发者编写代码，提高编程效率。Codex等的成功促使了更多类似模型的开发，如StarCoder和CodeLLaMA等。另一种方法是使用ChatGPT、GPT-4等闭源模型辅助代码生成，并用于评估其在生成功能代码方面的有效性。例如微软推出Phi-1通过在高质量数据上进行训练，以13B的参数量获得了与更大规模代码模型相当的性能。

观点争鸣

北京大学教授金芝以软件设计思维作为切入点，分析了软件需求工程的发展前景。她认为，从工程角度看，需求工程分为螺旋式上升的多个阶段，存在沟通不足、知识不足等问题。现有方法能利用

大模型以访谈的形式生成需求信息、进行需求分类、实现需求追踪、分析需求质量、撰写规格说明，等等。大模型具有补充缺失知识、结构化表述需求等能力，其面临的挑战在于如何在合适的时机给出合适的提示。此外，大模型较难区分语气差异，高度依赖提示，还存在其他固有问题。面对这些问题，金芝提出问题描述系统化、AI 智能体协作目标建模、建立智能体合作团队的解决方式。

北京大学教授李戈阐述了大模型技术如何塑造软件自动化的未来，并讨论了在实现高级软件自动化过程中面临的挑战与机遇。他指出软件自动化是提高生产率和质量的根本途径。大模型虽然在自然语言处理方面表现出色，但将其应用于代码生成和软件自动化仍面临诸多挑战，包括模型的准确性、可靠性、安全性以及对私有领域数据的依赖性。李戈预测了大模型在软件开发流程、DevOps 实践、软件测试自动化以及软件维护工作等方面的未来发展。他以“铁钳模型”为喻，强调了自动化的编码和测试在软件开发过程中的重要性，并表达了对大模型引领软件自动化未来的乐观态度。

复旦大学教授彭鑫强调，随着大模型技术的兴起，生成式 AI 可能在不久的将来极大地改变编程和软件开发的方式。AI 在代码补全、代码搜索等智能辅助方面展现出巨大潜力，但也面临着规模复杂性、抽象思维能力、难以捕捉的“暗知识”和长期维护支持等挑战。彭鑫提出，软件开发的根本性困难在于需求和设计的构思，尽管大模型在代码生成方面表现出色，但软件设计和维护仍需依赖良好的模块化设计和专业知识。彭鑫还指出通过加强数字化和知识化积累，利用大模型的记忆、理解和关联能力，实现知识增强的代码数字孪生，提升开发效率和保障软件可信度。未来软件工程工具面临的主要挑战包括如何搭建共建共享的软件开发知识平台，让大模型更好地理解复杂软件系统的全局信息及其业务和技术上下文，以及如何高效地实现软件开发知识的共享和利用，从而应对软件开发过程中的效率问题和可信性挑战。

北京理工大学教授刘辉详细剖析了代码优化在软件开发过程中的关键作用，并深入探讨了大模型技术在该领域的应用前景及其面临的挑战。刘辉强调，尽管代码生成与代码优化在问题空间和解空间上存在相似性，但它们在二义性、变化范围和转换规则上存在显著差异，而大模型的引入为代码优化提供了崭新的视角。当前基于大模型的代码优化面临的挑战主要包括：训练数据的收集、模型的定制与通用性，以及如何确保优化结果的可靠性和突破模型的输入输出限制等。他建议，为了推动代码智能向更高层次发展，需要在数据收集、模型优化、结果可信度以及技术扩展等方面进行深入研究。

大模型与软件质量保障

软件质量保障是确保软件可靠性不可或缺的一环，包含软件测试、代码检视、软件运维等，是整个软件开发过程的重要安全网。近年，大语言模型的出现为软件质量保障的发展带来了新的机遇，提升了软件测试、代码检视、软件运维等领域的研发效能，提高了各种软件系统的稳定性、安全性和性能。

实践分享

针对软件测试，研究者们提出了一系列技术旨在将单元测试代码的生成过程自动化，在一定程度上降低人工书写单元测试代码的开销。LLM 在代码生成等领域已经显示出了不错的成效。考虑到代码生成和单元测试代码的生成本质上都是源代码的生成，近期的研究开始将代码生成的应用范围拓展到单元测试代码的生成上。研究者们采取了对 LLM 进行预训练或微调的方法，例如 A3Test（Assertion Augmented Automated Test Case Generation）用待测方法和断言语句对 LLM 进行预训练，使 LLM 具有更强的断言基础知识；接着针对测试代码生成任务对预训练模型进行微调，使得在减少测试代码生成数量的同时，正确的测试代码数量相比 SOTA 增加了 147%。

代码检视是软件开发生命周期中的重要环节，通常需要开发人员等相关团队人员对代码修改进行

质量评审，在查看和理解代码修改的基础上评估待检视代码的逻辑、功能、风格等因素。在实际开发中，人工检视代码通常花费很多时间。因此，近年来通过智能化的方法（尤其是大语言模型方法）将代码检视工作自动化一直是软件工程领域的研究热点。代码评审通常涉及不同的任务和场景，包括自动评估代码修改的质量、自动生成代码评审的评论，以及自动改进低质量的代码等。目前大模型在不同的代码检视场景中均有应用，主要的技术途径包括：大模型预训练、面向特定代码检视任务上的模型微调、提示工程以及上下文学习等。

软件运维是指利用自动化、标准化和最佳实践管理和维护软件系统的过程，涵盖软件部署、监控、性能优化、故障排除以及安全性管理等方面。现代软件运维需要更多的自动化和智能化工具应对复杂性增加的挑战，以提高效率、节省成本、减少人为错误的发生，以及实现快速检测和响应问题。阿里云团队提出使用工具学习（tool learning）的方式，结合云系统中的代码数据和日志数据使大模型可以自动进行根因分析。微软则提出了一个置信度评估框架，在一定程度上缓解大模型因幻觉导致的输出不可信问题。此外，微软将大模型应用于故障分析的流水线中，以提高数据标注的效率和准确性，辅助理解云服务监控中的常见问题，为特定服务推荐监控策略。华为云团队提出了一种层次化表征方法，使用语言模型对故障工单自动分类，从而节省故障预测中的人工分类成本。

观点争鸣

香港科技大学教授张成志认为，即使给定程序中存在错误，大模型仍然能推断出其意图，但难以准确判断程序的对错、推断预期和实际输出。他研究了利用大模型生成引发错误的测试用例的方法。理论上，大模型可以利用差分测试生成参考程序并将其结果作为用例，但实际上存在参考程序有误、假阳性等问题。张成志由此提出差分提示框架，利用上述预期结果开展差分测试。实验表明，大模型逻辑推理能力较弱，而擅长生成与源码类似的程序。为了补齐逻辑推理的短板，张成志建议将推理问题转换为代码生成问题，并将大模型与其他方法融合。实验表明，大模型能对常见任务生成几乎完全正确的代码，但考虑到代码修复的性能，目前的大模型还不能作为自主可靠的开发助理。

香港中文大学教授吕荣聪认为，虽然自动化运维具有多重优势，但是其在现代软件运维的实践中仍然面临着多重挑战，主要在复杂性、性能、可扩展性这三个方面。大模型为应对这些挑战带来了新的机遇。国内外已有研究表明，大语言模型可以用于自动化故障发现、故障根因定位、事后分析方面，为传统运维模式带来了全新的可能性。大模型驱动的软件运维已逐渐成为新的运维范式，比如TimeGPT模型基于海量时序数据进行预训练，用于多节点的根因定位任务。

武汉大学教授谢晓园讨论了大模型对软件质量保障的意义，分析了大模型在软件测试、缺陷定位、自动程序修复三方面的应用。在软件测试方面，大模型的研究重心由早期的预训练及微调转变为提示设计；在缺陷定位方面，基于大模型的方法可以利用静态信息定位，实现优于传统方法的效果；在自动程序修复方面，部分方法尝试联合大模型与补全工具、利用对话驱动修复等完成任务。谢晓园认为，大模型改进了基于自然交互的人机协同，推动了多个代码评测基准的诞生。但如今的大模型也存在问题，例如能力被高估、对模型的静态评估与实时验证结果仍存在差距。为了解决这些问题，谢晓园建议扩展数据集以实现大模型动态演化，精细化验证大模型输出。她提出大模型仍存在诸多局限性，输入长度与模态、定位能力、输出不确定性等问题限制了大模型在缺陷定位领域的应用。

大连理工大学教授江贺提到大模型驱动的软件工程正引领软件开发进入3.0时代，显著改变了软件工程范式，涵盖流程、工具、指标与价值体系的重构。以大连信华信公司为例，该公司通过团队结构优化与技术革新，利用大模型实现代码自动化与测试自动化，提升了开发效率与经济性。大模型的具体应用包括云上模型训练与云下推理、安全防护、

资源管理等，并在代码生成、测试用例生成与修复上取得成效。然而，大模型仍面临逻辑推理复杂、小语种支持不足、遗产代码现代化，以及一体机部署与应用成本高等问题。学术界与工业界正积极研究应对方法，如通过提示工程提升测试断言精度，并探索自动化脚本生成与缺陷定位修复策略。

代码大模型的数据、评估与验证

大模型的发展推动了数据处理和模型优化等领域的技术创新。根据缩放定律（scaling law），随着模型规模和参数的不断增大，大模型对高质量数据的诉求变得更加急迫，大模型整合了全球各地的数据资源，而数据的多样性、准确性和无偏性要求成为了重要挑战 ；大模型的复杂性和多样性也使模型评估难度增加，传统评估方法可能无法全面评估大模型的性能，需要新的评估框架和指标。

实践分享

数据质量决定模型上限，大模型的性能和能力主要依赖所用数据的质量、数量和多样性。数据集越丰富、干净且覆盖领域越广泛，大模型就越能学习到全面的知识和复杂的模式。数据集的多样性确保大模型能够在不同领域中具有广泛的应用性和通用性。此外，高质量的数据还可以帮助大模型减少过拟合，提高模型的稳健性和泛化能力。因此，确保数据的质量和多样性是训练大模型成功的关键步骤之一，也是让模型在实际应用中表现出色的基础。随着大模型时代思维链（Chain-of-Thought，CoT）、少样本学习（Few-Shot Learning，FSL）、零样本学习（Zero-Short Learning，ZSL）等技术的发展，引导模型在没有专门训练的情况下快速学习，充分利用大模型已经接受过的广泛的知识和技能，让模型可以在有限数据条件下迅速适应特定任务，尤其在数据稀缺或昂贵的场景下减少模型对大量标注数据的需求，可以有效提升数据利用率，降低模型对数据的依赖。

对大模型科学系统全面地评估，是促进大模型迭代重要的一环，影响着模型在实际应用中的性能和可行性。在模型选型阶段，评估可以帮助比较和选择不同的模型，为进一步调优和改进提供参考。在训练过程中，对模型的可解释性和持续监测也是评估的重点，有助于理解模型的决策过程，提升模型在现实应用中的接受度和信任度。在训练完成阶段，通过准确性、精确度、召回率和 F1 得分等指标，可以衡量模型在特定任务上的表现，通过在新数据集上的评估检测出模型的泛化能力，避免过拟合现象等。在模型部署阶段，持续监测模型的表现，以确保模型在不断变化的环境中依然保持良好的性能。此外，评估模型在不同环境和任务下的可靠性和鲁棒性，对模型的伦理性和公平性进行评估，确保模型在面对不同人群时不带有偏见或歧视等，也应得到重视。

观点争鸣

香港中文大学教授吕荣聪对大模型时代下代码大模型数据生成与运用范式进行了详细介绍。他指出，数据是大语言模型的基础，大模型能力从数据中涌现，预训练数据和微调数据都很重要。使用传统的机器学习方法解决某项任务时，需要专业的人员使用高质量的训练数据进行专业的模型训练，使损失函数最小化后得到特定任务的模型，而大模型时代使用单样本 / 少样本学习技术，改变了这一范式。

腾讯公司技术专家王一男分享了代码大模型与代码智能化产品在腾讯内部的评测实践，包括腾讯自研的工蜂 Copilot 工具在代码补全、代码生成、知识问答、漏洞检测与修复、单元测试生成等多个编码任务上的应用和评估实践，对模型在实际编码开发过程中的性能进行评测，有效反映了代码大模型在产业实际项目开发过程中的应用情况。当前产业在模型应用及评测方面遇到的难点、关键点和痛点包括数据工程中存在数据源多、不方便采集、数据质量不好、数据标注效率低、交付速度慢等问题 ；在模型评测阶段难以满足全面、准确、快速的评测诉求，代码模型评估维度、方法及指标不全，人工构建标注数据集耗时、成本高等，影响了模型的迭

代周期；如何将评估结果与模型训练形成有效闭环，提高代码智能化产品的迭代速度和质量也是重要且亟待解决的问题。

华为公司技术专家刘逵指出，研究领域已有各种各样的代码大模型可以用来评估数据集和各种指标，但这些数据集和指标与开发实践效能提升是否强相关有待进一步验证；数据从源头到输入给大模型进行训练，有多个处理环节，虽然从部分开源大模型和研究报告中可以推测出部分数据处理的方案，但不完全透明，无法保证数据源的可追溯；在预训练阶段，多语言多领域的数据组合、配比，对模型的影响仍在探索初期，业界对高质量数据集的定义仍是“未解之谜”。

北京大学副教授熊英飞指出，代码大模型的参数越多，效果就会越好，但是成本也会更高，因此如何在保持模型性能不变的情况下降低参数量，或是在相同参数量的情况下提升模型性能，是值得重点探索的问题。如果按照文本训练的方式训练代码大模型会忽略语法类型等领域知识，使模型生成程序的空间变大，增大了模型的学习难度。他提出，可以借助语法规则序列表示程序，利用规则选择概率生成程序，采用神经网络实现玲珑框架满足语义和类型的程序搜索要求。类似的感知类型规则和文法规则也可以用于大型预训练模型，以提升模型性能。利用程序语法、语义和类型知识可以有效提升模型性能，更通用的编码方式和各种不同的知识在程序合成方面具有巨大潜力。

国防科技大学教授李姗姗详细阐述了代码智能化模型的发展和相关论断，并介绍了代码表示任务和模型泛化性的相关研究。对于代码表示任务，可以使用中间表示解决传统利用字符和抽象语法树（Abstract Syntax Code，AST）难以准确表示源代码的语义的问题。现有代码表示模型尚不具备泛化到其他任务的能力，可以借助多任务联动的方式，共享代码表示提高模型的泛化能力，引入课程式学习有助于模型在不同任务上实现统一的任务提升。适应多编程语言的参数高效微调方法、检索增强和提示模板相组合可以激发大模型在领域特定任务下的代码生成能力。

大模型与开源工程

在信息技术的快速发展中，开源软件和人工智能已成为推动技术创新和产业发展的重要力量。大模型的出现不仅在自然语言处理领域取得了革命性的进展，也对开源生态产生了深远的影响。开源生态，作为一个由全球开发者、用户和组织共同参与的复杂系统，正在经历由大模型驱动的变革。

实践分享

开源软件因其开放源代码、自由共享的特性，成为了软件开发的主流模式之一。开源软件的全球化进程从未停止，即便在社会经济全球化遭遇阻力之时，开源软件仍以其开放性和创新能力在全球范围内持续推动信息技术的进步。开源生态不仅汇聚了企业、开发者、开源基金会、产业联盟和政府等多元参与方，而且随着 GitHub、GitLab 等平台的普及和完善，软件项目以空前的速度增长，形成了一种全球范围内的分布式协同创新模式。截至 2023 年 1 月，GitHub 上有超过 4 亿个项目和 1 亿名开发者。在这样的背景下，人工智能，尤其是大语言模型，以其强大的数据处理和分析能力，正在改变我们与计算机的交互方式，提高软件开发的效率，甚至在某些领域替代人类完成复杂的任务，成为推动软件生态系统变革的重要力量。

随着大模型技术在开源社区的普及和深化，项目间的相互依赖呈指数增长，构建出一张纵横交错、紧密交织且难以追溯的软件供应链网络。在此网络中，每一款软件制品都不再孤立运作，而是作为生态的一部分，与其他制品形成紧密而动态的相互依存关系，共享着资源和安全风险。例如，截至 2024 年 4 月 28 日，通用漏洞披露（Common Vulnerabilities and Exposures，CVE）网站共注册了 228713 个漏洞。因此近年来许多学者和从业人员提出多种技术，从代码和开源社区中感知开源软件中潜在的漏洞和风险，以尽早发现开源软件中的漏洞

从而降低漏洞带来的损失。比如MemVul大模型中包含了一个外部存储模块，用于引入来自通用缺陷枚举（Common Weakness Enumeration，CWE）的外部漏洞知识，从而帮助开源软件用户及时感知潜在漏洞以采取消减措施。

此外，开源大模型因其迭代速度快、成本相对较低、功能强大且易于定制等优点，逐渐缩小了与闭源模型之间的性能差距，甚至在某些方面展现出更为优越的性能和灵活性。这导致越来越多的企业和个人开发者选择投入到开源大模型的研发和应用中，形成了良性循环，加快了整个开源生态系统的演进速度，推动了全球范围内开源软件和AI技术的协同发展和持续创新。随着人机物深度融合的新型社会形态的发展，软件生态的参与者也日益多元化，涵盖了传统开发者、新兴的AI工程师、普通用户，甚至各种智能设备制造商。这使得开源生态的构建和维护变得更具挑战性，同时也为软件生态的持续演化和创新创造了广阔的舞台。

观点争鸣

北京大学教授周明辉以开源全球化、ChatGPT被广泛应用为背景，讨论了大模型对开源工程的影响。“报团取暖”是开源发展的重要驱动力，且开源能有效对抗垄断和脱钩，汇聚群智，实现创新。目前的开源工程面对诸多挑战，例如新兴开源项目挑战已有成熟生态、培育自有生态极为困难；开源软件供应链错综复杂，受到全球的广泛关注。目前的开源工程存在诸多任务，例如度量开源社区和软件供应链、智能推荐新手任务和库迁移方案等。大模型由于其不可解释性，较难直接用于推荐。周明辉还讨论了复杂开源系统度量的未来，认为需要体系化和精细化能力实现供应链管理，实现可视、可管、可控，适应技术生态持续变迁的环境。她倡议产学研深度融合，精细化合作建设开源生态体系。

浙江大学副教授胡星强调了大模型在开源软件漏洞管理中的潜在作用，包括漏洞提前感知、代码级漏洞检测、漏洞依赖分析以及漏洞定位到补丁等多个环节。然而，大模型在实际自动化和协作应用中会面临数据不足、漏洞类型多样以及隐秘修复理解困难等问题。基于这些问题，胡星提出可以借助对比学习技术实现具备可解释性的隐秘漏洞感知算法，通过代码执行路径和函数调用关系提升漏洞识别和修复的准确性和效率，利用人工提取代码和漏洞描述的特征将漏洞定位到补丁。

哈尔滨工业大学（深圳）副教授高翠芸指出，开源数据作为推动软件开发智能化的核心驱动力在代码模型发展过程中发挥着关键作用，同时大模型已经逐渐成为数据使用与生成的核心枢纽。数据规模和质量对于模型的性能表现至关重要，“低质量”数据无序生成和传播，将影响模型的有效合规训练，甚至会污染人类知识库。高翠芸认为，可以通过数据高质量标签生成方法缓解开源数据中数据质量低的问题，利用反事实推理和蜕变测试等手段减缓模型对数据伪特征的依赖。

LLM4SE的未来与挑战

与会嘉宾围绕产学研各方如何分工合作、共同努力实现大模型在软件工程领域健康发展，达成以下共识并发出相应倡议。

共识

1. 利用大模型强大的知识利用能力，强化研发数字化和各类文档知识的价值，解决软件开发中知识的浪费、重复思考的浪费，将大模型作为大规模知识库，对缺失知识进行提示和补全，可有效提升软件开发和知识复用的效率。

2. 代码数据提供的平面化信息不足以支撑大模型获得更高层次的智能化开发能力；大模型推理能力有限，代码生成大多是基于搜索的综合，能够生成是因为“见过”。

3. 一方面要有通用大模型，利用微调或者上下文能力把领域知识展现出来；另外一方面还要有领域大模型，对于大模型不擅长的方面，可考虑构建专门的知识库，提升大模型的应用效果。

4. 大模型为软件生产方式的转变带来了新机遇。

在大模型颠覆技术的当下，积极思考软件生产方式的变化，把软件工程领域的问题变成封闭问题，减少人为引入问题，借助大模型，用可验证的方式进行验证，跳过编码、测试等阶段，探索从需求直接生产代码的自动化方式。

5. 大模型打开了解决问题的新空间，但因为大模型缺乏可信性判断，给软件的可信保障带来了难题与挑战，学术界与工业界要积极思考如何应对大模型带来的可信性保障挑战。

倡议

1. 大模型的成功带来的鲶鱼效应，几乎为所有领域打开了新的探索空间。在条件约束、场景适配、系统化适配等前提下，产学研各方应积极拥抱大模型带来的变化，通过不断自我锻炼实现逐步提升。

2. 大模型时代仍然需要使用知识、推理和因果关系，应用大模型要扬长避短，并将其应用到合适的领域中，实现大模型的持续健康发展。

3. 大模型给各行各业带来了改变与机遇，在任务驱动、数据驱动、模型驱动的大背景下，要持续结合产业界遇到的实际难题与挑战催化学术界对软件工程技术的前沿探索。

4. 通过数据治理、数字化项目解决数据基座问题，在软件开发的数字化和知识化积累的基础上，利用大模型强大的记忆、理解和关联能力加强智能化开发能力，实现效率提升和可信保障。

5. 发展驾驭 AI 的能力 ：能用 AI、用好 AI、用对 AI。对大模型生成的预测性内容进行可信性判断，通过全面提升人的专业能力获得驾驭 AI 的能力。 ■

整理：金　芝　夏　鑫　高翠芸

附：参会专家名单

特邀嘉宾：吕　建　胡事民　吕荣聪

参会嘉宾（按姓氏拼音排序）：

胡　星　江　贺　李　戈　李姗姗　李宣东
刘　辉　刘　逵　马晓星　彭　鑫　王利杰
王一男　谢晓园　熊英飞　张成志　张玉会
周明辉

会议组织：金　芝　夏　鑫

会议秘书：高　姗　高翠芸

（本文责任编委：**胡春明**）