# （一）立项依据与研究内容（4000-8000字）：

## **1．项目的立项依据**（研究意义、国内外研究现状及发展动态分析，需结合科学研究发展趋势来论述科学意义；或结合国民经济和社会发展中迫切需要解决的关键科技问题来论述其应用前景。附主要参考文献目录）；

### 1.1 研究意义

#### 1.1.1智能软件的应用背景

我们正在进入一个智能化时代（互联网+及工业4.0时代），大数据、云计算和人工智能已经成为这个时代进步的三驾马车(图1-1所示)，它们分别为智能化时代提供**数据**、**算力**和**算法**层面的支持，从而成为各行各业技术革新和社会发展的重要引擎。



**图1-1 拉动智能化时代的三架马车**

世界各国为推动智能化进程，特别是美、英、日以及欧盟等发达国家，分别紧锣密鼓地制定了相应的发展战略以及行动计划（表1所示），2016年10月，美国前总统奥巴马在白宫前沿峰会上发布报告《国家人工智能研究和发展战略计划》，12月20日，美国白宫又跟进发布了一份关于人工智能的报告——《人工智能、自动化与经济》，日本政府也先后发布《机器人新战略》和《人工智能技术战略》，世界各国已经把发展智能软件的政策提升到国家战略的高度。

近年来，为促进大数据、云计算和人工智能的发展，我国同样密集出台了一系列发展战略、行动计划和支持政策：2015年，国务院颁布《促进大数据发展行动纲要》，强调数据已成为国家基础性战略资源。李克强总理在两会的政府报告中，提出要“制定互联网+行动计划 ”的要求，推动移动互联网、云计算、大数据、物联网与现代制造业结合，促进电子商务、工业互联网和互联网金融健康发展，引导互联网企业拓展国际市场；2016年，国家发改委、科技部、工信部、中央网信办联合发布了《“互联网＋”人工智能三年行动实施方案》，提出了三大方向共九大工程，系统地提出了我国在2016-2018年间推动人工智能发展的具体思路和内容；2017年，国务院颁布《新一代人工智能发展规划》，指出要抢抓人工智能发展的重大战略机遇，构筑我国人工智能发展的先发优势，加快建设创新型国家和世界科技强国；同年，工业化信息部发布了《促进新一代人工智能产业发展的三年行动计划（2018-2020年）》，目的在于深入实施“中国制造2025”，加快人工智能产业发展，推动人工智能和实体经济深度融合，力争于2020年在一系列人工智能标志性产业取得重要突破，在若干重点领域形成国际竞争优势。

**表1 世界各国人工智能政策**



#### 1.1.2智能软件的广泛应用

在当前全面推进战略性新兴产业及高技术制造业建设的形势下，智能软件为提升现代企事业单位生产力水平提供重要支撑，为国民经济的飞速增长和社会的持续稳定发展提供有力保障。**智能软件是指能够产生人类智能行为的软件系统，通常通过学习或者自适应等方式获得处理问题的逻辑，具有强大的认知和问题解决能力，正在推动经济社会从数字化、网络化向智能化加速跃进。**在智能化时代背景下，大数据提供的海量数据、云计算带来的超强计算能力以及人工智能算法的不断演进为智能软件的飞速发展插上了腾飞的翅膀。

目前，**智能软件在各个领域表现良好甚至达到了人类的水平**，谷歌公司的AlphaGo围棋智能机器人依靠深度学习技术战胜了排名世界第一的围棋冠军柯洁及职业九段棋手李世石；IBM 公司的深蓝智能计算系统战胜了国际象棋特级大师加里·卡斯帕罗夫。人工智能技术还可以提升研究人员发现和解决问题的能力，助力科学研究与发现，造福人类。谷歌最新的人工智能AlphaFold，在一项极其困难的任务中击败了所有的对手，成功根据基因序列预测了生命基本分子-蛋白质的三维结构[9]；医学影像企业Enlitic开发了从X光照片及CT扫描图像中找出恶性肿瘤的图像识别软件，利用深度学习的方法对大量医疗图像数据进行机器学习，自动总结出病症的“特征”以及“模式”。智能软件已经渗透到我们生活的每个角落，正在逐渐改变家居、出行、医疗、教育、金融、工作等诸多领域（表2所示）。

**表2 智能软件的应用实例**

|  |  |
| --- | --- |
| **智慧领域** | **应用实例** |
| 智能家居 | 智能家电：人工智能技术丰富家用电器的功能，如三星公司的智能冰箱Family Hub、亚马逊的Echo智能音箱......  家居智能控制平台:智能控制门、窗、家用电子设备等，如谷歌的Google Home家庭设备的控制中心、扎克伯格的“贾维斯”智能管家、苹果的HomeKit智能家居平台...... |
| 智慧出行 | 无人驾驶汽车，如Google Driverless Car；  智能交通机器人，用于路口指挥交通，降低交通警察工作量；  智能交通监控可应用于停车场、高速路口收费站、车辆抓拍。 |
| 智慧工作 | 目前在服务行业有了广泛应用，如京东的无人仓储依靠智能控制系统实现了6倍于人类的思考决策速度及 10倍于传统人工仓库的货物处理效率；微软亚洲研究院利用智能软件来优化现有的航运操作，改善航运业网络运营，提升经济效益...... |
| 智慧医疗 | 智能医疗机器人：用于外科手术、功能康复及辅助护理等方面，如Verb Surgical公司，研发新一代辅助手术的机器人；智能药物研发，如IBM的Watson机器人；智能诊疗与智能影像识别；  智能健康管理，如Welltok的健康管理平台。 |
| 智慧教育 | 网络课程将占据主流，人机交互可以在线答疑，通过图像识别，可以进行机器批改试卷、识题答题等，教育资源丰富且共享。 |
| 智能金融 | 利用递归神经网络准确预测股票价格，客户信誉度量和风险管理方面也有应用。 |
| 智能零售 | 凭借丰富的客户数据，人工智能可以应用在定制产品推荐、购物助手、实时价格调整、库存管理、订单分配以及面向聊天机器人的客户服务等方面。如微软合作商推出Fellow Robots服务机器人，可识别产品并验证产品是否带有正确的标签，以及产品是否处于正确位置或者是否被错误的移动等来提高工人拣货的效率。 |
| 智能电商 | 用于电商平台的商品管理，创建个人推荐，预测商品价格，提高产品图像质量以及识别可疑广告和行为等，如Ubcoin Market。 |
| ...... | ...... |

#### 1.1.3智能软件的质量与安全隐患

智能化软件系统为人类带来了极大的便利，然而智能化软件系统也并非是完美的，在性能和安全可靠性上也会遇到种种挑战，软件安全是软件领域的一个重要子领域，已经成为评判软件质量的重要标准。单机时代的软件安全主要是操作系统容易感染病毒，而互联网普及以后，软件安全问题尤为突出。软件安全性是软件在受到恶意攻击时仍提供所需功能的能力。当前软件开发人员普遍缺乏必要的 安全意识和知识，开发的软件大多存在安全缺陷，而且不断增加的软件复杂性和可扩展性需求更凸显了软件安全问题的严重性。安全性相关缺陷不同于一般的软件缺陷，一个很难发现的软件缺陷可能只影响少部分用户或者产生不十分严重的影响，而一个很难发现的软件安全漏洞可能导致大量用户受到影响或产生极为严重的影响。像**智能化软件系统这类安全关键软件一旦发生故障，则可能造成 重大人员伤亡、财产损失、环境污染等危险事故**。下面列举了一些近年来智能软件系统发生的故障案例（如表3所示）。

**表3 智能软件故障案例**

|  |  |  |
| --- | --- | --- |
| **智能应用** | **案例描述** | **故障原因** |
| 亚马逊Alexa | 小孩向亚马逊Alexa下达播放“Digger digger（一首儿童歌曲）”的指令，亚马逊个人助手通过算法进行识别，竟然认为孩子想听情色内容，并开始播放。 | Alexa很难完全屏蔽成人内容，且很难鉴别是否是孩子在使用。 |
| Echo音箱 | 亚马逊Echo智能音箱会莫名其妙的自己发出难听的笑声，一个德国人的Echo音箱在他不在家的时候被意外地激活了，在午夜之后开始播放音乐，吵醒了邻居，邻居无奈报警。 | 笑声有可能是由于指令没有成功传达造成的误报，修复方案将包括不再使用“Alexa, laugh”这个命令短语。 |
| 微软旗下的聊天机器人Tay | Tay被网友教的种族歧视、反女权，各种爆脏话，不但说自己喜欢希特勒，还说911事件是小布什所为。 | 有人利用Tay无法甄别信息的虚假性漏洞发动了攻击，采用一些虚假谎言对Tay进行训练。 |
| 谷歌的无人驾驶汽车 | 2016年，谷歌的无人驾驶汽车和一辆公共汽车相撞。 | 系统错误的预测公共汽车会在一系列罕见的条件下减速或停车让行，而实际上公共汽车不可能停止 |
| 特斯拉的无人驾驶汽车 | 2016年，特斯拉的一辆无人驾驶汽车和一辆拖车相撞。 | 拖车的外表颜色和天空相近并且底盘较高，导致系统将一辆白色卡车误检测为天空。 |
| 医疗机器人“达芬奇” | 2015年“达芬奇”医疗机器人在**心瓣修复手术中**把病人的心脏“放错位置”，并戳穿其大动脉，病人最终由于多器官衰竭逝世。 | 测试无法真实模拟病人的情况下，所以难以全面测试系统所有可能的情况。 |
| 无人机 | 2017年日本一架无人机因操作失误在坠落的过程中砸伤一名工人 | 无人机飞行时受到无线电信号干扰，致使飞行器跟遥控器失联，导致失控。 |

类似的案例还有很多，这些案例很多都是智能化软件自身漏洞导致的，这些漏洞或许导致软件的失控，如Echo音箱莫名启动，但也有可能在一些安全关键的领域（如无人驾驶、航空航天等）带来灾难性影响，软件质量缺陷造成的损失往往是难以承受的。因此，在智能化软件系统充溢生活的今天，保障智能化软件的质量是一个意义深远的问题。**如何有效保障智能化软件系统正确、高效、可靠地实现其既定任务是一个需要解决的重要问题。**

目前，很多智能化系统都以软件作为其控制核心，软件规模和软件的复杂度与日俱增，例如在F-22战机中的综合航电系统中,软件实现的航电功能高达80%,软件代码达到170万余行。而在F-35战机的先进综合航电系统中，软件代码达到500-800万行。这些安全关键的控制软件的失效往往会给用户或者国家财产造成巨大的损失。影响安全关键软件可靠性的因素，主要包括两个方面：一是安全关键软件通常被用于严酷环境中，异常外围环境可能导致硬件故障，进而影响软件的执行；另一方面，软件自身的设计和实现缺陷可能导致软件失效。因此对这些系统在投入使用之前一定要进行充分的测试，消除系统内可能存在的故障和漏洞，同时也要在使用之前模拟未来使用环境中可能出现的环境扰动，来对系统的可靠性等指标进行充分的评测，根据测试结果采用必要的设计和防护方法，保证其满足可靠性要求。但是由于很难真实模拟严酷的运行环境和未来使用环境中可能出现的环境扰动，所以，软件漏洞的存在往往是不可避免的。因此，软件安全漏洞检测应成为智能化软件测试的一个重要组成部分，如何快速而准确地发现软件安全漏洞一直以来是软件安全领域的研究热点，也是智能化软件测试关注的一个难点。

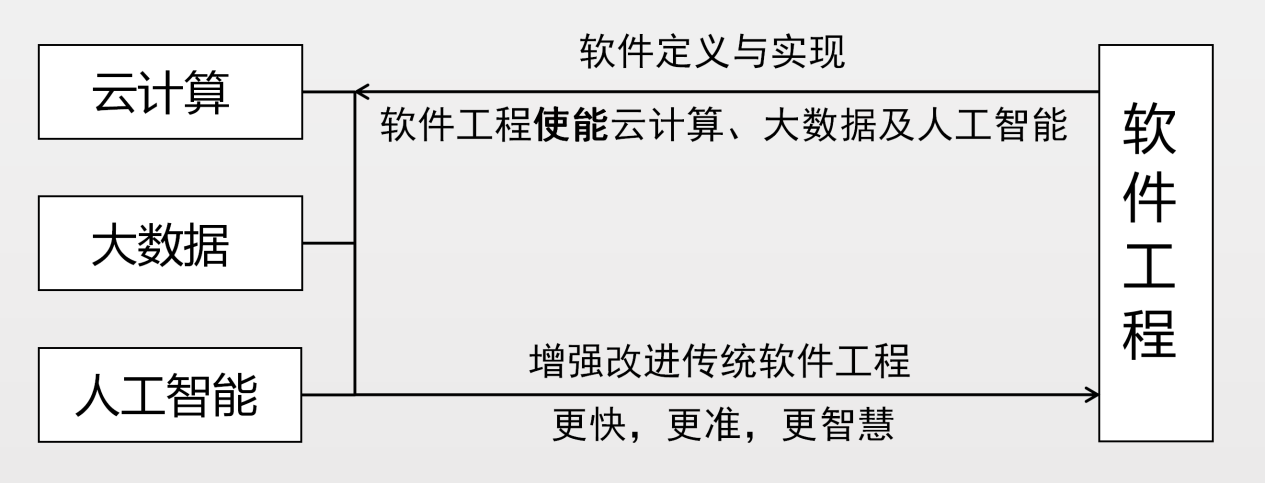
代码和缺陷报告作为软件开发和维护的重要组成部分，能够为软件安全漏洞检测提供重要的信息来源。通过软件测试的方法发现出的软件缺陷通常是以缺陷报告的形式记录的，尽管缺陷报告在被提交时需要被标明该缺陷是否是安全漏洞，但是由于对信息安全知识的欠缺，报告提交者往往无法正确的区分安全漏洞与常规的软件缺陷，因此许多安全漏洞常常被错误地标记为与安全无关的缺陷，从而导致这些安全漏洞无法被及时地修复。软件安全漏洞是在软件设计与实现过程中存在的一些容易被恶意攻击者所利用或有可能影响软件可靠运行的缺陷或不足。若Security相关的漏洞无法被及时地识别和修复，则容易被恶意攻击者所利用，使得系统被入侵进而造成系统运行结果不可靠，或者任意命令执行、任意文件读取等严重安全问题。例如人脸识别账号被盗刷、智能家居被入侵等，2017年央视315晚会上技术人员仅凭一张观众的自拍照，借助人脸关键点定位和自动化人脸动效技术，通过将自拍照由静态改为动态，可以完成刷脸登录需要的眨眼、微笑动作，成功破解了刷脸登录的人脸认证系统，说明像这种使用人脸识别的APP等智能软件也都存在被盗用的风险，使得个人的安全隐私受到威胁。若Safety相关的漏洞无法被及时地识别和修复，则会导致系统不能可靠地运行，从而有可能引发严重的安全事故。可见，程序代码一旦存在安全缺陷，就会造成比普通缺陷更为严重的影响。因此需要尽早识别、定位和修复安全相关的漏洞，使自动驾驶、医学成像等智能软件系统的部署和运行更为安全可靠。因此，**软件安全漏洞检测对于保障智能软件这类安全关键（safety-critical）系统的安全可靠具有十分重要的意义。**

### 1.2 国内外研究现状及发展动态分析

#### 1.2.1软件工程的智能化发展趋势

从1969年开始,人们就已经提出软件工程的概念,研究如何以最经济的方式，在最短的时间内开发用户最满意的软件产品。软件工程是一门研究用工程化方法构建和维护有效的、实用的和高质量的软件的学科。它涉及[程序设计语言](https://baike.baidu.com/item/%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E8%AF%AD%E8%A8%80/2317999)、[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)、[软件开发工具](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%BC%80%E5%8F%91%E5%B7%A5%E5%85%B7/4605523)、系统平台、标准、[设计模式](https://baike.baidu.com/item/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F/1212549)等方面。现代社会中，软件应用于多个方面。典型的软件有[电子邮件](https://baike.baidu.com/item/%E7%94%B5%E5%AD%90%E9%82%AE%E4%BB%B6/111106)、[嵌入式系统](https://baike.baidu.com/item/%E5%B5%8C%E5%85%A5%E5%BC%8F%E7%B3%BB%E7%BB%9F/186978)、人机界面、[办公](https://baike.baidu.com/item/%E5%8A%9E%E5%85%AC)套件、[操作系统](https://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F/192)、[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067)、[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728)、[游戏](https://baike.baidu.com/item/%E6%B8%B8%E6%88%8F/33581)等。同时，各个行业几乎都有计算机软件的应用，如工业、农业、银行、航空、政府部门等。这些应用促进了经济和社会的发展，也提高了工作效率和生活效率。

信息化时代下，由软件定义与实现的云计算、大数据和人工智能正蓬勃发展，悄然的改变着这个世界，极大的解放了生产力，进一步提升了工作和生活效率，同时云计算、大数据和人工智能的发展也深刻地改变着软件工程领域（图1-3所示），以下我们简单回顾一下这些改变，展望未来的软件工程的智能化发展趋势。



**图1-3 软件工程与“三驾马车”**

社会需求推动了云计算、大数据与人工智能产业的蓬勃发展。信息化时代下，软件及其用户无时无刻不在产生海量数据，如开发过程中的源代码、需求文档、缺陷报告、测试用例；系统运行中的运行日志、性能度量、事件记录；用户交互中使用行为序列、调查问卷、社交媒体反馈等等。随着互联网软件服务的普及，数据的种类更加丰富，规模也越来越大。**大数据的发展为我们提供了科学的数据采集、数据分析与数据处理流程与方法，云计算提供了安全的数据存储中心，可靠的数据计算平台和强大的数据计算能力，人工智能的发展同样离不开数据，科学的数据训练、数据挖掘与数据预测方法可进一步提取出海量数据中潜在的价值，增强传统软件工程的能力。**

智能化是未来软件工程发展的新趋势，欧洲信息技术研究计划（ES-PRIT）之前就曾提出要把人工智能技术和软件工程技术紧密的结合在一起，从而可以开发出一套比较有效的工具，这个工具同时也可以支持软件系统的具体分析和设计工作。而且近年来越来越多的研究也表明，人工智能技术和软件工程技术的联合发展是非常有必要的，同时也必将引起软件开发方法和软件程序管理模式的改变，这样就可以形成一个新的开发和管理规范，而人工智能技术也可以使软件的开发更加容易、更便于修改和维护。智能软件工程主要特征包括：

（1）需求分析自动化。针对软件项目中需求模糊，模型抽象，功能杂乱等难题，利用人工智能分类算法，Topic Model和自然语言处理相关理论进行处理。

（2）代码分析自动化。可以利用人工智能中卷积神经网络等方法实现自动化检测代码缺陷，甚至可以自动写代码。

（3）软件测试自动化。利用深度学习方法可以智能化选择合理的测试技术与方法对软件进行全方位、多角度的测试。

（4）故障诊断自动化。针对日志数据，挖掘其中有效信息，利用深度学习方法和聚类技术，可以自动的诊断定位故障。

（5）故障修复自动化。对于诊断定位的故障，利用历史修复策略等信息自动修复故障，实现故障处理的智能化。

大数据环境下的智能软件工程是未来软件开发的新方向，计算机软件正在重新定义和实现着整个世界。不久的将来，人工智能系统将替代各种软件进入人们生活的每个角落。人工智能和软件工程将空前的紧密融合在一起，一方面，软件工程通过人工智能的应用变得更加强有力，另一方面，软件工程为无处不在的各种人工智能系统开发和应用保驾护航，推动智能化时代的健康有序发展。

#### 1.2.2软件测试方法体系的研究现状

整个软件工程可以说是一个不断与软件错误和缺陷做斗争的过程，为了生产高质量的软件，软件工程为软件质量把好下图所示的“四道关”（图1-4所示）。第一道关是利用形式化方法[34-38]、高度集成的软件开发环境和各种支持工具等在软件需求分析、设计和编码阶段严防死守，预防可能出现的各种问题和错误，这一步虽然非常重要和有效，但一般不可能消灭所有错误，仍有很多错误防不胜防而隐藏在软件中[17]。第二关就是要通过软件测试方法来发现和纠正这些没有防得住而隐藏在软件中的错误。即使软件测试技术已非常成熟，我们也很难保证通过第二关软件测试之后，软件中就不可能再有错误，所以，我们需要第三关，通过容错计算技术，在软件系统中植入容错能力，使得即使在前两关中没有防得住的错误、没有检查出来的错误存在于软件系统，也不至于给系统造成重要损失。第四关是在前面三道关的基础上，对软件系统仍然存在错误风险的预测，即软件可靠性的研究。

第一关

预防错误

（形式化方法）

第二关

检测错误

（**软件测试**）

第三关

容忍错误

（容错计算）

第四关

预测错误

（软件可靠性）

**图 1-4 软件质量“四道关”**

软件测试是指以发现错误、度量和提高软件质量为目的而理解、分析和运行程序的过程。软件测试的目标是以最小的成本，及时准确地发现软件中隐藏的错误，从而提高软件质量，降低风险。人们根据软件开发的不同阶段、不同的测试对象、基于不同的理论和角度提出了一系列软件测试方法，形成了图1-5所示的软件测试方法体系，具体可以有如下几类：

（1）软件测试根据是否运行程序可分为静态测试[2]和动态测试[4]。静态测试包括桌面检查、代码审查和代码走查等方法。动态测试根据测试用例设计是否依据程序内部结构可以分为黑盒测试和白盒测试[2]，白盒测试包括语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖、线性代码序列及跳转测试等；黑盒测试包括等价类划分、边界值分析、因果图分析、错误猜测、状态转换测试等[3，40]。

（2）根据软件开发的不同阶段可以将软件测试划分为：单元测试[10]、集成测试、系统测试、验收测试、回归测试、验证测试、确认测试、Alpha测试、Beta测试和Gamma测试等[4]。

（3）根据被测试软件的开发方法和应用环境的不同可以分为[1,4,20]：面向对象软件测试[6]、面向方面软件测试、面向服务软件测试[15]、构件软件测试[5]、嵌入式软件测试[42]、Web 应用和网构软件测试[41]等，后面还要出现普适计算环境下的软件测试、云计算[7]和物联网环境下的软件测试等。

（4）根据软件不同特性和方面的测试可以分为[1,4]：负载测试、压力测试、性能测试[14]、安全性测试、安装测试、可用性测试、稳定性测试、授权测试、用户接受性测试、一致性测试、配置测试、文档测试、兼容性测试和Playtest（试玩测试）[37]等。

（5）根据不同特殊的测试技术可以有：组合测试[8,43]、蜕变测试[19]、变异测试[22]、演化测试[9，11]、FUZZ测试[29]、基于性质的测试[18]、基于故障的测试[1]、基于模型的测试[17]、反模型测试[28]、结对测试[31]、在线测试[32]、基于操作剖面的测试[20]、基于用例和／或用户陈述开发测试用例、基于规格说明的测试[23]、统计测试[21]、逻辑测试[39]、随机测试[16，25]、反随机测试[26，27]、自适应随机测试[12，24]、GUI测试[30]、冒烟测试和探索测试[33]等。



**图 1-5 软件测试方法体系**

可见，传统的软件测试方法体系已经非常完善，从不同开发方式和应用场景、不同特性和方面、不同开发阶段、静态与动态、黑盒与白盒等多个角度对测试方法进行了细致的区分与总结，每一种测试方法都有其对应的测试场景和其不可取代的测试价值。我们对几十种测试方法进行了系统的梳理与归纳，尤其是特殊的软件测试方法，如组合测试、蜕变测试、变异测试等，在软件测试领域积累了丰富的测试经验与扎实的理论基础。

#### 1.2.3软件测试两大关键问题

**软件测试是一种广泛采用的软件质量保证手段**[1]，通过运行有限的测试用例，比较测试用例的输出与预期输出是否一致来检测软件中潜藏的故障。软件测试包含两项关键的任务，即**测试用例的生成**和**测试结果的判定**。

##### 1.2.3.1 测试用例的生成

测试用例生成为待测软件产生用于检查程缺陷的软件输入及其对应的预期输出，直接影响着软件测试的有效性和效率。代表性的软件测试方面的研究工作包括[2]：

**（1）基于符号执行的测试用例生成：**使用符号值代替具体值分析程序所有可能的执行路径，并通过约束求解为执行路径生成测试用例[3]。符号执行技术由King等人[4]首次提出，其初衷是使用符号值执行程序并以此分析程序所有可能的行为。然而，该技术受制于困难的约束求解过程和庞大的程序分析开销。为解决约束求解难的问题，研究者开发了多种约束求解器，如z3[5]、STP[6]等；为解决程序分析开销大的问题，研究者提出了选择符号执行[7]、动态符号执行[8]等。此外，研究者通过符号执行生成高覆盖率的测试用例：Cadar等人[9]设计并实现了符号执行工具KLEE，可以为待测程序生成超过90%代码覆盖率的测试用例，Godefroid等人[10]在符号执行的约束表达式中引入了符号变量在执行中应满足条件的相关描述，提升了白盒测试中用例生成的有效性与效率。

**（2）基于模型的测试用例生成：**使用待测程序的形式化模型（有限状态机、UML模型等）自动为待测程序生成测试用例[11]。基于有限状态机的测试用例生成方法将待测程序所有可能的状态进行抽象，定义状态转换所涉及的输入及输出，构造待测程序的有限状态机模型，最后以状态覆盖、转移覆盖、路径覆盖等策略从状机模型中选择状态转移序列并生成测试用例[12]。例如， Kansomkeat和Rivepiboon [13]使用UML状态图为待测程序构建模型并自动生成满足一定覆盖条件的测试用例集。2018年Xue Han等人[14]分析三个大型开源项目中的300个错误报告,他们发现输入参数和配置的组合可以暴露性能错误，首次提出PerfLearner，根据错误报告自动生成测试帧，这是一种提取测试框架元素的自动化方法并为性能测试生成测试框架以指导实际性能测试用例生成。实验证明其能生成测试框架和检测实际性能错误。

**（3）基于组合测试的测试用例生成：**使用不同的组合策略将待测程序不同参数的值进行组合，为待测程序生成规模尽可能小而不显著损失故障检测能力的测试用例集（组合覆盖表）。目的是使用较少的测试用例分析软件中各参数之间的相互作用对整个系统产生的影响。目前，组合测试中主要有四类测试用例生成方法，包括贪心算法、启发式搜索算法、代数方法、随机方法。贪心算法以尽可能多地覆盖未覆盖组合为原则给待测程序逐一生成测试用例并形成测试集，典型的方法有one-row-at-a-time [15]及in parameter order 算法[16]。启发式规则对预先存在的测试集进行一系列转换，直至测试集覆盖所有的组合，典型的方法有Ghazi等人[17]提出的基于遗传算法的技术及Bryce和Colbourn[18]在贪心算法基础上改进的技术。代数方法通过数学函数或递归构造的方式推导测试用例，典型的方法有[19]。随机方法则以随机的方式从测试全集中选择测试用例，通常被作为实验对比的基准方法[20]。

**（4）适应性随机测试：**使新生成的测试用例与所有已生成测试用例具有最远的距离，使测试用例在软件输入域上更加均匀地分布。Chen等人[21]在此想法的基础上提出了适应性随机测试，改进了随机测试的故障检测能力。此后，研究者在最远距离测试用例选取方面提出了多种不同的策略，形成了适应性随机测试的多个变种。Chen等人[21]从已有的候选随机测试用例中选取与已生成测试用例差别最大的测试用例，并提出了基于固定规模候选集的适应性随机测试（FSCS-ART）。Chen等人[22]提出了镜像适应性随机测试（MART），通过在输入域中选择与已有测试用例“镜像”的新测试用例，实现最远距离测试用例的选取，减少了适应性随机测试的计算开销。Chen等人[23]提出了基于随机分区的适应性随机测试，使用已执行测试用例对程序的输入域进行划分，并从最大的分区中选取测试用例。

**（5）基于搜索的测试用例生成**：使用搜索算法自动化从待测程序的输入空间中寻找能够最大程度满足测试目标且最大限度降低测试成本的测试用例集。在程序结构测试方面，研究者针对多种程序覆盖准则提出了不同的测试用例生成技术[24]，如分支覆盖[25]、判定覆盖[26]、数据流覆盖[27]等。在基于模型的测试方面，程序模型的状态迁移覆盖一直是研究的热点，代表性的工作有：Li等人[28]为UML状态图表达的程序模型生成满足状态迁移覆盖的测试用例集，Lefticaru和Ipate[29]使用遗传算法自动为程序的状态图生成测试用例集。在变异测试方面，研究者通过搜索的手段生成能够杀死待测程序变异体的测试用例[30].

**（6）动态随机测试技术：**Cai等人将控制论引入到软件测试中，提出了利用测试过程的历史信息更新测试剖面的适应性测试技术[31]，但是适应性测试技术的时间开销太高。为了降低适应性测试技术的时间开销，Cai等人根据引起故障的输入趋向于集簇在连续的区域这一观察[32]，提出了利用当前的测试信息更新测试剖面的动态随机测试技术[33]。该技术的主要特点是在测试的过程中根据每一次的执行结果动态改变测试剖面，使得具有较高失效率的分区的选取概率变大。动态随机测试的性能受两方面的影响：初始测试剖面和动态随机测试中的参数值。这方面代表性工作包括：（a）初始测试剖面：Li等人提出了在测试过程中出现预定标准时测试剖面转换成理论最优剖面的最优动态随机测试技术[34]。（b）参数值：Yang等人提出了在测试的过程中动态调整参数的方法[35]；Lv等人通过理论分析的方式分析两个参数的比值在一定的区间时可以保证失效率大的分区选取概率不断增大[36]。

##### 1.2.3.2 测试结果的判定

测试结果判定是检查测试用例对应的实际输出是否符合预期的步骤，其结果是判断待测软件中是否潜藏故障的重要依据。然而，在很多实际情况中，难以验证测试用例对应的实际输出结果是否满足预期，此问题称为软件测试中的“测试预期问题”。测试预期问题一直是软件测试领域中的热点与难点问题[37]，代表性的研究工作包括：N-version 测试[38]、断言[39]、机器学习[40]、假设检验[41]、蜕变测试[42]。其中，蜕变测试依据待测软件的蜕变属性获取蜕变关系，执行原始测试用例（采用测试用例生成技术获得）与衍生测试用例（根据蜕变关系获得），检查对应的输出是否违反蜕变关系。如果违反了某种蜕变关系，则表明存在故障。不难看出，蜕变测试无需测试预期，因此有效地缓解了测试预期问题，并在多个领域得到成功的应用[43]。近年来，研究者在蜕变测试理论与应用等方面取得了丰富的研究成果。部分代表性工作如下：

**（1）蜕变关系识别与复合方面：**Chen 等人提出了一种基于范畴划分的蜕变关系识别方法[44]；Zhang等提出一种基于搜索的蜕变关系推理方法[45]；Su 等人提出了一种蜕变关系动态识别方法[46]；Sun 等人提出一种数据变异指导的蜕变关系获取方法[47]；Gotlieb 等人提出了一种蜕变关系验证框架[48]，对于给定的程序的某个蜕变关系植入故障，利用约束逻辑编程技术生成满足植错后的蜕变关系的测试用例集，如果测试用例集检测出故障，则表明植错后的蜕变关系是错误的；Liu 等人提出一种蜕变关系复合方法，通过对多个蜕变关系的复合形成新的蜕变关系[49]。

**（2）蜕变测试原始测试用例生成方面：**Batra等人提出了一种基于遗传算法的测试用例生成方法，旨在最大化覆盖程序路径[50]；Dong等人采用符号执行提取蜕变关系并生成相应的测试用例[51]；Chen 等人比较了特殊值法与随机方法产生的原始测试用例对蜕变测试效率的影响[52]。

**（3）蜕变测试与其它测试技术结合：**Xie等人将蜕变测试与基于频谱分析的故障定位技术结合，提出了无需预期的故障定位方法[53]； Dong 等人将蜕变测试与遗传算法相结合，在搜索过程中使用蜕变关系，将原始测试用例和衍生测试用例都视为候选方案，加速达到某种覆盖标准[54]。

**（4）蜕变测试在不同领域中的应用：** Sun 等人提出了面向 Web 服务的蜕变测试方法 [55]，首先从 Web 服务的 WSDL 描述文档中提取蜕变关系，然后根据 WSDL 文档随机产生测试用例并且运用蜕变关系得到相应的衍生测试用例；Chan 等人提出了一种面向SOA 的蜕变测试方法，将离线测试通过的测试用例作为原始测试用例，在线测试用例作为衍生测试用例[56]；Mayer等人将蜕变测试应用于图像处理程序[57]；Kuo等人将蜕变测试应用于图像处理程序时检测出一个真实故障[58]；Tse等人尝试将蜕变测试应用于上下文敏感的中间件软件[59]；Chan 等人将蜕变测试应用于能量感知的无线传感器网络应用软件[60]；Sun 等人将蜕变测试成功地用于几类典型加密程序的测试[61]。

#### 1.2.4 智能软件测试的研究现状

智能软件近年来受到学术界的广泛关注并得以快速发展，然而智能软件带给人们巨大便利的同时，其本身也存在一些隐含缺陷或者安全问题。对于一个非正常的输入，智能软件是否依然能够产生满意的结果，关于智能软件测试的重要性也渐渐引起研究者的注意，我们简单回顾一下代表性的对智能软件进行测试的研究工作（表4所示）。

**表4 智能软件测试代表性工作**

|  |  |  |
| --- | --- | --- |
| **测试框架** | **简单介绍** | **特点** |
| DeepXplore | Kexin等人提出针对真实深度学习系统(DL)进行系统性白盒测试的框架。 | 旨在解决两个难题：生成能够激发一个深度学习系统逻辑的不同部分的输入和自动识别深度学习系统的不正确行为。 |
| DLFuzz | Guo提出用于指导DL系统暴露不正确的行为的模糊测试框架，是最先进的DL白盒测试框架。 | 通过不断改变输入以最大化神经元覆盖，以及预测原始输入和变异输入的差异，无需手动标记或交叉引用来自其他DL系统的oracles。 |
| DeepTest | Tian等人提出的以神经元覆盖为导向的测试用例生成框架。 | 利用神经元覆盖的概念，对图像的转换生成增加神经元覆盖率的合成测试用例，在合成图像的基础上进行再训练，尽可能地得到更贴近现实状况的图像，以贴合真实的使用场景。 |
| DeepCover | Sun等人提出基于DNN测试标准的测试框架。 | 受传统MC/DC覆盖标准的启发，依照传统软件测试方法中的测试覆盖标准和测试用例生成算法的思想，针对深度神经网络(DNN)的不同特征提出四个测试标准。对于每个标准，章给出一种基于线性规划生成测试用例的算法。 |
| DeepGauge | Ma等人提出针对DL的多粒度测试覆盖方法 | 对DNN进行行为划分，对DNN内部状态细致观察，提出了神经元粒度的覆盖标准和神经网络层粒度的覆盖标准，指导测试用例生成。 |
| DeepConcolic | Sun等人提出了第一个用于深度神经网络（DNN）的分析测试方法 | 结合程序执行和符号分析，以探索软件程序的执行路径。使用各种覆盖指标指导生成测试输入，实现的一个导向性随机测试工具，具有高覆盖率并且能够发现大量的对抗样本。 |
| DeepCT | Ma等人提出的用于DNN的组合测试框架 | 提出一套用于DL系统的覆盖标准，以及用于CT覆盖引导的测试生成技术。 |
| DeepMutation | Ma等人提出的用于DNN的变异测试框架 | 从数据、源码级别和模型级别角度设计变异算子。通过对数据引入变化、对源码进行变异操作、对模型注入错误等方法，将故障引入DL模型。最终可以从分析中评估测试数据的质量，并可以检测注入的故障程度，进而对测试数据质量进行改善。 |
| DeepRoad | Zhang等人提出的针对自主驾驶系统的变异测试方法 | 通过生成具有各种天气条件的驾驶场景，以检测自主驾驶系统的不一致行为。 |
| FITEST | Abdessalem提出了一种通过将检测特征交互的问题转换为基于搜索的测试生成问题，进而检测特征交互是否失败的技术。 | 一种由混合测试目标指导的基于搜索的测试生成算法，定义了一组混合测试目标（距离函数），将传统的基于覆盖的启发式方法与专门用于揭示特征交互失败的新启发式相结合。 |

由表4可见，目前针对智能软件的测试思路主要有两种。一种是基于智能软件的内部状态和行为分析探索出的测试用例生成算法，另一种则是将传统软件测试方法与技术应用在智能软件的测试中来，如利用组合测试的DeepCT框架，利用变异测试的DeepMutation框架。两种测试思路都是智能软件测试的重要尝试。深入研究之后，我们也发现两种测试思路在测试的充分性与普适性上存在局限，上述的测试框架可能在某一领域或某一场景下，测试效果很好，如对深度学习系统的充分性测试，然而对于其他的智能软件而言，测试的契合度可能不高。

#### 1.2.5软件安全缺陷识别的研究现状

软件系统的安全性是软件系统在应对非正常的操作或恶意攻击情况下依然能够满足可用性、授权访问、数据的机密性等需求的关键。软件开发者和组织为了提高软件系统的安全性，研究提出了一系列软件开发安全原则和最佳实践，然而由于程序的复杂性和软件的广泛应用，针对软件的恶意攻击行为和软件系统安全缺陷越来越多地涌现出来，成为威胁软件系统安全的主要原因之一。

目前关于软件缺陷的研究主要集中在以下两方面：

**1.软件的脆弱性检测（Security vulnerability detection）**

软件系统在设计、实现、操作、管理上存在的可被利用的缺陷或弱点，称为软件的脆弱性。软件脆弱性分析是检查并发现软件系统中的内在安全缺陷和利用途径的主要手段之一，一直是软件安全领域的研究热点。传统的软件脆弱性检测主要依靠程序分析技术，如静态分析和动态分析，然而静态分析技术普遍存在误报率较高以及过分依赖检测规则的缺点，而动态分析技术则由于难以覆盖全部路径导致漏报问题并且有较大的时间开销和空间开销。近几年，面对静态和动态脆弱性分析方法的技术瓶颈，研究者开始重视软件漏洞或者脆弱性历史数据的挖掘与分析，并且随着开源代码量的增加，利用深度学习技术从大量的代码中学习软件漏洞模式也成为了可能，这为代码漏洞检测的研究提供了新的方向。国内外对于代码漏洞检测技术已经有了很多研究，这些技术大体上可以分成两类：基于程序分析的方法和基于统计模型的方法。

（1）基于程序分析的方法

基于程序分析的漏洞检测技术又可以分为静态分析技术和动态分析技术。静态分析技术在不执行程序的情况下获得程序的编译信息，并根据这些信息进行漏洞检测。DA Wagner等人[137]将静态分析技术用于缓冲区溢出和入侵检测，证实静态分析技术可以作为保证软件安全的有效手段。Kong等人[138]实现了一个基于数据融合的静态分析系统ISA，他们通过将不同的静态分析工具的检测结果进行有效融合，取得了更好的漏洞检测效果。静态分析克服了人工分析效率低下的缺点，但是普遍存在误报率较高的问题，并且可扩展性差，同时忽略了对程序功能的检测。

动态分析方法通过执行程序来得到程序的运行状态和运行数据，并利用这些运行信息对软件漏洞进行检测。动态分析是在程序运行时进行检测，因此准确率高。而且动态分析还具有针对性强、不需对源代码进行挖掘等优点。但是动态分析存在分析结果不完整的问题，一次或多次的程序执行情况并不能代表程序以后所有可能的运行情况。同时，动态分析对于检测人员的技术和经验也有着较高的要求。许多研究采用了模糊测试、渗透测试等方法检测软件漏洞。Wu等人[139]提出了一种多维模糊测试方法，并且避免了组合爆炸问题，该方法能够挖掘到单维模糊测试技术无法挖掘到的软件漏洞。Holler等人[140]提出了一种模糊测试方法LangFuzz，并应用到Mozilla Javascript解释器和PHP解释器的安全漏洞检测中，取得了很好的效果。Tian等人[141]提出了一种基于模型的渗透测试方法来检测SQL注入漏洞，该方法产生的测试用例能够覆盖更多的SQL注入攻击类型。

（2）基于统计模型的方法

近年来，机器学习和深度学习在语音识别、图像处理、自然语言处理等领域已经取得了重大成功。随着开源代码在数量和种类上的不断丰富，直接从大量的代码中学习软件漏洞模式也成为了可能。已经有许多研究将机器学习和深度学习技术应用到漏洞检测中，取得了很好的效果。Yamaguchi等人[142]将代码表示成向量，并利用主成分分析技术提取API使用模式，从而能够识别具有相似特征的代码漏洞。Pang等人[143]结合n元语法模型和基于机器学习的特征选择算法进行软件漏洞预测，并将该方法用于Java Android 应用的漏洞检测。除了传统的机器学习技术外，Li等人[144]首次将深度学习用于代码漏洞检测，他们根据代码中的库函数和API调用来提取程序切片并表示成向量，然后用Bi-LSTM训练漏洞预测模型，并用来检测两种类型的漏洞。随后，Li等人[145]又进一步提出了漏洞检测框架SySeVR，该框架利用代码的语法和语义信息生成代码的特征向量，并利用Bi-GRU训练模型，该框架能够检测等多类型的漏洞并取得更好的检测效果。

不过，目前基于统计模型的代码检测技术仍然存在一些问题有待进一步解决。比如如何将代码转换成能够体现代码的语义信息和结构信息的向量表示就是目前亟待解决的重要问题。

**2.安全缺陷报告识别（Security bug report identification）**

安全缺陷报告识别通常采用自然语言处理和机器学习技术，通过挖掘历史缺陷报告中自然语言描述的文本信息和缺陷分类信息，将安全缺陷报告识别转化为一个分类、预测或者推荐问题，其中文本挖掘的质量对模型性能的影响很大，因此有效地挖掘缺陷报告中的文本信息是提高模型性能的关键问题。

Gegick[8]根据缺陷报告中自然语言描述的文本信息，利用词袋模型根据从缺陷报告中抽取的关键词将所有缺陷报告表示成一个词项-文档矩阵，并使用该矩阵和相应的缺陷类别标签作为输入来训练统计模型，实现安全缺陷报告的识别。考虑到不同的关键词在识别安全缺陷报告时具有不同的权重，Behl[9]在利用词袋模型表示缺陷报告的基础上，引入了TF-IDF值作为词的权重并通过朴素贝叶斯模型来实现安全缺陷报告识别。Chawla[10]进一步利用了缺陷报告所包含的语义信息，通过TF-IDF、LSI模型，以及多项式朴素贝叶斯模型对缺陷报告进行分类。而Zou[11]不仅使用缺陷报告的文本信息，还充分利用了缺陷报告中的非文本信息来训练模型，以提高安全缺陷报告识别的准确率。

以上基于文本挖掘的安全缺陷报告识别模型主要面临以下几个难点问题：

（1）数据不平衡问题

由于在安全缺陷报告识别的任务中，训练数据存在严重的类不平衡问题，即安全缺陷报告的数量远远小于非安全缺陷报告的数量，这使得预测模型往往具有较高的准确率，但召回率较低，召回率低意味着漏检率高，即未被模型正确识别的安全缺陷报告数量较多，这给软件带来的安全隐患是无法容忍的，因此一些学者在针对安全缺陷报告识别的类不平衡问题进行了研究。Yang[12]利用词袋模型将缺陷报告表示成向量，并分别利用四种不平衡数据的处理方法（随机欠采样、随机过采样、合成少数类样本的过采样、代价矩阵调节方法）对训练集进行预处理，再利用预处理后的训练集训练分类器来降低类不平衡对安全缺陷报告分类造成的影响。Zhou[13]提出了K折叠加算法来集成多个个体分类器，利用集成学习缓解类不平衡的影响。Postojanova[14]提取了三种缺陷报告的特征向量：二元词袋频率（BF）、词频（TF）和TF-IDF，并分别利用监督学习和基于异常监测的非监督学习算法来识别安全缺陷报告，利用非监督学习克服监督学习需要大量人工标注数据以及准确标注为安全缺陷的数据较少带来的类不平衡问题。

（2）数据噪音问题

除了类不平衡问题外，数据噪音也是导致模型召回率较低的原因之一。Peters[15]的研究发现一些非安全缺陷报告同样包含和安全相关的关键词，同时出现在安全和非安全缺陷报告中的安全相关关键词被称为安全交叉词，包含安全交叉词的非安全缺陷报告作为训练数据集相当于在模型的训练中引入了噪音，从而增大了模型将安全缺陷报告误标记为非安全缺陷报告的概率，而这种不利的影响在类不平衡的情况下又被进一步扩大。因此他们提出了一个称为FARSEC的框架，其主要思想是在模型训练前将含有安全交叉词的非安全缺陷报告从训练集中移除，从而降低数据噪音对安全缺陷报告识别的影响。该方法虽然通过过滤部分含有安全交叉词的非安全缺陷报告缓解了类不平衡的影响，但类不平衡问题依然严重，而且由于该方法仅根据TF-IDF值最高的前100个词作为安全相关的关键词，未考虑缺陷报告的内容以及项目包含的实际安全相关关键词的规模，召回率的提高依然是有限的，并且还增大了误检，为了降低误检的影响，该方法将安全报告的识别转化为推荐问题，通过生成一个安全缺陷报告的排序列表，来缓解安全工程师确认安全缺陷的工作量。

（3）数据稀疏问题

数据稀疏问题主要是指缺少项目的历史数据或者项目历史数据中包含的安全缺陷报告极少，解决这一问题的主要方法是采用迁移学习的方法，利用项目外的其他项目作为训练数据集训练模型，从而实现跨项目的安全缺陷报告识别。

### 1.3智能软件系统的特点及测试存在的问题

#### 1.3.1智能软件系统的特点

与传统软件相比，智能软件呈现出**学习深层知识、融合跨界数据、数据处理层级化、决策逻辑不受控、系统输出难以验证、集成群体智慧**的特点，具体来说：

**（1）在学习深层知识方面：**智能软件能够从大数据表示的知识中进行深层次的认知、学习、推理，并通过动态调整自身业务逻辑实现持续演化，适应外部动态变化的需求，而传统软件依据既定的业务逻辑执行各种操作，难以进行自主演化并适应外部环境变化。

**（2）在融合跨界数据方面：**智能软件将跨领域的多源异构数据进行协同处理，实现跨界信息的关联融合，而传统软件系统仅仅分类型处理这些多源异构数据，难以实现信息的深度融合；多领域的数据进行组合导致智能软件的输入空间异常庞大，此外，大量异构的数据造成智能软件数据预处理模块异常复杂。

**（3）在数据处理方面：**智能软件首先采集数据，并对采集的数据进行预处理，得到符合决策模块输入类型的规则数据，然后将规则数据输入决策模块，并得到智能软件的输出。智能软件多层次的数据处理过程导致数据出错的可能性增加。

**（4）在获得决策逻辑方面：**智能软件的决策模块通常由程序开发人员利用高级语言（Java、Python等）实现，但是决策的逻辑从数据中习得，而不是开发人员指定。这种“在代码上堆代码”的方式让系统开发人员以及测试人员难以理解。

**（5）在验证测试结果方面：**智能软件多源化的输入、难以理解的内部逻辑以及所处场景的多样性使得测试人员判断系统行为耗时、耗力。

**（6）在集成群体智慧方面：**智能软件可依托互联网或大数据无缝整合多种智能，形成群体智慧，而传统软件系统更加聚焦于个体的智能。

#### 1.3.2智能软件系统测试存在的问题

由于上述新特点，**智能软件的测试面临诸多新的问题与挑战，具体来说：**

**（1）智能软件的测试任务难以准确描述**：传统软件系统的运行时功能依据明确定义的规格说明实现，因此，软件测试人员可以根据规格说明清晰且准确地了解与描述系统的待测功能，并以此制定测试计划与任务。然而，智能软件具有认知、学习、推理的能力，其功能随着应用场景与任务的变化、认知范围的增加和学习内容的改变而发生变化与衍进，难以预测且呈现出智能性。这些问题导致软件测试人员难以清晰而准确地了解与描述其待测功能。在此情况下，软件测试人员难以对智能软件制定测试任务，从而难以实施有组织有目标的软件测试。

**（2）智能软件的模拟测试平台难以搭建：**智能软件通常与外界环境存在密切的交互，所以智能软件的测试无可避免地涉及到与环境的交互。然而，在真实环境中测试智能软件将产生高昂且难以负担的成本。一个较好的解决方案是搭建智能软件的模拟环境测试平台。如何让模拟测试平台接近智能软件的真实应用场景是一个重要的问题。通常，模拟测试平台需要模拟三大类事物，包括人、人造事物、自然事物。 模拟人与自然事物存在诸多困难，其原因是人类在不同场景下的行为是不确定的，很多自然事物的规律及内在机制至今仍未研究透彻。因此，搭建智能软件的模拟测试平台存在诸多困难。

**（3）测试用例的可靠性与科学性难以验证：**智能软件的测试数据可能不准确并且基于学习的智能软件的学习模型经常出现过拟合的现象，出现不正确或者不被期待的行为。其次，智能软件系统的模块交互复杂，输入空间巨大，也有可能存在一些约束关系，测试用例的能否正常执行尚且认识不足，所以按照软件测试生成算法生成的测试用例很有可能难以执行。因此，保证测试用例的科学性需要考虑。

**（4）测试判定的困难：**目前主要的判定有两种：（a）将系统的行为与执行数据对应的标签对比；（b）测试人员判断。这两种方式都需要人工参与，然而在长时间的工作下，资深测试工程师的准确性也不能得到保证。对于智能软件而言，系统的可靠性往往较高，所以通常不会出现软件崩溃等执行结果，更多的可能是与预期结果相比微小的执行差异，包括色彩、音频、数据等，如果单纯依靠测试人员进行主观判断，那么工作量巨大的同时，精确度也很难保证，所以对于智能软件系统的测试执行判定，我们需要开发出更加智能化的判定方式。

**（5）测试充分性无法保证：**智能软件的输入空间十分巨大，例如无人驾驶汽车通过各种传感器获取无数种可能的外界信息。目前的测试技术主要通过三种方式获得测试数据：**（a）随机生成测试用例：**随机地在测试用例集中挑选测试用例或者随机地生成测试数据具有简单、易用的特点。该方法被谷歌等世界著名公司使用来生成无人驾驶车的测试数据[62]。然而，随机的方式没有利用任何系统内部信息以及测试过程信息，使得很多智能系统不正确的行为不能被发现[63]；**（b）人工生成测试用例：**该方式通过在普通测试数据的基础上做微小改动来得到新的测试数据[64, 65]，然后测试人员判断每一个新测试数据对应的行为。这种方式成功地运用在多种智能软件中[66-69]。随机生成和人工生成测试用例的方式都没有考虑智能软件的内部结构，不能覆盖智能软件的大部分逻辑，导致了这些测试用例只能发现少量的系统异常行为。Pei等人通过经验研究发现：随机生成一个测试用例几乎能够覆盖无人驾驶车的所有代码，但是只激活了不到 10%的神经元。基于上述观察，Pei等人提出了神经元覆盖指标，评估测试用例集激活的神经元数目[63]。Tian等人利用神经元覆盖指标，生成测试用例，激活智能系统大多数的神经元，并通过经验研究的方式验证该方法生成的测试用例集可以发现更多智能系统不正确的行为；**（c）基于场景和功能：**基于场景的测试技术通过给定的场景及任务测试智能软件的实际表现。例如，前文提到的 DARPA 无人驾驶挑战赛中，无人驾驶汽车需要在指定时间内安全地穿越莫哈韦沙漠中的一个区域。对于一些较为简单的智能软件的测试，测试人员可以枚举测试场景及任务以验证系统在这些场景下的表现是否符合预期。但是，对于复杂的智能软件，穷举测试场景与任务是十分耗时耗力且效率低下的。并且，在组合场景与任务下出现的组合爆炸问题使这种技术难以应用。此外，这种技术的测试结果仅从宏观层面对智能软件进行定性评价[70]，难以从微观层面对智能软件的功能质量进行定量评价。基于功能的测试技术将智能软件按功能划分子模块，并针对每个子模块生成测试用例并针对功能进行测试。例如，一个无人驾驶系统可以被划分为感知及识别模块、决策模块及动作执行模块[71, 72]。该技术为这些模块分别生成测试用例，并对模块的功能正确性进行检验。

**（6）智能软件的质量评价指标及模型存在缺失：**对于传统的软件系统，测试人员可根据McCall质量模型、ISO/IEC 25010质量模型等成熟的软件质量模型为待测系统选定质量特性，并评估系统在测试中（的表现是否）满足给定需求或特性的程度。智能软件与传统软件系统最大的不同是前者所特有的人工智能，因此，对智能软件的质量进行评价不可不免地要涉及到对系统智能程度的评估。现有的智能度评估方法主要有三类[73]，包括人工辨别法、基准任务法、伙伴对抗法，然而这三类方法分别存在实验设置困难、评估结果片面、结果依赖于对矿伙伴的问题，且未有较为系统的评价指标及模型。有研究者提出通过评价智能系统与人类在行为上的相似性来评估智能系统的质量[74, 75]，但仍然没有合适的度量标准。此外，很多智能软件是多目标的，例如，无人驾驶系统会考虑驾乘舒适性、燃油消耗等。智能软件在多目标情景下会依据目标优先级的不同而展现出不同的行为[76]。在缺乏针对性质量评价指标及模型的情况下，难以对智能软件的质量进行系统有效的评估。

**（7）缺少对安全缺陷识别的支持：**防危性（Safety）和安全性（Security）是智能软件系统的两个重要属性。尽管智能软件系统工程师可能为系统写下了很多安全规则，但也很难穷尽所有的情况。国家互联网应急中心发布了《开源软件代码安全缺陷分析报告——人工智能类开源软件专题》，对国际、国内知名的开源软件项目进行了评测，结果表明，这些人工智能类开源软件都存在不同程度的安全问题。Security漏洞无法被及时地修复，容易被恶意攻击者所利用，破坏系统；safety漏洞无法被及时识别和修复，则会有可能引发严重的安全事故。特别是随着智能软件系统越来越多地应用于医疗、交通、安全和国防等安全关键的领域中，如何尽早发现安全相关的软件缺陷，消除近期和长期可能的安全隐患，保证系统安全可靠地工作，成为智能软件系统评测中的一项关键任务。然而由于智能软件系统的特殊性，用于测试、验证软件的传统方法根本不适用于智能软件系统中，需要开发安全漏洞检测与识别的新方法。

### 1.4结论

智能化时代已经来临，智能软件系统正全方位、多领域的重新定义和实现整个世界，世界各国正在紧密部署智能化发展战略，我国也出台了一系列的方针政策旨在抓住智能化发展机遇，促进企业转型升级。然而智能化发展还远未成熟，智能软件中的缺陷与漏洞仍然会引发难以预估的灾难，例如无人驾驶屡出事故，在危及用户生命安全的同时也会引发普通民众对新兴领域信任危机，阻碍新兴行业的进一步发展。可见，智能软件的飞速发展离不开智能软件测试技术的逐步完善，软件测试是一种广泛采用的软件质量保证手段**，**而智能软件测试方面的研究工作仍存在诸多不足，智能软件与传统软件相比也存在一些特别之处，所以传统的软件测试技术和方法并不能完全照搬，整个智能化领域都亟待探索有效且高效的新型智能软件的测试方法与技术。**本课题旨在针对智能软件测试方面的问题与挑战，采用数据驱动式测试方法，探索智能软件的测试关键技术，以系统有效的方式检测与评估智能软件的质量，为开发可靠的智能软件提供新型测试理论与工具支持。**

**参考文献（需要重新筛选与标注）**

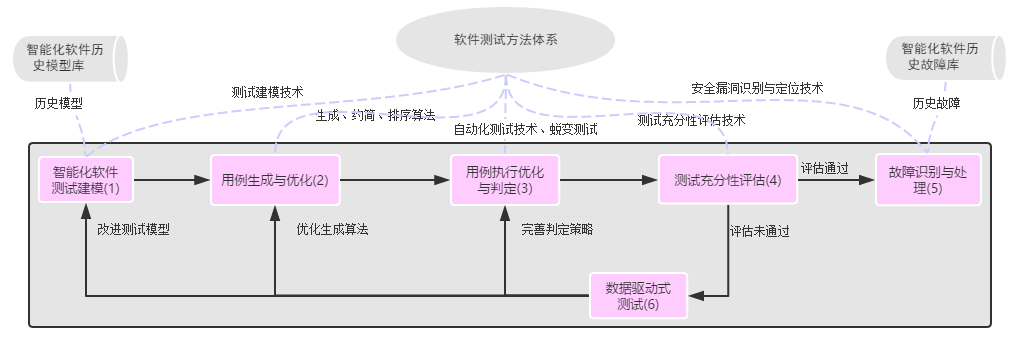
1. Pei, Kexin, et al. "Deepxplore: Automated whitebox testing of deep learning systems." Proceedings of the 26th Symposium on Operating Systems Principles. ACM, 2017.
2. Yosinski, Jason, et al. "Understanding neural networks through deep visualization." arXiv preprint arXiv:1506.06579 (2015).
3. Guo, Jianmin, et al. "DLFuzz: differential fuzzing testing of deep learning systems." Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ACM, 2018.
4. Tian, Yuchi, et al. "Deeptest: Automated testing of deep-neural-network-driven autonomous cars." Proceedings of the 40th International Conference on Software Engineering. ACM, 2018.
5. Y. Sun, X. Huang, and D. Kroening, “Testing Deep Neural Networks,” ArXiv e-prints, Mar. 2018.
6. Ma, Lei, et al. "Deepgauge: Multi-granularity testing criteria for deep learning systems." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM, 2018.
7. Sun, Youcheng, et al. "Concolic Testing for Deep Neural Networks." arXiv preprint arXiv:1805.00089 (2018).
8. Ma, Lei, et al. "Combinatorial Testing for Deep Learning Systems." arXiv preprint arXiv:1806.07723 (2018).
9. Ma, Lei, et al. "DeepMutation: Mutation Testing of Deep Learning Systems." arXiv preprint arXiv:1805.05206 (2018).
10. Zhang, Mengshi, et al. "DeepRoad: GAN-based Metamorphic Autonomous Driving System Testing." arXiv preprint arXiv:1802.02295 (2018).
11. Abdessalem, Raja Ben, et al. "Testing autonomous cars for feature interaction failures using many-objective search." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM, 2018.
12. Chen, Boyuan, et al. "An automated approach to estimating code coverage measures via execution logs." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM, 2018.
13. Han, Xue, Tingting Yu, and David Lo. "PerfLearner: learning from bug reports to understand and generate performance test frames." Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. ACM, 2018.
14. V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, J. K. Hedrick. Learning aDeep Neural Net Policy for End-to-End Control of Autonomous Vehicles. *Proceedings of the 2017 American Control Conference* (ACC’17), IEEE Computer Society, 2017, pp. 4914-4919.
15. K. He, X. Zhang, S. Ren, J. Sun. Deep Residual Learning for Image Recognition.

*Proceedings of the IEEE conference on computer vision and pattern recognition* (CVPR’16), IEEE Computer Society, 2016, pp. 770-778.

1. M. A. O. Vasilescu, D. Terzopoulos. Multilinear Image Analysis for Facial Recognition. *Proceedings of the 16th In International Conference on Pattern Recognition* (ICPR’02), IEEE Computer Society, 2002, pp. 511-514.
2. W. Xiong, J. Droppo, X. Huang, F. Seide, M. L. Seltzer, A. Stolcke, G. Zweig. Toward Human Parity in Conversational Speech Recognition. *Proceedings of the IEEE/ACM Transactions on Audio, Speech and Language Processing* (TASLP’17), IEEE Computer Society, 2017, pp. 2410-2423.
3. Silver D, Huang A, Maddison C J, et al. Mastering the game of Go with deep neural networks and tree search[J]. Nature, 2016, 529(7587): 484-489.
4. Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. Nature, 2017, 550(7676): 354-359.
5. <https://www.iyiou.com/p/73145.html>.
6. <http://m.elecfans.com/article/674086.html>.
7. <https://deepmind.com/blog/alphafold/>.
8. <https://search.proquest.com/docview/2015377429?pq-origsite=gscholar>.
9. <http://www.theverge.com/2016/2/29/11134344/google-selfdriving-car-crash-reprt>A Google Self-Driving Car Caused a Crash for The Frst Time.
10. [https://electrek.co/2016/07/01/understandingfatal-tesla-accident-autopilot-nhtsa-p robe/](https://electrek.co/2016/07/01/understandingfatal-tesla-accident-autopilot-nhtsa-p%20robe/), Understanding the Fatal Tesla Accident on Autopilot and the NHTSA Probe.
11. <https://en.wikipedia.org/wiki/Tay_(bot)>.
12. G. J. Myers, C. Sandler, T. Badgett. The Art of Software Testing. John Wiley and Sons, 2011.
13. Anand S, Burke E K, Chen T Y, et al. An orchestrated survey of methodologies for automated software test case generation[J]. *Journal of Systems and Software*, 2013, 86(8): 1978-2001.
14. Baldoni R, Coppa E, D’elia D C, et al. A survey of symbolic execution techniques[J]. *ACM Computing Surveys* (CSUR), 2018, 51(3): 50:1-50:39.
15. King J C. A new approach to program testing[C]. In *Proceedings of the International Conference on Reliable Software*. ACM, 1975, 228-233.
16. De Moura L, Bjørner N. Z3: An efficient SMT solver[C]. In *Proceedings of the 14th International conference on Tools and Algorithms for the Construction and Analysis of Systems* (TACAS’08). Springer, 2008: 337-340.
17. Ganesh V, Dill D L. A decision procedure for bit-vectors and arrays[C]// In *Proceedings of the 19th International Conference on Computer Aided Verification* (CAV’07). Springer, 2007: 519-531.
18. Choco Solver [EB/OL]. [2018-11-18]. <http://www.choco-solver.org/>.
19. Chipounov V, Georgescu V, Zamfir C, et al. Selective symbolic execution[C]// In *Proceedings of the 5th Workshop on Hot Topics in System Dependability* (HotDep). 2009.
20. Odefroid P, Klarlund N, Sen K. DART: directed automated random testing[C]. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation* (PLDI’15). ACM, 2005, 213-223.
21. Cadar C, Dunbar D, Engler D R. KLEE: Unassisted and Automatic Generation of High-Coverage Tests for Complex Systems Programs[C]// In *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation* (OSDI’08). 2008, 209-224.
22. Godefroid P, Levin M Y, Molnar D A. Automated whitebox fuzz testing[C]// In *Proceedings of the 15th Annual Network and Distributed System Security Symposium* (NDSS’08). 2008, 151-166.
23. Dias Neto A C, Subramanyan R, Vieira M, et al. A survey on model-based testing approaches: a systematic review[C]. In *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies*, in conjunction with the *22nd IEEE/ACM International Conference on Automated Software Engineering* (ASE’07). ACM, 2007: 31-36.
24. Lee D, Yannakakis M. Principles and methods of testing finite state machines-a survey[C]. In *Proceedings of the IEEE*. IEEE Computer Society, 1996, 84(8): 1090-1123.
25. Kansomkeat S, Rivepiboon W. Automated-generating test case using UML statechart diagrams[C]// In *Proceedings of the 2003 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology* (SAICSIT’03). South African Institute for Computer Scientists and Information Technologists, 2003: 296-300.
26. Cohen D M, Dalal S R, Fredman M L, et al. The AETG system: An approach to testing based on combinatorial design[J]. *IEEE Transactions on Software Engineering*, 1997, 23(7): 437-444.
27. Lei Y, Kacker R, Kuhn D R, et al. IPOG: A general strategy for t-way software testing[C]//In *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems* (ECBS’07). IEEE Computer Society, 2007: 549-556.
28. Lei Y, Kacker R, Kuhn D R, et al. IPOG/IPOG-D: efficient test generation for multi‐way combinatorial testing[J]. *Software Testing, Verification and Reliability*, 2008, 18(3): 125-148.
29. Ghazi S A, Ahmed M A. Pair-wise test coverage using genetic algorithms[C]. In *Proceedings of the Congress on Evolutionary Computation* (CEC’03). IEEE Computer Society, 2003, 1420-1424.
30. Bryce R C, Colbourn C J. One-test-at-a-time heuristic search for interaction test suites[C]. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (GECCO’07). ACM, 2007: 1082-1089.
31. Williams A W. Determination of test configurations for pair-wise interaction coverage[M]. Testing of Communicating Systems. Springer, Boston, MA, 2000: 59-74.
32. Kobayashi N, Tsuchiya T, Kikuno T. A new method for constructing pair-wise covering designs for software testing[J]. *Information Processing Letters*, 2002, 81(2): 85-91.
33. Calvagna A, Fornaia A, Tramontana E. Random versus combinatorial effectiveness in software conformance testing: A case study[C]. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing* (SAC ’15). ACM, 2015: 1797-1802.
34. Vilkomir S, Starov O, Bhambroo R. Evaluation of t-wise approach for testing logical expressions in software[C]. In *Proceedings of the 6th International Conference on Software Testing, Verification and Validation Workshops* (ICSTW’13). IEEE, 2013: 249-256.
35. Medeiros F, Kästner C, Ribeiro M, et al. A comparison of 10 sampling algorithms for configurable systems[C]. In *Proceedings of the 38th International Conference on Software Engineering* (ICSE’16). ACM, 2016: 643-654.
36. Chen T Y, Leung H, Mak I K. Adaptive random testing[C]. In *Proceedings of the 9th Annual Asian Computing Science Conference* (ASIAN’04). Springer, 2004: 320-329.
37. Chen T Y, Kuo F C, Merkel R G, et al. Mirror adaptive random testing[J]. *Information and Software Technology*, 2004, 46(15): 1001-1010.
38. Chen T Y, Merkel R, Wong P K, et al. Adaptive random testing through dynamic partitioning[C]. In *Proceedings of the 4th International Conference on Quality Software* (QSIC’04). IEEE, 2004: 79-86.
39. Barr E T, Harman M, McMinn P, et al. The oracle problem in software testing: A survey[J]. *IEEE Transactions on Software Engineering*, 2015, 41(5): 507-525.
40. Li L, Lin Y L, Zheng N N, et al. Artificial intelligence test: a case study of intelligent vehicles[J]. *Artificial Intelligence Review*, 2018: 1-25.
41. Campbell M, Egerstedt M, How J P, et al. Autonomous driving in urban environments: approaches, lessons and challenges[J]. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 2010, 368(1928): 4649-4672.
42. Li L, Huang W L, Liu Y, et al. Intelligence testing for autonomous vehicles: a new approach[J]. *IEEE Transactions on Intelligent Vehicles*, 2016, 1(2): 158-166.
43. Huang W L, Wen D, Geng J, et al. Task-specific performance evaluation of UGVs: case studies at the IVFC[J]. *IEEE transactions on Intelligent Transportation Systems*, 2014, 15(5): 1969-1979.
44. Li L, Wen D, Zheng N N, et al. Cognitive cars: A new frontier for ADAS research[J]. *IEEE Transactions on Intelligent Transportation Systems*, 2012, 13(1): 395-407.
45. Pei K, Cao Y, Yang J, et al. DeepXplore: Automated whitebox testing of deep learning systems[C]//In *Proceedings of the 26th Symposium on Operating Systems Principles* (SOSP’17). ACM, 2017: 1-18.

## **2．项目的研究内容、研究目标，以及拟解决的关键科学问题（此部分为重点阐述内容）；**

在智能软件测试与软件测试方法理论研究成果基础上，我们拟围绕智能软件测试流程中关键步骤进行研究，探索智能软件的测试框架与优化技术。测试过程主要包括**智能软件测试建模(1)**、**测试用例生成与优化(2)**、**测试用例执行判定(3)**、**测试充分性评估(4)、故障识别与处理(5)及数据驱动式测试(6)**（图2-1所示）**。**



**图2-1 智能软件的数据驱动式测试**

为了尽可能提升测试的科学性，我们拟采用**测试产生数据，数据优化测试**的数据驱动式测试理念。测试建模过程充分借鉴智能软件历史模型库及软件测试方法体系中的建模技术，测试用例生成及执行判定旨在结合传统的软件测试方法，如组合测试、蜕变测试等，测试评估旨在从测试充分性上考虑测试是否完全，数据驱动式测试技术旨在通过软件运行数据及测试数据的研究学习，利用数据挖掘技术，对测试建模、测试生成、测试执行等步骤进行改进，提升测试效果，故障识别与处理旨在基于智能软件历史故障库，根据测试报告识别与定位出智能软件潜在的安全漏洞，具体的研究内容如下：

### 2.1 研究内容

#### 2.1.1测试模型的建立

建立测试模型是软件测试的第一步，测试模型的好坏决定了整个测试周期的优劣，模型是否真实的还原待测软件，是否易于搭建可执行的模拟平台，是否便于生成高质量的测试用例等问题都是我们在测试建模阶段需要考虑的。智能软件往往功能庞大，模块众多且交互复杂，模块调用依赖数据传递，这些给智能软件的充分性测试带来诸多难题，包括测试平台不易搭建，测试任务难以描述，测试模块过于抽象难以执行等问题。我们拟研究基于场景的智能软件模型建立方法，在充分研究智能软件历史模型库的基础上，结合传统软件的测试建模技术与方法，明确细致的描述测试任务，针对不同类型的智能软件，分析软件系统在主要应用场景下的真实历史运行数据，细化应用场景下的事物、交互对象、交互方式，并以此为基础**构建智能软件在各应用场景下的测试模型**。具体研究内容如下：

**（1）传统软件建模技术研究：**智能软件与传统软件存在诸多差异，但传统软件的建模技术与经验同样值得借鉴，我们拟研究软件测试方法体系中的具体测试建模技术，结合智能软件特征和模型特点建立合适的智能软件测试模型，例如利用等价类划分方法来整合智能软件不同模块之间的关系，将组合测试建模技术应用于对某一功能组合爆炸问题的充分性测试的建模中。

**（2）智能软件历史模型库研究：**智能软件广泛应用于家居、出行、医疗、教育、科技、娱乐等诸多方面，虽然软件表现形式多样，但内在模型有一定的共通之处。我们拟对智能软件模型进行具体分类，总结不同类别对应的显著特征，丰富智能软件历史模型库，归纳对于某一类别的智能软件的建模思路与建模案例分别有哪些。

**（3）智能软件待测任务描述：**智能软件有认知、学习、推理的能力，其功能随着应用场景与任务的变化、认知范围的增加和学习内容的改变而发生变化与衍进，难以预测且呈现出智能性。充分性测试需要深入浅出，由表及里，既要对智能软件的诸多功能进行完备测试，也要对功能与功能的交互进行充分测试，所以我们拟研究如何对智能软件待测功能进行细分，对待测任务进行明确描述。

**（4）智能软件模拟平台搭建：**针对真实环境下的智能软件测试存在高昂且难以负担的成本的问题，我们拟研究如何科学搭建智能软件的模拟测试平台，即以智能软件的应用场景模型为指导，分析软件系统在主要应用场景下的真实历史运行数据，细化应用场景下的事物、交互对象、交互方式，并以此为基础构建智能软件在各应用场景下的模型，模拟系统实际应用环境下的各种事物及对象，较为真实地构建智能软件的实际应用环境，为智能软件的测试奠定基础。

#### 2.1.2测试用例的生成与优化

智能软件测试的两大关键任务包括测试用例生成与测试执行判定，测试用例生成是软件测试领域至关重要的研究问题，理想的测试用例集是以少量的用例数量覆盖尽可能大的测试场景，其次测试用例的生成也需要充分考虑用例的执行判定。智能软件要求具有类似人类的智能行为，且大多数智能软件需要和外界进行频繁的交互，导致智能软件的输入域异常庞大、系统复杂性较高。显然，穷尽测试无法实现。生成有效的、多方位测试智能软件的测试用例是一项具有挑战性的工作。现有的智能软件测试技术利用随机或人工的方式生成测试用例并且主要依靠测试人员验证系统行为，导致测试效率可能不高。另一方面，现有的智能软件测试工作忽略了测试过程对测试效率的影响。在测试资源有限的情况下，利用数据驱动的方式（基于测试的历史信息），控制测试过程，提高测试效率。

针对现有智能软件测试用例生成技术的诸多局限性，考虑智能软件的特点，依据现有测试用例生成技术与相关理论，研究新型数据驱动的智能软件的测试用例生成技术，包括：测试用例生成与选择，测试用例执行与测试用例优先级排序等，具体研究以下内容：

**（1）在测试用例生成与选择方面：**智能软件输入参数的多源性导致输入空间巨大。增加测试用例集的多样性有助于更有效地检测智能软件的异常行为。为了实现测试用例集多样性，探索满足某种覆盖准则的测试用例生成方法；结合场景与功能的测试用例生成技术，将场景测试数据与功能测试数据相结合来实现智能软件自顶向下的测试；基于参数组合优化的测试生成技术，实现减少测试用例数目的同时覆盖尽可能多的组合。此外，我们拟研究如何将用例生成算法与后续的结果判定结合起来，尽可能生成一系列易于判定或者为可判定的测试用例。

**（2）在测试用例优化排序方面：**测试用例集很多情况下存在着较大的优化空间，我们可以通过真实场景分析及执行结果预测等方式删减一批冗余或者无效测试用例，在不影响测试效果的情况下，简化测试用例集，降低测试成本。另一方面，通过改变测试用例的执行顺序以提早执行到揭示故障的测试用例。基于此，即将执行的测试用例应尽可能与已经执行的测试用例不同。通过测试用例距离度量的方式选择差异最大的测试用例，并以此为基础，研究基于距离的智能软件测试用例优先级排序技术等。

#### 2.1.3测试用例的执行判定

大部分情况下，智能软件的测试执行结果主要通过测试人员主观判断系统行为是否正确，这样的人工判定方式不仅需要大量的测试资源（大量的专业测试人员），还易出现误判的情况（将错误的行为判断为正确的行为）。在传统软件测试过程中，测试人员主观判断的方式尽管也存在一定误判情况，但是总的来说是非常有效的测试结果判定方式，但是对于智能软件而言，测试执行之后，系统输出可能非常复杂，或者输出加密数据，或者数据随机性较大，或者根本没有显著结果，这都给测试人员主观判断带来极大的阻碍。因此我们旨在研究测试用例的自动化执行判定的相关内容，利用辅助算法或程序，对待测用例的执行结果进行尽可能准确的判定，我们拟研究自动化验证系统行为的方法，进而提高测试效率与准确度，具体研究内容如下所示：

**（1）研究自动或者直观的执行判定方式：** 人工验证系统行为的方式不仅需要占用大量的测试资源而且不能保证判断的准确度。因此，提出一种自动地验证系统行为的方式是有必要的。传统软件测试中缓解测试预期问题的技术有很多（例如：N版本、断言等），其中一些技术运用到智能软件测试效果可能不好。例如，断言技术需要在代码中插入一些可以获取执行信息的代码，但很多智能软件的执行逻辑不是由程序控制的。考虑将N版本以及交叉验证的方式作为验证系统行为的机制。

**（2）研究基于蜕变测试的智能软件的测试结果验证技术：**蜕变测试是一种无需测试预期的软件测试技术，在待测软件的测试预期不存在的情况下也能对其进行有效测试。该测试技术通过判断待测软件的多个测试用例之间是否满足一些必要的属性来测试程序。研究基于蜕变测试的智能软件测试结果验证技术，需要研究如下内容：（a）面向智能软件的蜕变关系识别方法：智能软件中隐含的蜕变关系是验证测试结果的关键。如何从智能软件中识别蜕变关系是一个重要的问题。研究基于范畴划分方法的和基于数据变异的智能软件蜕变关系识别方法；（b）基于蜕变测试的智能软件测试结果验证实现机理： 通过判断蜕变关系所涉及测试用例的输出结果之间是否满足蜕变关系，实现智能软件测试结果的验证，判断待测软件是否满足功能需求。

#### 2.1.4测试充分性评估

测试用例执行之后，我们拟对测试结果进行有效评估，评估测试用例集是否达到充分完全的覆盖，评估是否存在遗漏的测试场景，评估是否需要针对某些场景生成额外的测试用例，评估未通过的测试用例集中是否包含隐藏故障，评估待测软件在容错性质量上是否达标等。传统软件测试中的测试充分性指标（语句覆盖、分支覆盖和MC/DC覆盖等）不能完全适用于智能软件测试。原因是在智能软件测试中，不仅要考虑数据处理模块、决策模块等对应的代码是否正确实现，还需要考虑影响决策逻辑的内部因素（神经元、激活函数和阈值等）。另一方面，由于智能软件获取逻辑的方式，传统软件测试中评估测试用例集充分性的技术（变异分析和各种覆盖准则）也不能完全适用于智能软件。我们拟结合传统软件测试方法体系中的评估指标，例如语句覆盖率，条件覆盖率等，确定合适的面向智能软件的充分性测试评估指标，该评估指标可以针对某一维度，给出待测软件的测试用例集的覆盖情况和遗漏的测试场景等，最终从多个维度给出待测软件的可信度或可靠度分析。

测试评估是对整个测试流程的评估，评估的结果并不取决于发现问题的多少，而是在于某一维度下，测试用例集是否达到了充分覆盖，是否存在重要场景遗漏的情况。若评估未通过，则进入数据驱动式测试阶段（图2-1所示），重新优化测试模型，生成算法及判定策略等，开启新一轮测试，直到测试评估结果达到预期。

#### 2.1.5故障识别与处理

为了保障智能化软件等安全关键系统的可靠性和安全性，我们不仅要通过软件测试发现软件系统内部的故障，还要能够辅助系统开发和维护人员快速识别出那些隐藏在软件中的不易被软件测试识别和发现的安全漏洞，尽管安全性测试方法和工具可以帮助测试人员识别软件中潜在的安全性缺陷，但是安全性测试并不能最终证明软件是安全可靠的，不可避免地存在一些漏报或误报，软件安全工程师在对测试报告的缺陷进行确认时，如果其对安全相关的专业领域知识缺乏，则安全相关的缺陷有可能会被误报为非安全相关的缺陷，从而给软件带来严重的安全隐患。为了降低漏报和误报对软件安全可靠性造成的影响，控制软件开发过程中可能出现的软件安全漏洞，减少智能软件系统可能面临的安全风险，我们拟从与软件测试相互补充的角度研究安全漏洞的检测与识别方法，帮助软件安全工程师准确识别软件中存在的与安全相关的缺陷，避免安全漏洞的漏报和误报给智能软件系统安全可靠工作带来的严重影响，从而保障软件的质量。

为了与软件测试手段相互补充，确保**软件的自身安全（Safety**）和**软件的信息安全（Security），**我们拟从程序代码和缺陷报告两个角度，研究安全漏洞的识别技术。一是基于程序源码研究软件自身安全漏洞的检测方法，二是基于缺陷追踪系统中的缺陷报告研究安全缺陷报告的识别与定位方法。具体研究内容如下：

**（1）代码安全漏洞识别方法研究**：研究结合深度学习和程序分析技术的代码安全漏洞预测框架。具体包括：安全漏洞特征模式库的创建，代码的结构和语义表示方法研究，适合安全漏洞预测的深度学习框架。

**（2）安全缺陷报告的识别方法研究：**研究结合排序学习和词嵌入模型的安全缺陷报告识别方法。具体包括：训练数据集的去噪方法研究，缺陷报告的分布式向量表示方法研究，基于机器学习的预测模型研究，基于迁移学习的跨项目预测模型研究。

#### 2.1.6数据驱动式测试

随着互联网软件服务的普及，软件数据的种类更加丰富，规模也越来越大。软件生命周期中会产生大量的、各种类型的数据，例如开发过程中的源代码、需求文档、缺陷报告、测试用例；系统运行中的运行日志、性能度量、事件记录；用户交互中使用行为序列、调查问卷、社交媒体反馈，软件版本信息、更新日志等等。其次，在智能软件测试过程中，也会产生很多数据，例如测试日志、测试数据、测试用例集及执行结果、故障报告、安全报告、故障诊断与修复日志等等。我们拟研究数据驱动式测试，对于软件生命周期中涉及的数据（如训练数据集、输入数据集、输出数据集、日志报告等）和测试过程中产生的数据（如测试报告等）进行系统处理，拟研究如何利用数据挖掘算法提取海量数据中潜在的价值，研究如何利用这一类数据信息，丰富数据驱动式测试技术，完善智能软件测试框架，提升智能软件测试能力。具体研究内容如下：

**（1）数据筛选方法：**大数据的五个基本特征为大量、高速、多样、价值、真实，对于多样且庞大的数据集，我们拟研究如何甄别高价值数据，过滤无效数据或噪音数据，拟研究如何科学存储海量数据。

**（2）基于数据挖掘技术的数据处理：**我们拟研究如何利用数据挖掘技术，提取数据集智能蕴含的数据价值，构建合适的数据模型。

**（3）优化测试框架及改进测试技术：**直观而言，软件数据及测试数据可以帮助软件测试人员重现缺陷，识别缺陷以及定位缺陷。甚至可以对软件质量及可靠性做出有效评估。深层次而言，基于这些海量数据，软件从业者可以提取出关于软件质量和开发动态的重要信息。再依靠这些信息，指导我们进行高效测试，制定数据驱动式的测试框架，达到对智能软件的高效测试。具体包括**(a)改进测试模型：**智能软件的测试建模不是一成不变的，第一次建模也不是尽善尽美的，我们需要根据对测试不足的分析，动态修改建模方式，进一步完善测试模型，**(b)优化生成算法：**测试用例的生成基于测试模型和软件测试方法体系，在智能软件测试中，生成测试用例也需要灵活的根据测试场景及对象的差异而动态调整，数据驱动式测试旨在进一步优化生成算法，在效率上增强生成算法的竞争力。**(c)完善判定策略：**结果判定问题是测试执行的关键，而判定策略更多的基于对待测软件的研究不断深入而逐步完善，例如蜕变关系的提取等，但判定策略的完善也不是一蹴而就的，需要根据测试数据不断优化，很多之前难以判定的测试场景可以在之后的测试中加以判断。**(d)提升故障诊断与定位能力：**因为智能软件可靠性与冗余性较高，故智能软件故障隐蔽性较高，而算法的故障诊断能力也随着测试的不断进行，数据的不断挖掘会不断提升，故障定位能力可能也会由可疑模块定位转向精确模块定位，甚至提升到代码定位等。

### 2.2 研究目标

针对上述诸多研究内容，我们拟达到以下研究目标：

#### 2.2.1建立完备的智能软件测试建模理论体系

智能软件数量庞大，种类众多，我们拟通过归纳不同智能软件的特点，对智能软件的架构方式与开发运行方式进行有效总结，对多样的智能软件进行科学分类，对智能软件待测任务进行科学化描述，建立完备的智能软件测试建模理论体系。具体目标如下：

**（1）基于传统软件测试建模技术，总结智能软件建模方案集：**从传统的软件测试方法体系中提取适合本课题的建模理论与技术，提出一系列可能的软件建模方案，在后续的测试开展中加以比较和修改，最后加以综合。

**（2）建立一个完备的科学的智能软件模型库：**该模型库包含典型的智能软件的建模案例，在建模科学性方面有详细论述，可用于对某具体智能软件的建模指导。

**（3）构建科学的智能软件待测任务描述方式：**可根据某一场景或任务下的智能软件典型特征，分析出智能软件为适应或满足这一特征而必须满足的运行时特性，利用这些特性描述智能软件的待测任务。

**（4）搭建一个真实可靠的测试平台：**模拟测试平台是基于真实智能软件搭建且真实可靠，抽象模型可应用在测试平台之上，对于诸多待测功能与模块，在模拟测试平台上都有对应的近似方法以保证真实性、可靠性与可扩展性，为开展模拟测试奠定基础。

#### 2.2.2建立完备的智能软件测试用例生成与优化方法库

我们拟通过分析智能软件测试模型的特征及待测任务的特点，依据现有测试用例生成技术与相关理论，研究新型数据驱动的智能软件的测试用例生成技术及优化技术，具体目标如下：

**（1）完善面向智能软件测试的测试用例生成方法库：**该用例生成方法库可以根据不同的覆盖指标生成相应的可判定的测试用例集，该用例集可以在某一指标下，对智能软件中进行多角度，全方位进行充分性测试，满足场景与功能相结合技术。

**（2）完善面向智能软件测试的测试用例优化方法库：**该优化方法库可以通过真实场景分析及执行结果预测等方式删减一批冗余或者无效测试用例，简化测试用例集，降低测试成本。另外，可以提供一系列排序方案，利用测试用例执行顺序的改变尽早揭示软件系统内部可能存在的故障和缺陷。

#### 2.2.3建立执行结果的智能化判定方法体系

智能化是软件测试的发展趋势，针对智能软件执行结果判定困难的问题，我们拟建立智能化判定方法体系，在原有的判定基础上，逐步丰富和发展其他的智能化判定方法。智能化判定方法体系可以帮助我们根据实际情况，对执行结果采取最合适的判定方法。对于判定不明的情况，我们可能采用机器学习的方式，对执行结果进行预判定或者部分判定，继而使用人力或者其他方式进行逐一分析，可能的判定方法如下：

**（1）自动或者直观判断的结果判定方法：** 自动判断或直观判断并不是一定不适用于智能软件的执行判定的，相反在某些场景或某些条件下，直观判断或自动判断却是最有效的方法。我们通过对一批易于直观判断执行结果的软件或场景进行归纳，归纳出适合于此的场景或条件。

**（2）基于蜕变测试的结果验证方法：** 基于蜕变测试的结果验证技术是智能化执行判定的重要法宝，该方法可以指导测试人员快速识别出隐藏在智能软件内部的蜕变关系，对于满足某一蜕变关系的一批测试用例集的执行结果集进行合理性分析。

**（3）基于特征提取的结果判定方法：**智能软件的输出是多样的，基于特征提取的结果判定方法指导我们对不同输出的显著特征进行提取，利用特征之间的差异来判定执行的正确性。对于特征模糊或特征不明的情况，该判定方法可利用数据挖掘技术，对不同特征与输出进行聚类划分，通过多特征提取方法对执行结果进行预判定或部分判定。

#### 2.2.4建立科学的智能软件测试的评估体系

我们拟提出一系列智能软件评估指标，包括测试充分性与测试集充分性的两种评估。传统软件的评估体系并不完全适用于智能软件的评估。我们拟从科学性角度建立一种科学的面向智能软件测试的评估体系，并给出合理性、科学性与适用性说明，应用该方法体系对我们的测试方法及测试用例集进行高效评估。主要目标如下：

**（1）提出测试充分性的评估指标：**传统软件的测试充分性度量指标可以评估智能软件中代码部分的测试充分性，不能对决策模型进行有效的评估。为此，借鉴传统软件测试充分性评估指标的提出思路，并结合决策模型的特点，提出适用于智能软件的测试充分性指标。

**（2）提出测试用例集充分性的评估指标：**变异分析和覆盖准则可以评估传统软件测试用例集的充分性，但由于缺少针对智能软件的变异算子，变异分析技术不能直接运用到智能软件测试中。为了解决这一问题，分析以往的测试历史，提取故障模式，总结面向智能软件的变异算子，改进变异分析与覆盖准则以评估测试用例集的充分性。

#### 2.2.5建立基于程序代码和缺陷报告的软件安全漏洞检测与识别的统一框架

针对安全关键领域的智能软件系统对软件自身安全和信息安全要求较高的问题，为了辅助软件开发和维护人员快速识别隐藏在代码中的安全漏洞，并且辅助维护人员及时修复相关漏洞。我们拟结合自然语言处理、程序分析和深度学习技术开发一套自动化的代码安全漏洞检测与安全缺陷报告识别工具，代替人为制定相关规则检测代码安全漏洞，以及人工判定缺陷报告是否和安全相关，指导维护人员在软件测试的后期维护中及时消除安全漏洞，提高软件质量，保障软件安全，避免开发或维护人员因缺少安全领域知识而将软件安全漏洞漏报或者误报，给智能软件系统带来严重的安全隐患。具体目标如下：

（1）**代码安全漏洞检测的目标**：以往通过代码相似度计算以及模式匹配的方法会产生大量的误报，需要大量人工参与以降低误报，并且针对不同的安全漏洞需要制定不同的检测规则和策略，缺少统一适用的检测框架，因此我们的目标是通过搜集大量的安全漏洞模式、建立软件安全漏洞特征库，并结合程序分析和深度学习技术建立一个统一的安全漏洞检测框架，以预测代码是否含有某种指定类型的安全漏洞，在软件开发过程中保障代码的质量，提高软件自身的安全性。

**（2）安全缺陷报告识别的目标：**智能软件系统的复杂性，使得软件开发和测试通常都是团队合作完成的，每个开发人员和测试人员都可能在开发和测试的过程中提交大量的缺陷报告，由于软件测试的充分性难以保证，即使软件已经被发布，用户在使用软件的过程中也可能发现一些与安全相关的缺陷，这些缺陷往往也是以缺陷报告的形式提交到缺陷追踪系统中的，这些缺陷一旦被发现，必须被及时修复和打补丁，否则将有可能给恶意攻击者以可乘之机，然而以人工的方式从这些大量的缺陷报告中快速准确地识别需要优先修复的安全相关的缺陷报告，是非常困难的，因此我们的目标是能够借助软件仓库的历史信息并结合自然语言处理和数据挖掘技术从大量的缺陷报告中有效识别安全相关的缺陷报告，提高安全缺陷报告识别的召回率，避免安全缺陷被误识别为非安全缺陷而导致软件安全隐患。

**（3）构建完善的智能软件历史故障库：**“前事不忘，后事之师”，历史故障库记录了智能软件的历史故障及其修复信息，这些信息有助于指导测试人员重点关注那些易出错以及频繁出错的模块，并且根据这些信息可以辅助安全工程师准确识别安全相关的缺陷，避免测试人员在故障诊断上浪费多余精力，避免在安全缺陷识别上出现误判和漏判，提升测试人员的故障诊断能力和效率，还可以为维护人员修复同类故障提供参考。

#### 2.2.6建立数据驱动式的回归测试理论体系

我们拟建立一套完整的数据驱动式测试理论体系，对于软件生命周期中产生的数据和软件测试过程中产生的一系列数据，都作为数据驱动式测试方法的输入，进而建立集数据采集、数据筛选、数据处理、数据存储、数据挖掘、数据预测等诸多方法为一体的测试理论体系，提取出与软件质量及测试相关的内容进而丰富数据驱动式测试技术，完善智能软件测试框架，提升智能软件测试能力。具体研究目标如下：

**（1）建立高效的数据预处理流程：**该流程可对软件生命周期及测试过程产生的数据进行采集与筛选，构建数据价值密度指标，对海量数据的价值密度及可挖掘程度进行细致刻画，剔除无效扰动信息，保留高价值数据，存入数据库，并将不规则的文本信息规范化成可直接利用的数据信息。

**（2）提出基于数据挖掘的数据处理方法集：**该方法集可提取数据集智能蕴含的数据价值，构建合适的数据模型并应用适合的挖掘技术，对于每一类数据类型，都有相应的高效的数据处理方式。

**（3）制定数据驱动式的测试框架：**该测试框架可利用数据挖掘技术提取潜在的数据信息，指导测试人员进行高效的回归测试，对之前的测试阶段进行重新改进与优化，从而达到对智能软件的高效测试。具体包括测试模型的改进、生成算法的优化、判定策略的完善和故障诊断与定位能力提升。

### 2.3 拟解决的关键科学问题

#### 2.3.1智能软件测试科学建模问题

模型是测试的依据、出发点和归宿，正确的模型是测试效果的保障，所以，科学建模是智能软件系统测试的需要解决的首要问题。

传统软件测试领域已经存在一部分测试建模的相关研究工作，但智能软件具有数据处理层级化、决策逻辑不受控、系统输出难以验证等特点，且实际运行成本较高，传统的软件测试建模技术不能完全照搬照用。智能软件的测试建模问题具体包括如何对智能软件进行模块整合，对功能模块抽象化处理，如何搭建科学的测试模拟平台，如何建立完备的智能软件模型库，如何构建科学的描述方式，精确的刻画待测任务等子问题。

#### 2.3.2智能软件测试用例生成与优化问题

测试用例的生成问题是软件测试领域关键问题之一，对于智能软件而言，用例的生成算法在借鉴传统软件测试领域已有算法的同时，也需要基于场景与模型进行动态调整优化。所以我们需要考虑在已有的生成算法的基础上，如何基于对智能软件的场景和模型分析，构建测试用例生成算法框架？如何调整相应的生成策略，对智能软件中遗漏的场景或模型中缺失的部分进行充分性测试？如何保证生成的测试用例集具有代表性、充分性、高效性、可执行性与可判定性等特征？

智能软件测试用例的优化问题指如何在不影响测试效果的基础上删减一批无效或冗余测试用例以及如何通过改变用例执行顺序，尽快的揭示软件安全故障的问题。我们需要考虑测试用例无效或冗余的标准是怎么样的？什么样的用例是无效的或冗余的？无效或冗余的条件是什么样的？这些问题需要我们基于场景和模型进行逐一探索。另一方面，我们是否可以寻找一些新的排序标准应用在智能软件测试用例排序中来？不同排序标准对应的应用场景是怎样的？能否给出理论证明或实验模拟论证比较不同排序方法的优劣？

#### 2.3.3智能化测试预言及判定问题

测试预言及判定一直是传统软件测试中一个很难解决的问题，智能软件执行结果不易观察且具有随机性，测试预言及判定更加复杂。我们拟建立智能化判定方法体系，关键问题是如何将智能化判定方法体系具体应用在智能软件中，具体包括，如何根据智能软件特点，总结出多样的执行判定方法？针对不同的执行判定方法，如何总结出具体实现方案是什么？其适用场景或条件是什么？优劣与准确性上彼此差异在什么地方？例如，智能化判定方法体系中包含基于蜕变测试的判定方法，那么如何根据某一具体的智能软件识别出对应的蜕变关系？其次，测试判定策略能否动态调整也是一个难题，我们对智能软件的认知是从无到有的，如何根据测试情况的变化动态调整判定策略，甚至基于判定策略的修改调整测试用例的生成与排序都是我们需要解决的科学问题。

#### 2.3.4智能软件的测试评估科学性与多样性问题

智能软件测试的目标同样是充分测试，但是与普通软件不同的是，智能软件测试评价系统缺失，软件质量评价指标缺失，我们需要考虑如何建立科学的测试评价体系。软件测试的评估往往是多维度的（例如条件覆盖、分支覆盖、语句覆盖、MC/DC覆盖等），单一维度的充分性覆盖并不能表明测试的充分性覆盖，我们需要解决如下问题：如何基于场景与模型多样化软件测试评估指标？如何科学性地逐一论述不同评估指标确定的依据、适用的场景、彼此的差异与强弱？如何根据多维覆盖指标的协同评估进一步对软件质量进行可靠性评价？

#### 2.3.5智能化软件的安全漏洞检测与识别问题

为了尽早识别软件中隐藏的安全问题，需要解决如何在降低人为参与的条件下尽可能的识别出各种原因导致的安全漏洞问题。目前已有的自动化安全漏洞检测方法能够检测的安全相关的缺陷类型有限，并且针对不同类型需要不同的检测方法，缺少适用于绝大部分安全漏洞检测的统一框架。由于智能软件系统的复杂度以及流行度越来越高，缺陷类型也越来越多，因此研究适用于智能软件系统代码安全漏洞检测的统一框架刻不容缓。如何根据某一种类型的安全漏洞，自动提取待测程序中的关键要素并自动学习获得其代码语义信息和结构信息是代码安全漏洞检测需要重点解决的难点问题。除此之外，还需要在软件测试之后根据缺陷追踪系统中开发、维护人员或用户报告的缺陷信息尽早地识别出其中与安全相关的软件漏洞。如何有效利用软件历史仓库中的大量缺陷报告信息，从海量的缺陷报告中自动识别出可能危及智能化软件系统安全的缺陷报告是一个亟待解决的难点科学问题，而且安全缺陷报告识别需要具体解决类不平衡、数据噪音和数据稀疏等三个难点问题。

#### 2.3.6数据驱动式测试与回归测试结合问题

在经过测试建模、测试用例生成与执行、测试评估等步骤之后，很难保证测试充分性，特别是在软件系统经过修改或者完善之后，因此，需要将回归测试加入算法框架，在测试的充分性评估未通过的时候，考虑如何利用先前的测试经验与教训，进一步认识测试对象，优化测试技术，改善测试效率等。更具体的来说，对于算法执行过程中产生的软件数据和测试数据，如何利用数据挖掘算法？怎样对数据进行充分利用？怎样构建数据驱动式测试框架？数据驱动式框架对测试算法的每一步能产生何种影响？怎样产生改进和修正作用？如何利用数据驱动改进测试模型，优化生成算法，完善判定策略等，都是值得研究的科学难题。

## **拟采取的研究方案及可行性分析**（包括研究方法、技术路线、实验手段、关键技术等说明）；

### 测试模型的建立

#### 研究方法

本课题将结合传统软件测试的建模方法理论和应用实例，探索高效的面向智能软件的测试模型建模技术。针对智能软件可能存在的测试平台不易搭建、测试任务不易描述及测试模块难以独立执行等较为困难的问题，结合智能软件的历史模型库和待测软件显著特征，研究利用软件仿真模拟测试平台和利用形式化技术准确描述待测任务的方法。

#### 技术路线

（1）构造智能软件模型库的技术路线：首先对当前智能软件进行详细调查研究，加深对智能软件的组织架构、功能特点的理解。然后收集一批实际应用的智能软件，根据不同特征，对其进行合理分类，对不同类别的智能软件构造UML模型描述，在此基础上，构造一个智能软件模型库。其次，分析智能软件的历史数据及应用环境，结合相关领域知识，重点探究从场景及任务中可提取出哪些软件质量相关的特征，以及如何从场景及任务中提取出这些特征。在得到这些与软件质量相关的特征后，需要分析智能软件系统为适应或满足这一特征而必须要满足的运行时特性。在指定的应用场景下，再依据这些运行时特性为待测的智能软件制定具体的测试任务。

（2）智能软件测试仿真平台的技术路线：综合智能软件模型库、传统软件测试建模技术和现有智能软件测试建模技术三方面的研究，以开源智能软件仿真平台为基础，添加智能软件相关的必要功能模块，应用软件工程方法，开发一个具有高仿真性与高可测性的智能化仿真平台。

#### 关键技术

（1）智能软件模型库的关键技术，包括基于UML的软件应用系统建模技术，软件质量管理的关键技术，基于深度学习的特征提取技术，软件测试模型及测试任务的形式化等。

（2）智能软件测试仿真平台的关键技术，包括支持基于传统软件工程的软件测试模型（V/W/H/X等）的仿真技术，面向嵌入式软件测试的仿真建模技术，基于可靠性的软件仿真测试平台开发技术等。

### 测试用例生成与优化

#### 研究方法

本课题将结合传统软件测试的生成与优化理论，探索高效的面向智能软件的测试用例生成与优化技术。针对智能软件测试可能存在的测试用例覆盖程度低、边缘场景易遗漏等问题，以大量、多类型（历史、实时、开发与维护等）数据为基础，研究智能软件系统的测试用例生成与优化技术。

#### 技术路线

（1）基于场景的测试用例生成技术关注系统层面的测试，而基于功能的测试用例生成技术更加关注系统的独立模块，考虑将两种技术进行优势互补，实现场景到功能的贯通，在此基础上，可以通过使用软件系统每个独立模块的测试结果来度量智能软件系统在给定应用场景及任务下的功能质量。

（2）对于智能软件的测试用例选择，通过组合优化技术在测试用例集中选择满足一定组合覆盖条件的测试用例子集，代替原有测试用例全集；研究满足不同组合覆盖条件测试用例集在故障检测能力方面的差异，找出在一定条件下具有最优故障检测能力的组合覆盖条件；

（3）对于智能软件的测试用例排序，通过借鉴适应性随机测试的思想，每次在测试用例全集中选择与已执行测试用例最远的测试用例；提出新的距离度量指标量化输入之间的距离并且研究不同的距离度量方法对排序后测试用例集故障检出速度的影响；

（4）对智能软件的测试用例执行，考虑将控制理论引入智能软件测试中，利用测试的历史信息，控制下一个测试用例的选择，达到尽快揭示智能软件故障的目的。

#### 关键技术

基于场景的测试用例生成与基于功能的测试用例生成相结合的技术，利用组合覆盖生成测试用例集技术，智能软件测试用例排序技术，基于控制论的测试用例生成与优化技术等。

### 测试用例的执行判定

#### 研究方法

本课题在传统测试预言问题研究的基础上，探索高效的面向智能软件的测试结果判定技术。采用蜕变测试、变异测试、交叉引用方式及特征提取方式等技术手段，解决智能软件测试结果不明确、人工判定准确度低的问题。

#### 技术路线

1. 以蜕变关系为基础实现智能软件系统测试结果验证。首先对待测软件应用不同蜕变关系识别技术，得到蜕变关系集，然而利用变异分析技术研究该蜕变关系集的故障检测能力，为智能软件的蜕变关系选择提供启发式规则，在此基础上选择合理的蜕变关系作为执行结果判定依据，最后判断智能软件系统的多个测试用例的实际输出结果之间是否满足该蜕变关系，若不满足，则表明待测系统应满足的必要属性被违反，即系统潜藏故障。
2. 利用交叉引用方式判断智能软件系统的行为是否符合预期。以无人驾驶汽车为例，具体方法是将同一输入在不同公司开发的无人驾驶汽车上执行，观察无人驾驶汽车的行为，若存在一辆或者若干辆无人驾驶汽车的行为与其它不一致，表明其中的某些无人驾驶汽车存在故障。

（3）利用特征提取方式预判定智能软件执行结果是否正确。首先对测试用例集的执行结果进行分类，执行正确、执行错误和结果未知，结合数据挖掘方法，对智能软件执行输出进行特征提取，计算当每一种特征出现时，执行正确的概率。最后通过当前用例的特征分析，计算出执行结果正确的概率，达到对执行结果的预判定。

#### 关键技术

蜕变关系的识别技术，利用变异分析技术研究该蜕变关系集故障检测能力的技术，合理蜕变关系选择技术，智能软件多维输出空间的特征选择技术，数据挖掘分类技术等。

### 测试充分性评估

#### 研究方法

本课题将结合传统的软件测试评估技术和智能软件的特征，针对智能化系统测试体系尚未发展完全、缺少严格的测试评估指标、智能软件测试模型难以一次确定而需要不断调整优化等较为困难的问题，制定科学的评估指标，根据指标评估测试充分性，并分析评估结果，优化智能软件测试模型。

#### 技术路线

（1）制定科学评估指标：借鉴传统软件测试评估中的路径覆盖率、组合覆盖率等指标，确定面向智能软件测试的充分性评估指标（代码覆盖、模型覆盖等），该指标可以定量的从某一方面对智能软件测试进行衡量，给出待测智能软件的测试用例集覆盖情况和遗漏测试场景等，最终从多个维度给出待测智能软件的可信度或可靠度分析。在指标确定后，还需要对该指标的科学性和可靠性进行规范证明，明确该指标的适用场景、研究意义等，并利用具体的案例研究详细解释该指标的应用。

（2）测试充分性评估及结果分析：应用之前提出的智能软件测试评估指标，对测试用例集进行评估，分析其在覆盖充分性上的不足，生成完备的智能软件测试报告。对整个测试流程进行评估，检查在某一维度下测试用例集是否达到了充分覆盖，是否存在重要场景遗漏的情况。若评估未通过，则进入数据驱动式测试阶段（图2-1所示），重新优化测试模型，生成算法及判定策略等，开启新一轮测试，直到测试评估结果达到预期。

#### 关键技术

（1）软件测试评估技术，包括白盒方法（如语句覆盖、判定覆盖和路径覆盖等）、黑盒方法（基于模型的软件测试覆盖率评估方法等）以及灰盒方法等。

（2）充分性评估及结果分析技术，包括软件测试充分性度量方法，基于覆盖率的测试用例评审方法，回归测试技术。

### 安全漏洞的检测与识别

#### 研究方法

本课题将结合深度学习和自然语言处理领域的最新研究成果，探索有效的面向智能化软件系统的安全漏洞检测与安全缺陷报告识别技术，针对智能软件系统对安全可靠性要求较高以及数据驱动的智能软件测试产生的缺陷数量和缺陷种类可能较多、人工排查安全缺陷较为困难的问题，从程序源代码和软件缺陷报告两个角度，同时结合深度学习、自然语言处理和程序分析技术，研究安全相关的漏洞检测和安全缺陷报告识别方法。

#### 技术路线

（1）代码安全漏洞检测的技术路线：结合程序分析和深度学习技术以及安全漏洞特征模式库，自动识别待测程序中的安全漏洞，整体技术路线如图3-1所示。 首先，针对想要检测的某种类型的安全漏洞，从安全漏洞特征模式库中提取相关的漏洞特征，确定产生这些安全漏洞的可能关键元素，然后对待测程序建立数据依赖和控制依赖图，提取关键元素所对应的程序切片并生成可能包含漏洞结构信息和语义信息的关键代码片段，再利用结合注意力机制的基于AST的循环神经网络和普通的循环神经网络对代码标准化后的token串分别进行结构和语义编码。最后，将两种编码向量加权得到最终的混合代码向量表示，并送入softmax层，以预测代码中是否含有此类安全漏洞。

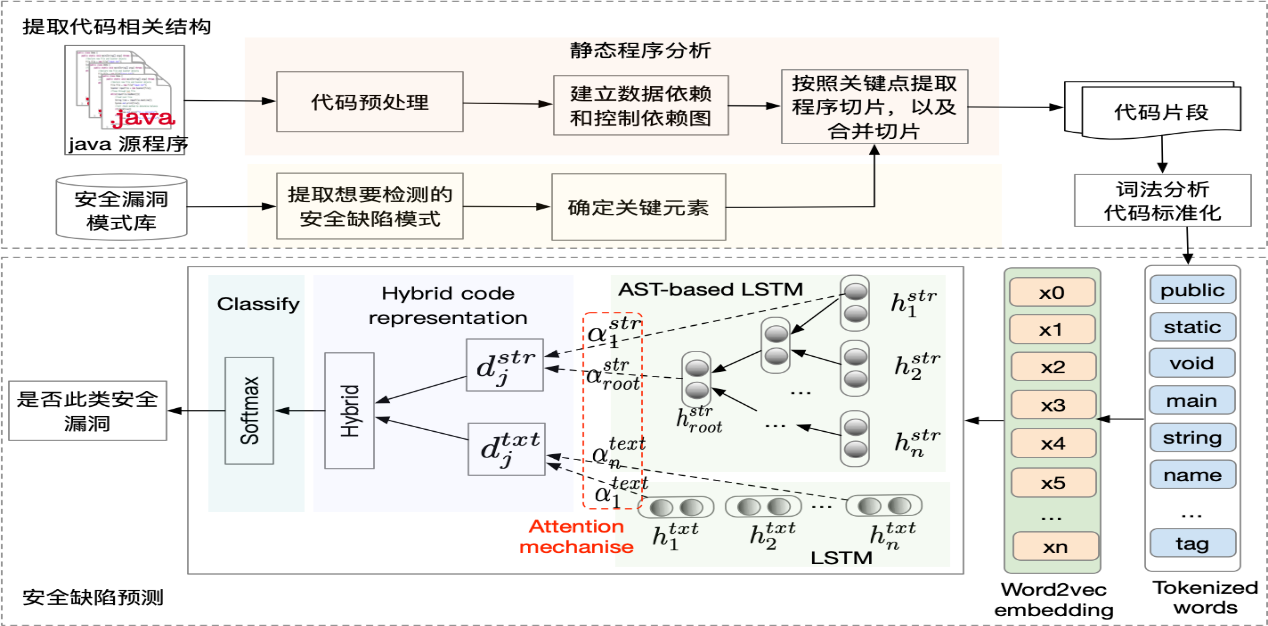


图3-1 代码安全漏洞检测的技术路线图

（2）安全缺陷报告识别的技术路线：将排序学习、自然语言处理中的词嵌入、机器学习和深度学习技术相结合识别安全缺陷报告，详细技术路线图如图3-2所示。首先，针对非安全缺陷报告同样包含与安全相关的关键词从而导致训练数据存在噪音以及训练数据严重失衡的问题，采用learning to rank技术按内容相关性对训练数据集中的非安全缺陷报告进行排序，选择与安全缺陷报告内容相关度较低的非安全缺陷报告作为预测模型的训练数据集，并采用word embedding 技术将去噪后的训练数据集中的缺陷报告转化为分布式向量表示，用以训练基于机器学习的预测模型，最后利用预测模型识别新的缺陷报告是否为安全缺陷报告。针对源项目有可能缺少项目的历史缺陷信息或者项目历史数据中包含的安全缺陷数量较少的问题，采用迁移学习的方法建立跨项目预测模型，实现跨项目的安全缺陷报告识别。

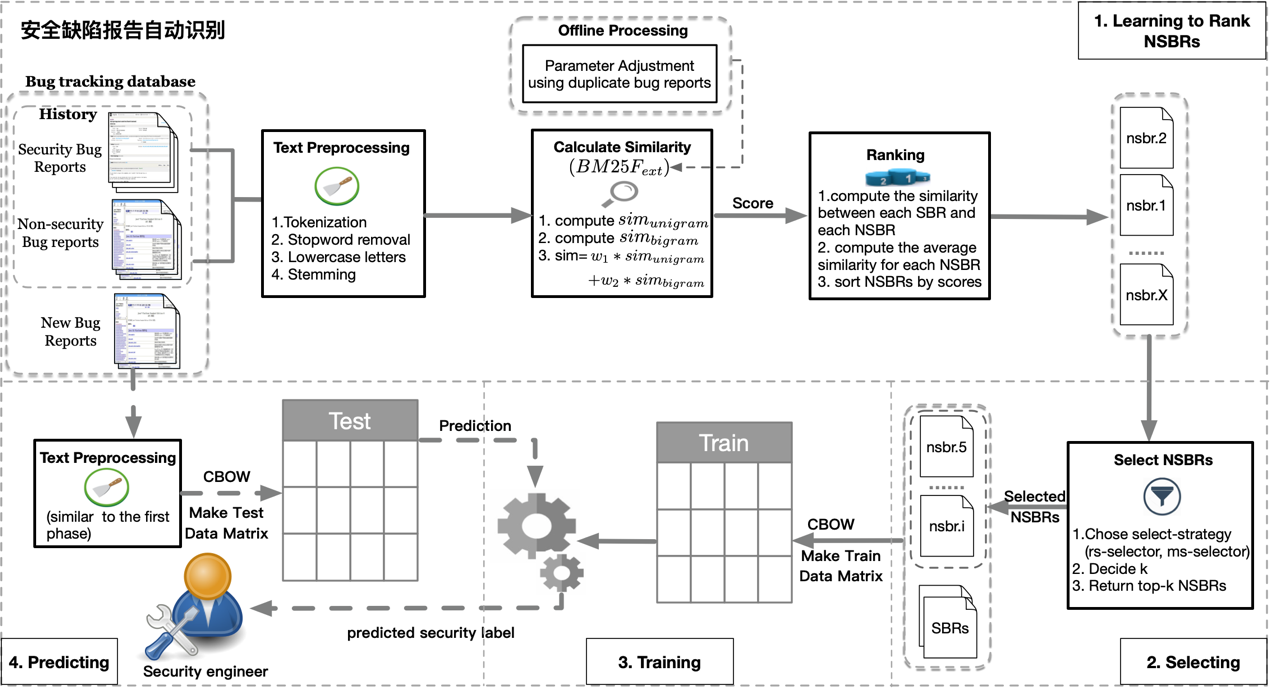


图3-2 安全缺陷报告识别的技术路线图

#### 关键技术

（1）代码安全漏洞识别涉及的关键技术：基于程序分析的关键元素切片提取方法，代码标准化方法，基于LSTM的代码表示学习方法，基于注意力机制的向量混合表示方法，基于深度学习的安全漏洞预测技术。

（2）安全缺陷报告识别涉及的关键技术：基于learning to rank的训练数据去噪方法，基于word embedding的缺陷报告表示学习方法，基于迁移学习的跨项目预测方法。

### 数据驱动式测试

#### 研究方法

针对智能软件系统的学习深层知识、融合跨界数据、数据处理层级化、决策逻辑不受控、系统输出难以验证、集成群体智慧等特点带来的问题，本课题利用软件训练数据、开发与测试中产生的海量数据，探索数据驱动式测试方法与技术，结合大数据处理和数据挖掘相关技术，优化智能软件测试策略，构建面向智能软件的数据驱动式测试框架。

#### 技术路线

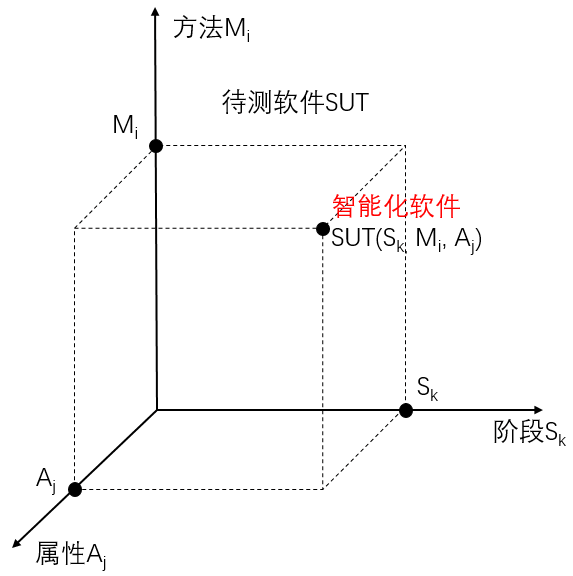
1. 数据选取与数据处理的技术路线：数据源包括软件数据（如软件版本信息）、输入数据、训练数据、中间数据和测试数据（如测试报告、安全报告、测试日志等），我们首先构建数据价值密度指标，对不同类型和来源的数据的价值密度及可挖掘程度进行区分，其次针对不同类型及价值密度的数据选择适当的数据提取技术，采用自然语言处理对海量文本信息进行关键词提取，采用分类预测算法对测试报告的判定结果进行整理等，建立一个完备的产品运行与测试数据库。
2. 测试策略优化的技术路线：数据处理完成之后，利用数据驱动对测试策略进行合理优化。具体包括，模型的数据输入更加丰富，测试报告显示可能存在部分测试场景的遗漏，若干模块功能相近出现冗余等，这些可指导测试人员重新优化测试模型，调整生成与优化策略；通过对测试报告的判定结果的分类预测，总结出更加明确的执行结果判定策略，获取新的蜕变关系等。

#### 关键技术

1. 数据选取与数据处理的关键技术：数据采集与数据挖掘技术、数据并行处理技术等。
2. 测试策略优化的关键技术：自适应测试模型技术、自适应测试生成与优化技术及自适应执行判定技术等。

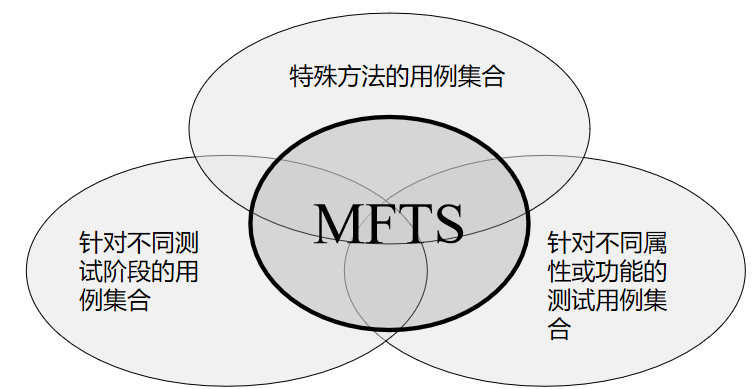
## **本项目的特色与创新之处；**

1. 我们系统梳理了软件测试方法体系（见图4-1a），将100种软件测试方法分成四大类：Mi各种专门的软件测试方法；Aj针对各种属性和方面的测试方法；Sk各个开发阶段的软件测试方法；SUTl各种类型的软件测试方法。智能化软件是一种属于SUTl的特殊类型的软件，因此需要对智能化软件应用已有的软件测试方法体系进行系统的测试（见图4-1b）。



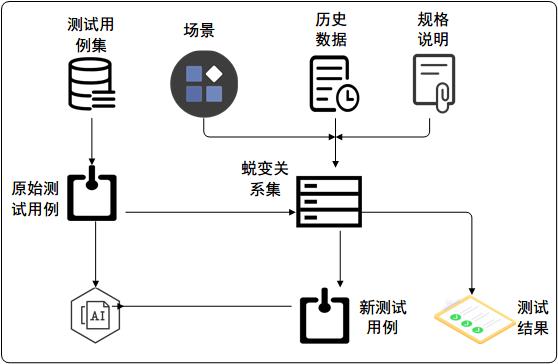
**图4-1a 软件测试方法体系 图4-1b软件测试方法体系中的智能软件测试**

1. **我们的智能软件测试生成算法从不同角度出发，提出一套测试生成方法体系，基于不同测试场景、测试目标、测试粒度生成合适的测试用例集合。**我们认为对任何待测软件而言，客观上都存在一个最小的可以触发潜在缺陷和错误的测试用例集合（Minimal Failure-causing Test Suite, MFTS）。具体来说，我们从智能软件不同测试阶段出发，针对智能软件各种属性或功能，结合特殊的测试生成方法（组合测试、变异测试等）生成测试用例集合，尽可能从多个角度覆盖到MFTS，尽可能保证MFTS不遗漏，从而对智能软件进行科学有效的测试，发现潜在的缺陷和错误（如图4-2所示）。



**图4-2 测试阶段、测试属性、特殊方法三个角度覆盖MFTS**

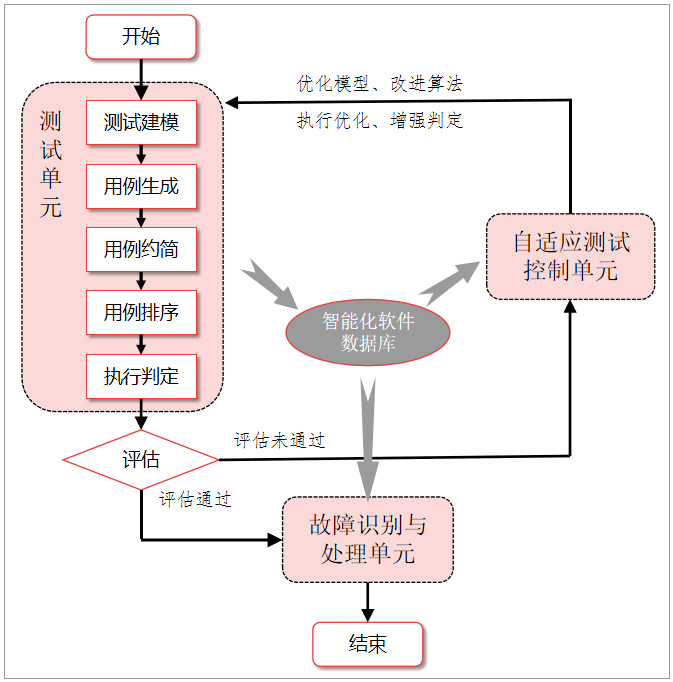
**3、我们利用蜕变测试技术解决智能软件测试中的测试预期问题：**智能软件系统输出的可预言性导致测试预期问题在智能软件测试中频繁出现，且难以解决。本课题利用蜕变测试技术中的蜕变关系来缓解智能软件测试中的测试预期问题。与已有的工作相比，本课题多角度、深层次地获取蜕变关系并利用蜕变测试技术来补足智能软件系统的测试用例集（图4-3所示），具体来说：（a）提出了基于场景变换归纳蜕变关系；深入挖掘历史数据来提取故障模式，从而获取蜕变关系；分析规格说明书来识别蜕变关系等3种蜕变关系获取方法；（b）在蜕变关系集的基础上，已有的测试用例集可以派生出更多的新测试用例，补足已有的测试用例集；（c）通过验证多个输入的结果是否违反蜕变关系来判断智能软件系统是否存在故障。

****

**图4-3 蜕变测试框架**

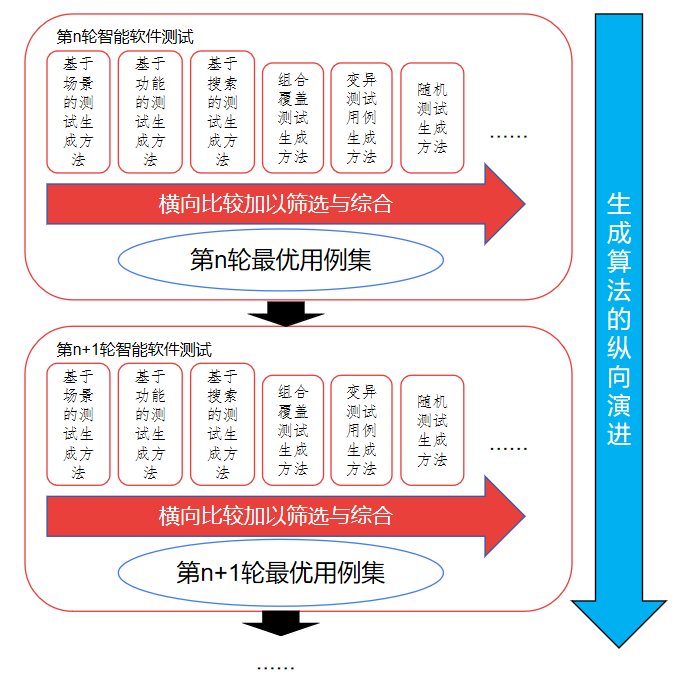
**4、**我们结合深度学习、程序分析和自然语言处理技术建立统一的安全漏洞检测与安全缺陷报告识别模型。传统的代码安全漏洞识别模型只能检测常见的有限的几种缺陷类型，并且不同的缺陷类型需要不同的检测方法，缺少适合所有安全漏洞的统一检测框架，**本项目通过建立安全漏洞特征模式库和关键代码要素提取，并结合深度学习技术建立统一的安全漏洞预测模型**，如图3-1所示为检测智能软件系统中具有复杂模式的安全漏洞提供了一种新思路。传统的软件测试更多关注导致软件失效的故障，对潜在的软件安全漏洞的识别缺乏准确有效的测试方法，**本项目将安全缺陷的识别处理技术应用于智能软件系统，借助软件仓库和开源社区为我们提供的大量缺陷报告信息及其修复的历史信息，基于自然语言处理和机器学习技术从海量的缺陷报告中自动识别出可能危及智能软件系统安全的缺陷报告**，如图3-2所示避免安全缺陷的误识别给智能软件系统安全带来的隐患，该方法是软件安全测试的一种必要和有益的补充。

5、**我们采用了灵活的自适应测试框架应用于智能软件测试中**（图4-5所示），测试建模策略、用例生成算法、约简排序算法、执行判定策略等都不是固定的，我们不必过于担心每一步骤是否考虑周全，而是可以根据测试结果反向改进建模、生成、优化、判定等策略。实际测试过程中，测试人员对具体智能软件的认知是从无到有的，从模糊到具体的，这就意味着初始模型可能偏差很大，初始用例集可能遗漏很多场景，也可能重复测试同一类场景，执行判定规则可能非常缺乏。但经过一轮测试之后，通过对测试数据分析，测试执行过程及结果研究，测试人员可能对智能软件有了深层次认识，也会重点关注测试活动中难以测试的特殊场景，从而我们可利用自适应测试 框架改进测试模型，调整测试策略，丰富判定规则。我们制定灵活的自适应测试框架，尊重了人类对陌生事物的认知规律，保证了测试活动的有序高效运行。



**图4-5 智能软件测试自适应框架**

1. **我们将数据驱动式的测试技术应用在智能软件测试框架中**。智能软件离不开数据驱动，智能软件测试更离不开数据驱动测试，智能软件数据包括算法训练数据、软件运行数据（输入输出数据、运行日志、输出数据、版本信息、说明文档等）和软件测试数据（测试模型数据、测试用例集、测试报告等）。本项目建立一整套完备的数据驱动式框架，包括数据采集、数据转换、数据存储、数据计算、数据挖掘、数据预测及数据决策等关键步骤，对软件开发、运行、测试活动中产生的不同类型的数据建立科学高效的数据处理流程，挖掘不同类型的数据中潜在的价值。其次，数据驱动的目标是利用数据信息，改进智能软件测试流程，包括修正测试模型、改进生成算法、完善判定策略等，达到对智能软件的全方位、多角度的测试。特别是，**测试用例的生成算法也是通过数据驱动不断演化改进而逐步完善的**（图4-6所示）。我们既可以对多种算法生成的用例集合进行横向比较与综合，得出最优用例集，也可以采用回归优化方法进行纵向比较，指导生成算法不断改进与完善，进一步提升智能软件测试效果。



**图4-6 数据驱动的智能软件测试生成算法的横向比较与纵向演进**

## **5．年度研究计划及预期研究结果**（包括拟组织的重要学术交流活动、国际合作与交流计划等）。

(1) 2020.1-2020.12 在已有研究的基础上，进一步完成软件测试理论与方法的调研，深入分析一批智能软件系统，研究各种软件测试理论与方法的更广阔的适用范围和使用方法。继续研究改进各种面向智能软件系统测试数据生成方法。进行相关论文的写作。每年积极参加软件测试领域几个主要的国际会议（ISSTA，ICST）。

(2) 2021.1-2021.12 研究面向智能软件系统的测试理论与方法的比较和组合，探索各种方法之间的相互关系和可能的组合情况。并将这些方法和技术进行实践研究。进行相关论文的写作。

(3) 2022.1-2022.12 研究智能软件系统数据驱动式测试方法的支持工具和配套管理，结合一些典型的智能化应用软件开发相应的支持工具、管理规范和文档。

(4) 2023.1-2023.12智能软件测试方法的实证研究，特别是建立针对智能化系统的完备的测试用例集和相应的自动化测试工具集，试验数据的收集，相关文档的完善。

(5) 2024.1-2024.12智能软件的安全漏洞检测与识别研究，进一步完善智能软件系统的数据驱动式测试理论、方法与技术，以及项目结题准备工作。

(6)在国内国际重要学术会议和期刊上发表学术论文20多篇，其中在国际顶级会议或学术刊物上5-10篇；出版软件测试学术专著或教材一本；参加国际学术活动或交流6-8次，争取举办相关领域国际学术会议一次。

# （二）研究基础与工作条件

## 1．**研究基础**（与本项目相关的研究工作积累和已取得的研究工作成绩）；

**项目组由南京大学聂长海教授团队、哈尔滨工业大学苏小红教授团队和北京科技大学孙昌爱教授团队组成，这三个团队一直工作在软件测试领域，具有深厚扎实的研究基础，各有特色且相互补充，例如南京大学团队在组合测试及测试用例生成优化等领域很有建树、哈工大团队在软件安全及智能软件系统方面具有很深的积累，而北京科技大学团队在测试预言及测试评估等领域具有扎实的基础。以下我们分别进一步介绍这3个团队的相关研究基础。**

**（1）南京大学团队**负责人聂长海教授在多年从事软件测试课程教学的基础上，结合科研实践，经过多年的努力，在清华大学出版社出了一本软件测试教材《软件测试的概念与方法》[1]，该书以软件测试方法的分类为线索，以软件测试的各种方法为内容，系统地介绍各种软件测试方法的概念、理论、特点和在工程实践中应用的例子。与已有的软件测试教材和相关书籍相比，该书特色是以各种软件测试方法为中心，系统地介绍到目前为止，几乎所有的可以查到或见到的软件测试方法，是对目前已经出版的软件测试书籍的一个重要补充.为了写好这本书，项目组调研1000多篇学术论文，100多本相关教材和专著，这些已成为本项目非常厚实的研究基础。该教材自2013年出版后，受到了很多来自高校和产业界的专家的好评，并于2015年入选江苏省高校重点建设的教材，将于2019年后重新修订，届时将有大量新的内容和更深入的研究成果充实进去。

我们在各种软件测试方法的比较方面也已经做了很多基础性工作和积累，例如我们针对组合测试、随机测试与自适应随机测试在最小故障模式的检测能力上进行了系统的比较研究[2]，文章发表在CCF指定的很有影响的B类期刊上。同时我们对这三种方法的实证研究已被软件工程顶级期刊TSE正式接受[2’]。

我们已经在测试方法的交叉结合和组合应用方面做了一些初步尝试，例如给出了可以动态修改测试模型和测试策略的自适应组合测试框架[3]，并进行了实证研究[4]，在此基础上我们系统地整理了这个领域的工作，长文[3+4]将投软件工程领域最高级别的期刊TOSEM.

我们还将基于搜索的软件测试与组合测试结合，建立了基于搜索的组合测试分支[5]，提出了一种新的用于覆盖表生成的离散粒子群算法[6]，我们还总结了目前组合测试领域中搜索技术的研究现状[7]，讨论了基于搜索的组合测试未来的研究方向。

我们具有针对某个测试方法深入研究的成功经验，例如我们在组合测试领域具有10多年的深入研究经验，发表相关学术论文40多篇,获得多项软件著作权和发明专利。其中最有代表性研究成果是提出了关于组合测试的最小故障模式概念，并给出了一种基于最小故障模式的组合测试方法学，该项成果发表在软件工程领域著名期刊ACM Transactions On Software Engineering and Methodology上[8]；另一项代表性工作是在充分分析已有文献的基础上，结合自己在组合领域已有研究实践，总结出其中的关键问题、方法、重要应用和未来发展方向，论文发表在计算机领域著名学术期刊ACM Computing Survey上[9]，得到了国际同行在FSE，ICSE，TSE 和TOSEM等一流国际会议和期刊上发表论文的引用和关注。除此之外，我们已经在该领域建立了完整的组合测试支持工具，并进行广泛的实证研究，且于2015年初出版了专著《组合测试》[10]。我们还建设了最完整的组合测试研究文库(repository)，收集了700多篇相关研究论文，放在我们的网站上供所有研究人员共享，网址是：gist.nju.edu.cn。最近我们又在软件工程领域最好的期刊TSE录用了一批高水平论文[11,12]。

[1] 聂长海 ，软件测试的概念与方法 ，北京: 清华大学出版社, 205页, 2013/05/01

[2] **Changhai Nie**, Huayao Wu, Xintao Niu, Fei-Ching Kuo, Hareton Leung and Charles J. Colbourn. Combinatorial Testing, Random Testing, and Adaptive Random Testing for Detecting Interaction Triggered Failures. Information and Software Technology 62 (2015) 198–213.

[2’] Huayao Wu, Changhai Nie, Petke Justyna, Yue Jia and Mark Harman, An Empirical Comparison of Combinatorial Testing, Random Testing and Adaptive Random Testing, IEEE Transactions on Software Engineering (TSE), in press, 2018 [[IEEE Xplore](https://ieeexplore.ieee.org/document/8405609/)]

[3] Nie, Changhai、Chen, Siyang、Leung, Hareton、Cai, Kai-Yuan ，A case study of adaptive combinatorial testing ，2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, COMPSACW 2013, pp 47-52, Kyoto, Japan, 2013

[4] Nie, Changhai、Leung, Hareton、Cai, Kai-Yuan， Adaptive combinatorial testing ，13th International Conference on Quality Software, QSIC 2013, pp 284-287, Nanjing, China, 2013

[3+4] Changhai Nie, Huayao Wu, et al. A methodology of adaptive combinatorial testing, under revise (will be resubmit to TOSEM).

[5] Changhai Nie, Huayao Wu, Liang Yalan, Hareton Leung, Fei-Ching Kuo and Zheng Li. Search Based Combinatorial Testing, The 19th Asia-Pacific Software Engineering Conference (APSEC 2012). 2012, Dec. Hong Kong.

[6] Huayao Wu, Changhai Nie, Fei-Ching Kuo, Hareton Leung, Charles J. Colbourn. A Discrete Particle Swarm Optimization for Covering Array Generation. IEEE Transactions on Evolutionary Computation, 19(4): 575-591, 2015

[7] Huayao Wu, Changhai Nie. An overview of search based combinatorial testing. In: Phil McMinn eds. The 7th International Workshop on Search Based Software Testing (SBST). Hyderabad, India: ACM, 2014, 27-30

[8] Changhai Nie, Hareton Leung. The minimal failure-causing schema of combinatorial testing, ACM Transaction on Software Engineering and Methodology (TOSEM), 20(4), 2011.

[9] Changhai Nie, Hareton Leung. A survey of combinatorial testing, ACM Computing Survey, vol. 43, issue # 2 ,Jan. 2011.

[10] 聂长海，组合测试，科学出版社，北京，2015

[11] Xintao Niu, Changhai Nie, Hareton Leung, Yu Lei, Xiaoyin Wang, Jiaxi Xu and Yan Wang, An Interleaving Approach to Combinatorial Testing and Failure-inducing Interaction Identification, IEEE Transactions on Software Engineering (TSE), in press, 2018 [[IEEE Xplore](https://ieeexplore.ieee.org/document/8438906/)]

[12] Xintao Niu, Changhai Nie, Hareton Leung, Yu Lei, Xiaoyin Wang, Jiaxi Xu and Yan Wang, An Interleaving Approach to Combinatorial Testing and Failure-inducing Interaction Identification, IEEE Transactions on Software Engineering (TSE), in press, 2018 [IEEE Xplore]

**（2）**哈尔滨工业大学研究团队具有多年从事程序分析及智能软件相关理论和技术的研究基础。课题组负责人苏小红教授主持或作为主要研究人员参与完成了多项软件工程领域的研究课题。

（1）从事国家自然科学基金项目研究工作的基础

课题组成员苏小红自2000年以来，先后主持或参与完成多项国家自然科学基金项目的研究工作。

作为技术负责人，分别于2002年和2005年年底完成两项国家自然科学基金（批准号：69975005，60273083）的研究工作，排名第2。

作为项目负责人，于2009年年底主持完成一项国家自然科学基金“基于程序转换和语义分析的编程题自动评分方法研究”（批准号：60673035，项目起止时间：2007年1月-2009年12月）的研究工作。研究成果已应用于C语言编程题考试自动评分系统中，该系统先后被国防科大、北京邮电大学、东北大学等60余所院校试用。在该项目中对程序静态分析、程序标准化、程序转换、程序相似度计算、程序匹配等关键技术进行了研究和探讨。

作为项目负责人，于2011年年底完成国家自然科学基金项目“数据挖掘和静态分析相结合的克隆代码缺陷检测及重构方法”（批准号：61073052，起止时间：2011年1月-2011年12月），重点研究基于数据挖掘和静态分析相结合的语法相似的克隆代码检测及相关缺陷检测方法。

作为项目负责人，于2015年年底完成国家自然科学基金项目“无定型克隆代码检测及重构方法”（批准号：61173021，起止时间：2012年1月-2015年12月），结合软件自动测试、程序静态分析、数据挖掘和模式聚类等理论，研究建立将克隆代码检测、相关缺陷检测和克隆代码重构有机融为一体的无定型克隆代码检测与重构模型。

作为项目负责人，于2011年年底完成了软件缺陷检测方面的国防基础科研项目1项，该项目主要为航空航天等安全关键系统提供软件质量保障。

（1）发表论文和专利授权情况

软件缺陷检测方向发表的研究论文：

[1] Qiu J , Su X , Ma P . Using Reduced Execution Flow Graph to Identify Library Functions in Binary Code[J]. IEEE Transactions on Software Engineering, 2016,42(2):187-202.

[2] Qiu J , Su X , Ma P . Identifying functions in binary code with reverse extended control flow graphs[J]. Journal of Software: Evolution and Process, 2015, 27(10):793-820.

[3] Jing Qiu, Xiaohong Su and Peijun Ma, "Library functions identification in binary code by using graph isomorphism testings," 2015 IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER), Montreal, QC, Canada, 2015, pp. 261-270.

[4] Qiu J, Yadegari B , Johannesmeyer B , Su Xiaohong. Identifying and understanding self-checksumming defenses in software[C]// ACM Conference on Data & Application Security & Privacy. ACM, 2015: 207-218.

[5] Bian Y, Koru G, Su X, et al. SPAPE: A semantic-preserving amorphous procedure extraction method for near-miss clones[J]. Journal of Systems & Software, 2013, 86(8):2077-2093.

[6] Wang T , Harman M , Jia Y , et al. Searching for better configurations: A rigorous approach to clone evaluation[C]// Joint Meeting on Foundations of Software Engineering. ACM, 2013:455-465.

[7] Wang T, Wang K , Su X , et al. Detection of semantically similar code[J]. Frontiers of Computer Science, 2014, 8(6):996-1011.

[8] Su X , Zhang F , Li X , et al. Functionally Equivalent C Code Clone Refactoring by Combining Static Analysis with Dynamic Testing[C]// Proceedings of International Conference on Soft Computing Techniques and Engineering Application. Springer India, 2014: 247-256.

[9] Fanqi Meng, Xiaohong Su. Reducing WCET Overestimations by Correcting Errors in Loop Bound Constraints[J]. Energies, 2017, 10(12): 2113-2131.

[10]Meng F, Su X. WCET optimization strategy based on source code refactoring. Cluster Computing, 2017:1-10.

[11] 苏小红, 张凡龙. 面向管理的克隆代码研究综述[J]. 计算机学报, 2018, 41(3):628-651.

[12] 苏小红, 禹振, 王甜甜, et al. 并发缺陷暴露、检测与规避研究综述[J]. 计算机学报, 2015(11) :1557-1568.

[13] 逄龙, 苏小红, 马培军, et al. 流敏感按需指针别名分析算法[J]. 计算机研究与发展, 2015, 52(7):1620-1630.

[14] 边奕心, 王甜甜, 苏小红, et al. 一种语义保持的C克隆代码无定型过程提取方法[J]. 计算机研究与发展, 2013, 50(7):1534-1541.

软件缺陷预测方向发表的研究论文：

[1] Zhang F , Khoo S C , Su X . Predicting Change Consistency in A Clone Group[J]. Journal of Systems & Software, 2017, 134: 105-119.

[2] Fanlong Z , Siau-Cheng K , Xiaohong S. Machine-Learning Aided Analysis of Clone Evolution[J]. Chinese Journal of Electronics, 2017(06):26-32.

[3] Cholmyong P , Tiantian Wang, XiaoHong Su . An Empirical Study on Software Defect Prediction Using Over-Sampling by SMOTE[J]. International Journal of Software Engineering and Knowledge Engineering, 2018, 28(06):811-830.

[4] Fanqi Meng, Xiaohong Su, Zhaoyang Qu. Interactive WCET prediction with Warning for Timeout Risk[J]. International Journal of Pattern Recognition and Artificial Intelligence, 2017, 31(5): https://doi.org/10.1142/S0218001417500124.

[5] Syed Rizwan, Wang Tiantian, Su Xiaohong, Salahuddin, Empirical Study on Software Bug Prediction [C], 2017 International Conference on Software and e-Business,2017 December 28-30.

软件故障定位方向发表的研究论文：

[1] Gong D , Su X , Wang T , et al. State dependency probabilistic model for fault localization[J]. Information and Software Technology, 2015, 57:430-445.

[2] Dandan G , Tiantian W , Xiaohong S , et al. A test-suite reduction approach to improving fault-localization effectiveness[J]. Computer Languages, Systems & Structures, 2013, 39(3):95-108.

[3] Su X H , Gong D D , Wang T T , et al. A Survey of Automated Software Fault Localization Approach[J]. Applied Mechanics and Materials, 2014, 556-562:6102-6105.

[4] Wang Tianitan, Wang Kechao, Su Xiaohong, etc. Invariant based Fault Localization by Analyzing Error Propagation[J]. Future Generation Computer System, 2019,94:549-563.

[5] Wang K C, Wang T T , Su X H . Test case selection using multi-criteria optimization for effective fault localization[J]. Computing, 2018,100(8)：787-808.

[6] 苏小红, 王甜甜, 杨劭君, et al. 一种基于加权软件行为图挖掘的软件错误定位方法[J]. 计算机学报, 2016, 39(11):2175-2184.

[7] 王克朝, 王甜甜, 苏小红, et al. 软件错误自动定位关键科学问题及研究进展[J]. 计算机学报, 2015(11):2262-2271.

[8] 苏小红, 龚丹丹, 王甜甜, et al. 结合用例约简与联合依赖概率建模的错误定位[J]. 软件学报, 2014(7):1492-1501.

[9] 王克朝, 王甜甜, 苏小红, et al. 面向有效错误定位的测试用例优选方法[J]. 计算机研究与发展, 2014, 51(4):865-873.

在程序分析方向获得国家发明专利6项，申请国家发明专利2项：

[1] 苏小红,龚丹丹,王甜甜,马培军,王煜,赵玲玲 ，国家发明专利：基于状态依赖概率建模的软件错误定位方法 ，已授权， 2015.3.3, 201310099998.0

[2] 王甜甜,王克朝,苏小红， 国家发明专利：一种程序代码编程模式著作权归属检测模型的著作权归属检测方法，已授权，2015.5.6, 201210508663.5

[3] 龚丹丹,苏小红,王甜甜,马培军,王煜,赵玲玲 ，国家发明专利：一种面向错误定位需求的测试用例约简方法 ，已授权，201310099877.6

[4] 苏小红,龚丹丹,马培军,王甜甜,赵玲玲,王煜 ，国家发明专利：基于联合依赖概率建模的软件错误定位方法 ，已授权，201310099997.6

[5] 邱景, 苏小红, 马培军, 赵玲玲, 王甜甜. 国家发明专利： 一种新的使用逆向扩展控制流图的静态函数识别方法, 已授权，2013102919410

[6] 邱景, 苏小红, 马培军, 赵玲玲, 王甜甜. 国家发明专利：一种使用收缩执行依赖图识别库函数的方法, 已授权，2013105721740

[7] 王甜甜; 王克朝; 苏小红, 一种程序错误自动修正方法、装置、电子设备及存储介质，申请,2018.10.9，2018109506835

[8] 王甜甜; 王克朝; 苏小红, 一种基于程序不变量的软件故障定位方法及装置，申请,2018.11.9，201811096080X

缺陷检测和软件测试相关的软件著作权7项：

[1] 苏小红, C\C++代码编码规范审查系统, 2018R11S1045351

[2] 苏小红, 基于缺陷模式的C++缺陷检测工具, 2018R11L1008634

[3] 苏小红, 建筑全性能仿真平台内核功能正确性在线测试系统, 2018R11S1114736

[4] 苏小红, 隐式规则自动提取与反例检测系统, 2017SR123675

[5] 苏小红, C程序冗余代码缺陷检测工具,2011SR068570

[6] 苏小红, C语言重复代码检测系统,2010SR049470

[7] 苏小红, 克隆C程序代码引起的软件缺陷检测工具, 2010SR049161

（2）与国外大学联合培养博士研究生的工作基础

项目组与国外多所著名大学在程序分析领域联合培养博士生5人，如表1所示。

表1 与国外大学联合培养博士研究生情况

|  |  |  |
| --- | --- | --- |
| 姓名 | 大学 | 国别 |
| 逄龙 | 新加坡国立大学 | 新加坡 |
| 边奕心 | 美国马里兰大学巴尔的摩分校 | 美国 |
| 张凡龙 | 新加坡国立大学 | 新加坡 |
| 邱景 | 美国亚利桑那大学 | 美国 |
| 孙瑞 | 哈佛大学 | 美国 |
| 王俊杰 | 新加坡国立大学 | 新加坡 |

综上，申请人具有多年从事国家自然科学基金项目的研究工作经历，在与本项目相关的研究领域，具备扎实的理论基础和丰富的实践经验，并且取得了丰硕的研究成果，完全具备完成本项目的研究工作基础。

**（3）北京科技大学**课题组是一支由1名教授、1名副教授、1名博士后、3名博士研究生、8名硕士研究生组成的学术队伍。课题组成员已经有多年从事本课题相关的基础研究。课题组负责人孙昌爱教授主持或作为主要研究人员参与完成了多项软件测试领域的研究课题，包括：（1）主持并完成了国家自然科学基金项目“面向SOA软件的蜕变测试技术研究”；（2）主持并完成了北京市自然科学基金项目“蜕变测试应用于服务组装的若干关键问题研究”；（3）主持并完成了教育部博士点基金项目“Web服务组装中增量及声明式事务管理机制研究”；（4）主持并完成了教育部留学回国人员科研启动基金项目“基于输入域的软件失效模式识别及分布研究”；（5）主持并完成了北京市优秀人才培养资助项目“高效新型的SOA 软件测试技术与工具研究”；（6）主持并完成了中国科学院软件研究所计算机科学国家重点实验室开放课题“面向Web 服务及其组合的蜕变测试技术与框架研究”；（7）主持并完成了中央高校基本科研业务费资助课题“面向SOA的新型软件测试技术与工具”;（8）参与并完成了澳大利亚研究委员会资助的ARC Discovery 项目“Software Testing with Enhanced Partitioning Schemes”;（9）参与并完成了澳大利亚研究委员会资助的ARC Discovery 项目 “Fault-based Test Case Generation for Software”。

课题组在本课题相关领域进行了大量的前期研究工作与技术积累，包括：

1. **软件测试理论方面：**课题组长期从事软件测试方面的研究工作，在程序分析与静态测试[1-7]、回归测试[8-9]、变异测试[10-18]、模型驱动的测试[19-22]、适应性随机测试[23-24]、基于故障的软件测试[25-29]、蜕变测试[30-37]、动态随机测试[38]、面向场景的测试[39-40]、适应性分区测试[41]、遵从性测试[42]等方面提出了多种提高软件可靠性的测试理论与方法。特别地，提出的蜕变测试理论与方法为本课题在解决智能软件测试结果判定方面提供了理论与技术积累。**上述软件测试理论方面的研究工作，为本课题提供了理论基础与技术积累**。
2. **智能软件通常以服务的方式对外提供功能或参与复杂的系统集成。课题组在服务软件测试方面开展了系统的研究工作**，针对Web服务的可测试性差的特点，提出了在测试预期不存在的情形下Web服务的测试方法，开发了面向Web服务的蜕变测试框架与支持系统[30-34]，授权国家发明专利1项[32]，获中国计算机学会主办的服务创新大赛二等奖1项[37]；探索了面向Web服务的动态随机测试技术，提高Web服务的测试效率[38]；针对Web服务的规格说明与服务实现潜在的不一致问题，提出了一种基于约束的模型驱动的遵从性测试方法[42]；尝试将模型驱动的测试方法应用于BPEL服务组合程序，可以有效地服务组合程序中潜藏的故障；开发了面向BPEL服务组合程序的场景测试技术[39-40]，开发了面向BPEL 服务组合程序的变异测试系统[12]，获得中国计算机学会主办的全国软件原型竞赛二等奖1项[12]。在服务组合程序故障定位方面，课题组提出一种基于块的BPEL程序故障定位框架，在框架中合成了三种程序频谱的故障定位技术，采用BPEL程序实例验证了框架的可行性[43-44]；探索了谓词切换与切片分析相结合的故障定位技术[45]。上述服务软件测试方面的研究工作，为智能软件测试提供了重要的参考价值。

**参考文献（**\*通信作者**）**

1. **孙昌爱**, 刘超, 金茂忠. 一种有效的程序结构图的布图算法. 北京航空航天大学学报, 2000, 26(6):1305-1309.
2. **C. Sun\***, C. Liu, M. Jin, M. Zhang. Architecture framework for software test tool. *Proceedings of the 36th International Conference on Technology of Object -Oriented Language and Systems* (TOOLS 2000), IEEE Computer Society, 2000, pp.40-47.
3. **孙昌爱\***,金茂忠. 基于程序插装的动态测试技术实现. 小型微型计算机系统, 2001, 22(12): 1475-1479.
4. **孙昌爱\***, 金茂忠, 刘超, 靳若明. 程序执行时间的静态预估技术与可视化分析方法.软件学报, 2003, 14(1): 68-75.
5. **C**. **Sun\***, J. Zhou, J. Cao, M. Jin, C. Liu. ReArchJBs: a tool for automated software architecture recovery of JavaBeans-based applications. *Proceedings of the 16th Australian Conference on Software Engineering* (ASWEC 2005), IEEE Computer Society, 2005, pp.270-280.
6. **C. Sun\***. An incremental and practical approach to enable the component run-time evolution. *Proceedings of the 12th Asia-Pacific Software Engineering Conference* (APSEC 2005), IEEE Computer Society, 2005, pp.603-610.
7. **孙昌爱\***. 基于约束的软件失效域识别与特征分析. 软件学报,2012, 23(7): 1688-1701.
8. **C**. **Sun\***. A constraint-oriented test suite reduction method for conservative regression testing. *Journal of Software*, 2011, 6(2): 314-321.
9. **C. Sun\***, C. Fan, Z. Wang, H. Liu. dμReg: A Path-Aware Mutation Analysis Guided Approach to Regression Testing. *Proceedings of the 12th IEEE/ACM International Workshop on Automation of Software Testing* (AST 2017), *collocated with the 39th International Conference on Software Engineering* (ICSE 2017), IEEE Computer Society, 2017, pp.59-64.
10. **C**. **Sun\***, G. Wang, K. Y. Cai, T. Y. Chen. Distribution-aware mutation analysis. *Proceedings of the 9th IEEE International Workshop on Software Cybernetics* (IWSC 2012), *collocated with the 36th Annual IEEE International Computer Software and Applications Conference* (COMPSAC 2012), IEEE Computer Society, 2012, pp.170-175.
11. **孙昌爱\***, 王冠. MujavaX:一个支持非均匀分布的变异生成系统. 计算机研究与发展, 2014, 51(4): 874-881.
12. **孙昌爱**，潘琳，赵彦. μBPEL:一个面向BPEL服务组装程序的变异测试系统.全国软件原型竞赛二等奖，中国计算机学会软件工程专业委员会, 2015年11月.
13. **C. Sun\***, F. Xue, H. Liu, X. Zhang. A Path-aware Approach to Mutant Reduction in Mutation Testing. *Information and Software Technology*, Elsevier, 2017, 81(1):65-81.
14. **C. Sun**, L. Pan, Q. Wang, H. Liu, X. Zhang. An Empirical Study on Mutation Testing of WS-BPEL Programs. *The Computer Journal*, Oxford University Press, 2017, 60(1):143-158.
15. 孙昌爱,贾婧婷. 一种变异测试方法及装置. 中国发明专利, 专利号: ZL 201510700704.4，授权日期: 2017年11月.
16. **C. Sun**\*, J. Jia, H. Liu, X. Zhang. A Lightweight Program Dependence based Approach to Concurrent Mutation Analysis. *Proceedings of 42nd IEEE International Computer Conference on Computers, Software, and Applications* (COMPSAC 2018), IEEE Computer Society, 2018, Tokyo Japan, pp.116-125.
17. **孙昌爱**, 郭新玲, **张翔宇**, 陈宗岳. 一种基于数据流分析的冗余变异体识别方法. 计算机学报, 2019, 42(1):44-60.
18. **孙昌爱**\*, **王真**, 潘琳. 面向WS-BPEL程序的变异测试优化技术, 计算机研究与发展, 2019, 刊出中(DOI: 10.7544/issn1000-1239.2019.20180037).
19. 张楣, 刘超, **孙昌爱**.基于UML活动图的测试测试用例自动生成技术研究. 北京航空航天大学学报, 2001, 27(4): 433-437.
20. **C**. **Sun\***. A transformation-based approach to generating scenario-oriented test cases from UML activity diagrams for concurrent applications. *Proceedings of the 32nd Annual International Computer Software and Applications Conference* (COMPSAC 2008), IEEE Computer Society, 2008, pp.160-167.
21. **C**. **Sun\***, B. Zhang, J. Li. TSGen: A UML activity diagram-based test scenario generation tool. *Proceedings of the 2009 IEEE/IFIP International Symposium on Trusted Computing and Communications* (TrustCom 2009), IEEE Computer Society, 2009, pp.853-858.
22. **C. Sun\***, Y. Zhao, L. Pan, X. He, D. Towey. A Transformation-based Approach to Testing Concurrent Programs using UML Activity Diagrams. *Software: Practice and Experience*, 2016, 46(4):551–576.
23. 陈宗岳, 郭斐菁, **孙昌爱\*.** 失效区域紧致性对适应性随机测试的性能影响(英文). 软件学报, 2006, 17(12):2438-2449.
24. F.-C. Kuo,K. Y. Sim, C. Sun, S. F. Tang, Z. Q. Zhou.Enhanced random testing for programs with high dimensional input domains. *Proceedings of the 19th International Conference on Software Engineering and Knowledge Engineering*(SEKE 2007), 2007, pp.135-140.
25. **C**. **Sun\***, K. Y. Sim, T. H. Tse, T. Y. Chen. An empirical evaluation and analysis of the fault-detection capability of MUMCUT for general Boolean expressions. *Proceedings of the 2004 International Computer Symposium* (ICS 2004), Taiwan, 2004, pp.926-932.
26. **C**. **Sun\***, Y. W. Dong, K. Y. Sim, R. Lai, T. Y. Chen. Analyzing and extending MUMCUT for fault-based testing of general Boolean expressions. *Proceedings of the 6th IEEE International Conference on Computer and Information Technology* (CIT 2006), IEEE Computer Society, 2006, pp.184-189.
27. T.Y. Chen, M.F. Lau, K.Y. Sim, **C**. **Sun**. On detecting faults for Boolean expressions. *Software Quality Journal***,** Springer, 2009, 17(3): 245-261.
28. **孙昌爱\***, 程庆顺. 基于故障的布尔表达式测试技术综述. 计算机科学, 2013, 40(3):16-23, 40.
29. **C. Sun\***, Y. Zhai, H. Liu. Evaluating and Comparing Fault-based Testing Strategies for General Boolean Specifications: A Series of Experiments. *The Computer Journal*, Oxford University Press, 2015, 58 (5): 1199-1213.
30. **C**. **Sun\***, G. Wang, B. Mu,H. Liu, Z. Wang, T. Y. Chen. Metamorphic testing for web services: framework and case study. *Proceedings of the 9th IEEE International Conference on Web Services* (ICWS 2011), IEEE Computer Society, 2011, pp.283-290.
31. **C**. **Sun\***, G. Wang, B. Mu,H. Liu, Z. Wang, T. Y. Chen. A metamorphic relation-based approach to testing web services without oracles. *International Journal on Web Service Research,* 2012, 9(1): 51-73.
32. **孙昌爱**.一种无需预期的Web服务测试方法. 中国发明专利, 专利号: ZL201110109427.1, 授权日期: 2014年10月.
33. **C. Sun\***,Z. Wang, G. Wang. A Property-based Testing Framework for Encryption Programs. *Frontiers of Computer Science,* Springer, 2014, 8(3):478-489.
34. **C. Sun\*,** G. Wang, Q. Wen, D. Towey, T. Y. Chen. MT4WS: An Automated Metamorphic Testing System for Web Services. *International Journal of High Performance Computing and Networking*,IndersciencePress,2016, 9(1):104-115.
35. D. Towey\*, Y. Dong, **C. Sun**, T. Y. Chen. Metamorphic Testing as a Test Case Selection Strategy. *Science China: Information Sciences*, 2016, 59(5): 050108:1-050108:2.
36. **C. Sun\***, Y. Liu, Z. Wang, W. K. Chan. μMT: A Data Mutation Directed Metamorphic Relation Acquisition Methodology. *Proceedings of the First International Workshop on Metamorphic Testing* (MET 2016), *collocated with the 38th International Conference on Software Engineering* (ICSE 2016), IEEE Computer Society, 2016, pp.12-18.
37. **孙昌爱**，**付安**，张媛钰，张在兴. MT4WS:面向Web服务的蜕变测试系统，软件服务创新大赛二等奖，中国计算机学会服务计算专业委员会，2017年9月.
38. **C**. **Sun\***, G. Wang, K.Y. Cai, T. Y. Chen. Towards dynamic random testing for web services. *Proceedings of the 36th Annual IEEE International Computer Software and Applications Conference* (COMPSAC 2012), IEEE Computer Society, 2012, pp.164-169.
39. **C**. **Sun\***, Y. Shang, Y. Zhao, T. Y. Chen. Scenario-oriented testing for BPEL service compositions. *Proceedings of the 12th International Conference on Quality Software* (QSIC 2012), IEEE Computer Society, 2012, pp.171-174.
40. **C. Sun\***, Y. Zhao, L. Pan, H. Liu, T. Y. Chen. Automated Testing of WS-BPEL Service Compositions: A Scenario-Oriented Approach. *IEEE Transactions on Services Computing*, 2018, 11(4):616-629.
41. **C. Sun\***, H. Dai, H. Liu, T. Y. Chen, and K.-Y. Cai. Adaptive partition testing, *IEEE Transactions on Computers*, 2019, 68(2):157-169.
42. **C. Sun**\*, M. Li, J. Jia, J. Han. Constraint-based Model-driven Testing of Web Services for Conformance Improvement, *Proceedings of the 16th International Conference on Service Oriented Computing* (ICSOC 2018), Lecture Note in Computer Science (LNCS), Springer, 2018, pp. 543-559.
43. **C**. **Sun\***, Y. Zhai, Y. Shang, Z. Zhang. Toward effectively locating integration-level faults in BPEL programs. *Proceedings of the 12th International Conference on Quality Software* (QSIC 2012), IEEE Computer Society, 2012, pp.17-20.
44. **C. Sun\*,** Y. Zhai, Y. Shang, Z. Zhang**.** BPELDebuger: An effective BPEL-specific fault localization framework. *Information and Software Technology*, Elsevier, 2013, 55(12): 2140-2153.
45. **C. Sun\*,** Y. Ran, C. Zhen, H. Liu, D. Towey, X. Zhang. Fault localization for WS-BPEL program based on Predicate Switching and Program Sclicing. *Journal of Systems and Software*, Elsevier, 2018, 135:191-204.

## 2．**工作条件**（包括已具备的实验条件，尚缺少的实验条件和拟解决的途径，包括利用国家实验室、国家重点实验室和部门重点实验室等研究基地的计划与落实情况）；

南京大学计算机软件新技术国家重点实验室目前拥有各类高性能服务器300余台套,其中大型设备公共云计算平台一套（20余台多核高性能服务器、2台IBM Flex刀片服务器及13台GPU服务器作为计算资源，同时具备60T的高速数据存储能力和160T的备份存储、IBM虚拟存储网关SVC 2台、IBM Flash System 840闪存阵列1台）、大数据并行计算共享平台一套（140个节点，提供MapReduce、MPI、BSP等多种并行编程模型和环境）、EasyStack云平台一套（包括Dell 高性能计算服务器12台）、IBM刀片服务器2台（14片HS21、12片HS22、2片JS12）、IBM 3850 M2高性能服务器4台（1台96核、1台64核，2台32核）、TMS和华赛高速存储各1台、40余台GPU服务器。机房采用160KVA UPS三机系统并机运行，保证各设备24小时不间断工作。万兆接入校园网，同时支持IPv4/IPv6双栈协议，串接万兆安全网关一台，支持防火墙、IPS、WAF等多项安全功能。实验验证、仿真和测试环境非常好。此外，实验室还购买了大量软件资源并订阅了大量的电子刊物全文数据库，例如ACM、Blackwell、Elsevier、IEEE/IEE、Kluwer、Springer、Wiley等，方便研究人员查阅国际最新文献资料。

哈尔滨工业大学在相关领域拥有计算机信息安全国家重点实验室、网络环境下特定信息获取与处理技术国防科技创新团队等国家级科研基地及团队，还拥有黑龙江省企业智能计算与智能工程重点实验室等6个省部级科研基地，哈工大-IBM中国研究实验室、Google Android联合实验室等9个国际合作实验室。项目组所在团队科研支撑条件良好，目前建有一个高性能计算中心，可支撑大规模服务系统的研发与测试。申请者所在的哈尔滨工业大学计算机科学与技术学院智能软件技术研究中心，现配有设备如下：服务器3台，微机50余台，激光打印机5台。以上设备和条件可以充分保证开发工作顺利进行。

承担单位北京科技大学计算机与通信工程学院，拥有“计算机科学与技术”一级学科博士学位授予权，拥有“计算机科学与技术”、“软件工程”博士后流动站。近10年来，承担了国家科技攻关项目、国家“973”项目、国家“863”项目、国家自然科学基金课题及各类横向科研项目500余项，获得国家科技进步一、二等奖6项、部委特等奖及一等奖20余项，省部级及其它奖励200余项。在“211工程”和北京市重点学科等项目的支持下，购置了大批先进的仪器设备，现有设备原值近4000万元，实验、研究室使用总面积达到近4000平方米，软硬件环境达到国际先进水平。拥有“材料领域北京市重点实验室”、5个研究所和一个教学实验中心，配有相关的办公设施及计算机等实验设备。此外，北京科技大学已经购买了国际上主流计算机科学与技术、软件工程方面的电子数据库，订购了国内计算机领域主要的学术刊物，为本项目研究提供了便利的信息检索平台。除了计算机和网络升级与维护外，本课题所需实验条件已经基本就绪。

## 3．**正在承担的与本项目相关的科研项目情况**（申请人和项目组主要参与者正在承担的与本项目相关的科研项目情况，包括国家自然科学基金的项目和国家其他科技计划项目，要注明项目的名称和编号、经费来源、起止年月、与本项目的关系及负责的内容等）；

苏小红教授目前承担一项国家自然科学基金面上项目，名称“基于启发式选择变异和软件行为特征挖掘的软件错误定位方法”(编号61672191，起止年月2017.01-2020.12)。

与本项目的关系及负责的内容：本申请项目与已承担项目的研究目标和内容均不同，研究方法也有所区别。已承担项目的研究目标是针对人工定位错误不仅难度大而且极其耗时的问题，以及软件自动化调试的应用背景，通过研究启发式选择能快速有效定位软件错误的变异对象和变异算子，以达到提高错误定位效率和准确性并降低软件开发和维护成本的目的，为软件测试和软件自动化调试奠定理论基础。已承担项目的主要研究内容包括：（1）基于软件历史仓库挖掘的变异对象选择方法。（2）研究基于错误修正模式挖掘的变异算子选择方法。（3）研究变异算子的模糊多属性综合评价方法。（4）结合启发式选择变异和软件行为图挖掘的软件错误定位方法。该项目综合运用程序分析、变异测试、数据挖掘、机器学习和多属性决策等技术，研究软件错误自动定位方法。

孙昌爱教授目前承担一项国家自然科学基金面上项目，名称“面向服务组合程序的故障定位与修复技术研究”(编号61872039，起止年月2019年1月至2022年12月)。

与本项目的关系及负责的内容：该在研究项目主要研究服务组合程序的故障定位与修复问题，属于程序调试范畴；本项目研究智能软件的测试问题，主要参与者孙昌爱主要负责智能软件测试的判定问题，属于软件测试范畴。二者研究的对象与问题不同，但两个项目存在较好关联性，研究成果对于提高软件的质量均具有重要意义。

## 4．**完成国家自然科学基金项目情况**（对申请人负责的前一个已结题科学基金项目（项目名称及批准号）完成情况、后续研究进展及与本申请项目的关系加以详细说明。另附该已结题项目研究工作总结摘要（限500字）和相关成果的详细目录）。

我们一直在软件测试领域进行深入系统和细致的研究， 2016年结题的项目“软件测试的搜索方法（61272079）(2013～2016，项目负责人：聂长海)”是在已完成项目的基础上的深化、扩展和延伸，它们之间有很大的区别，同时也有密切的联系。在过去的这几个项目执行期间，我们作出了比较丰硕的研究成果。今年拟申请的项目“智能软件系统的数据驱动式测试方法与技术”是在已有基础上的稳步扩展和推进。

附1项目软件测试的搜索方法(61272079, 2013.1～2016.12)研究工作总结摘要

（已结题项目：软件测试的搜索方法（61272079），2013～2016，项目负责人：聂长海)

软件测试的过程就是利用各种方法搜索软件中潜藏的错误的过程，软件测试的演化搜索方法是利用各种启发式搜索技术解决软件测试中的问题，它将随机性与目标导向性进行有机结合，为软件测试提供了一种有效方法，成为传统软件测试方法的一个重要补充。本项目拟从以下方面进行深入系统研究：（1）各种搜索技术在测试用例生成及优化中的应用与比较；（2）研究在交互测试、结构测试、功能测试、性能测试等传统测试领域中应用基于搜索的软件测试技术，并进行比较；（3）研究基于搜索软件测试的性能改进，预期输出，质量评估和自动化等问题；（4）研究基于搜索的软件测试的模型、可测试性转化、故障检测能力和故障诊断方法等；（5）与其他方法的交叉结合、比较和在云计算、物联网等新技术新领域中的应用。在已有研究基础上,通过理论与工程实践的紧密结合,开拓创新,为搜索方法在软件测试中的应用和推广提供理论、方法、实证和工具支持。

已结题项目软件测试的搜索方法(61272079, 2013.1～2016.12) 相关成果的详细目录

期刊论文

1. 第一作者论文

(1) **Changhai Nie**(#)(\*)，Huayao Wu，Xintao Niu，Fei-Ching Kuo，Hareton Leung，Charles J. Colbourn，Combinatorial testing, random testing, and adaptive random testing for detecting interaction triggered failures，Information and Software Technology，2015.6.01，62：198-213，SCI，第一标注

(2) **聂长海**(#)(\*)，蒋静，覆盖表生成的可配置贪心算法优化，软件学报，2013.7.15，(07)：1469-1483，第一标注

2. 通讯作者论文（勿与第一作者论文重复）

(1) 曾梦凡(#)，陈思洋，张文茜，**聂长海**(\*)，利用蚁群算法生成覆盖表:探索与挖掘，软件学报，2016.1.01，(04)：855-878，EI，第一标注

(2) Wu, Huayao(#)，**Nie, Changhai**(\*)，Kuo, Fei-Ching，The optimal testing order in the presence of switching cost，Information and Software Technology，2016.12.01，80：57-72，SCI，第一标注

(3) Huayao Wu(\*)，**Changhai Nie**(\*)，Fei-Ching Kuo，Hareton Leung，Charles J. Colbourn，A Discrete Particle Swarm Optimization for Covering Array Generation，IEEE Transactions on Evolutionary Computation，2015.01.01，19(4)：575-591，SCI，第一标注

(4) 钮鑫涛，**聂长海**(\*)，CHAN Alvin，组合测试故障定位的关系树模型，计算机学报，2014.01.01，(12)：2505-2518，第一标注

会议论文

1. 第一作者论文

(1) **Nie, Changhai1**(#)(\*)，Jiang, Jing1，Wu, Huayao1，Leung, Hareton2，Colbourn, Charles J.3, 4，Empirically identifying the best greedy algorithm for covering array generation，ICSTW 2013，卢森堡，2013.3.18-2013.3.20，EI，第一标注

(2) **Nie, Changhai1**(#)(\*)，Chen, Siyang1，Leung, Hareton2，Cai, Kai-Yuan3，A case study of adaptive combinatorial testing，2013 IEEE 37th Annual Computer Software and Applications Conference Workshops, COMPSACW 2013，日本，2013.7.22-2013.7.26，EI，第一标注

(3) **Nie, Changhai1**(#)(\*)，Leung, Hareton2，Cai, Kai-Yuan3，Adaptive combinatorial testing，13th International Conference on Quality Software, QSIC 2013，中国，2013.7.29-2013.7.30，EI，第一标注

2. 通讯作者论文（勿与第一作者论文重复）

(1) Wu, Huayao，**Nie, Changhai**(\*)，Kuo, Fei-Ching，Test suite prioritization by switching cost，ICSTW 2014，美国，2014.3.31-2014.4.4，EI，第一标注

(2) Huayao Wu(\*)，**聂长海**(\*)，An overview of search based combinatorial testing，SBST，印度，2014.6.10-2014.6.15，EI，第一标注

(3) Li, Xiaohua，Dong, Zhao，Wu, Huayao，Nie, Changhai(\*)，Cai, Kai-Yuan，Refining a randomized post-optimization method for covering arrays，ICSTW 2014，美国，2014.3.31-2014.4.4，EI，第一标注

(4) Niu, Xintao1，**Nie, Changhai1**(\*)，Lei, Yu2，Chan, Alvin T.S.3，Identifying failure-inducing combinations using tuple relationship，ICSTW 2013，卢森堡，2013.3.18-2013.3.20，EI，第一标注

学术专著

(1) **聂长海**(#)(\*)，组合测试，科学出版社，386000，2015.4.1

(2) **聂长海**(#)(\*)，软件测试的概念与方法，北京: 清华大学出版社，339000，2013.05.01

授权专利

(1) 中国专利，吴化尧，**聂长海**，测试用例集执行顺序确定的方法及系统，授权，2016.01.13，CN201310532732.0

(2) 中国专利，**聂长海**，吴化尧，李晓花，董昭，刘克，蔡开元，覆盖表约简的方法及系统，授权，2016.05.12，CN201310477511.8

(3) 中国专利，**聂长海**，钮鑫涛，系统故障模式确定的方法及装置，授权，2016.05.11，CN201310572384.X

科研奖励

1. **聂长海**，质量需求驱动的软件演化管理和故障侦探方法，教育部，国家自然科学奖，省部二等奖，2015.02.01

人才培养

1. 出站博士后/毕业博士/毕业硕士/在站博士后/在读博士/在读硕士

(1) 陈思洋，毕业硕士，覆盖表生成蚁群算法：探索、挖掘与应用，聂长海，2012.09.01-2015.06.01

学术交流

1. 2015.06.13-2015.06.14，举办中国基于搜索的软件工程研讨会，南京大学，南京大学计算机系，聂长海

# （三）其他需要说明的问题

## 1. 申请人同年申请不同类型的国家自然科学基金项目情况（列明同年申请的其他项目的项目类型、项目名称信息，并说明与本项目之间的区别与联系）。

无

## 2. 具有高级专业技术职务（职称）的申请人或者主要参与者是否存在同年申请或者参与申请国家自然科学基金项目的单位不一致的情况；如存在上述情况，列明所涉及人员的姓名，申请或参与申请的其他项目的项目类型、项目名称、单位名称、上述人员在该项目中是申请人还是参与者，并说明单位不一致原因。

无

## 3. 具有高级专业技术职务（职称）的申请人或者主要参与者是否存在与正在承担的国家自然科学基金项目的单位不一致的情况；如存在上述情况，列明所涉及人员的姓名，正在承担项目的批准号、项目类型、项目名称、单位名称、起止年月，并说明单位不一致原因。

无

## 4. 其他。