# 智能软件系统的数据驱动式测试方法与技术

# 项目的立项依据

## 1.1研究意义

### 1.1.1智能化时代三驾“马车”

我们正处在一个智能化时代（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，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 拉动智能化时代的三架马车

世界各国都高度重视智能化发展进程，特别是英、美、日和欧盟等发达国家，都制定了相应的发展战略和行动计划。李克强总理在2015年两会的政府工作报告中，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、物联网等与现代制造业结合，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。习近平总书记在十九大报告中指出，要“推动互联网、大数据和人工智能与实体经济的深度融合”。从2015年发布《促进大数据发展行动纲要》，2016年发布《互联网+人工智能三年行动计划》，2017年发布《新一代人工智能发展规划》，2018年发布《新一代人工智能产业创新发展重点任务揭秘工作方案》，我国政府密集出台了一系列发展战略、行动计划和支持政策。

为营造工业互联网的发展环境，我国政府密集出台了一系列支持工业互联网的政策。2017年11月，国务院印发的《关于深化“互联网+先进制造业”发展工业互联网的指导意见》，工信部先后出台《工业互联网发展行动计划（2018—2020年）》《工业互联网专项工作组2018年工作计划》《工业互联网APP培育工程实施方案（2018—2020年）》等文件，支持工业互联网发展的政策体系、平台建设和应用路线图基本形成。

由软件定义与实现的云计算、大数据和人工智能正在悄然第改变着这个世界，同时也深刻地改变着软件工程领域，以下我们简单回顾一下这些改变。

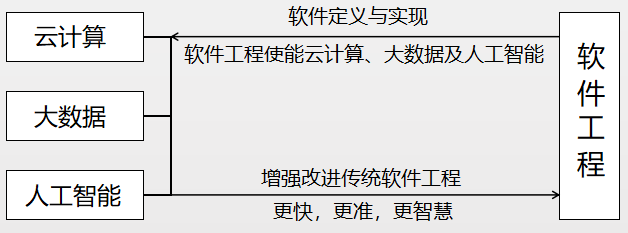


图2 软件工程与“三驾马车”

**云计算与软件工程**

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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、[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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)等方面。

社会需求推动了云计算的日益发展，云计算平台以及云计算环境下的软件可靠性和安全性越来越成为人们关注的焦点，并已经成为云计算发展和应用的瓶颈，这给传统的软件工程提出了一系列挑战，例如：（1）云计算环境的特殊性研究，研究云计算环境下资源的服务化、易接入性、可扩展性和可度量性等，研究这些特性给云计算环境下软件可靠性、安全性及软件测试带来的特殊性问题。（2）云计算环境下的软件可靠性理论与方法研究，研究开放的运行环境下软件运行剖面的变化规律及建模方法，云计算环境下软件可靠性建模理论等。（3）云计算环境下的软件安全性理论与方法研究，研究云计算环境下私密性、完整性、可追踪性和可用性等。

另一方面，云计算平台提供的几乎无限强大和灵活的计算能力，极大地推动了软件工程的理论和实践：例如，继基于组件的软件工程之后，面向服务的软件工程，人们提出在云平台上软件即服务（SAAS），一切皆服务（XAAS）的观念。并开始研究如何利用云平台基础设施所提供的资源来进行在线需求获取、软件构造、软件测试等的一般方法，并且研究如何将传统的过程、方法以及他们的基础设施向云端迁移。开源社区、众包空间和软件生态系统异常活跃。

**大数据与软件工程**

大数据是由软件定义实现的，同时大数据分析也可以有效增强软件工程实践，两者在现实世界中互相丰富和发展。

一方面，软件工程助力大数据软件系统开发。人们已经做了很多工作来研究如何通过软件工程来提高和改进大数据系统的能力，因为大数据系统给人们提出了很多挑战，例如，任何时候，任何需要，大数据系统需要能支持快速的、弹性的和跨多数据中心的扩展。因此，需要构建一个可灵活扩展，可自由合成的和无需人工干预的大数据系统是关键。

（1）大数据软件系统的系统架构

人们研究了大数据系统模块化和可扩展等需求，以及数据资源、数据存储和数据集成等数据流功能模块，提出了很多适合大数据系统的软件体系结构。

（2）大数据系统的测试和调试

研究大数据系统的测试标准，利用形式化方法对大数据系统的正确性进行验证。大数据系统运行过程中产生的日志等数据也具有大数据特性，这些数据也是我们对大数据系统进行测试和调试的重要依据。

（3）利用适当的软件过程开发大数据软件系统

人们已经开始研究利用CMMI和敏捷软件工程方法对大数据系统进行开发，提高开发效率和质量。

（4）大数据项目的管理

研究大数据项目开发的支持工具和技术，为大数据项目的需求、设计和实现等过程提供全程管理。

大数据软件质量保证需要采纳和扩展传统软件工程的质量保证技术，例如，测试大数据软件需要一些能够生成充分且有代表性测试用例的新方法，然而，由于数据规模巨大，穷尽测试不可能，因此需要采用形式化验证技术对大数据系统进行分析。数据源来自多个方面，并不是每种数据源都与大数据分析有关，且采集处理这些数据都需要相应的成本，大数据软件的成本效益需要在事前和事后进行评估。

大数据分析的需求给软件工程提出了很多挑战，例如为了处理大数据的大容量性、多样性、高速性、准确性等问题，不仅需要一些新的分析算法和工具，同时也需要持续发布及质量保证等非功能改进。

另一方面，大数据助力软件工程。人们一直研究如何利用大数据方法来改进和提升软件工程能力，具体包括以下几个方面：

（1）软件工程分析及其可视化

研究软件工程过程中的数据收集，整理和应用等问题。具体解决的问题包括：基于大数据的软件需求分析、结构、设计、编码、测试和维护等。

（2）软件社区数据挖掘

通过对一些大型组织或开源社区长期积累的数据进行挖掘，发掘一些对软件工程方法和技术有价值的认识。具体解决的问题如：源代码中哪些代码是新写的，哪些代码来自其他项目？从成百上千的项目观察中，我们能否跟开发者推荐最佳实践？

（3）自适应系统

通过对软件工程数据的应用，使得软件工程具有自适应性。具体解决的问题包括基于大数据的故障诊断、隐私保护、安全性和可靠性等问题。

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。基于这些数据，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。因此，数据对于现代软件开发的作用日益明显，并且至关重要。

数据驱动的软件工程运用机器学习、数据挖掘、信息可视化以及大规模数据处理等技术，旨在帮助软件从业者以数据驱动的方式进行软件的开发、运行和维护，有效处理、浏览和分析软件生命周期中生成的数据，从中提取有用的信息，做出正确决策。数据驱动的软件工程关注大数据分析技术在软件行业中的具体应用，是对现代软件工程方法的有效拓展，同时也与软件开发的实践密切相关。

**人工智能与软件工程**

“人工智能”这个词汇最早是出现在1956年，是在Dartmouth学会上美国的一个专家首先提出来的。从此以后，各个国家的相关研究者从中发展出了很多的人工智能理论和原理，人工智能的概念也逐渐的被人们所了解和认知了。简单来说，人工智能技术主要研究的目标是要能够使一台计算机或者一台机器可以完成一些通常需要我们人类亲自动手或动脑来完成的工作，也就是说只有依靠人类本身的智能才能完成的一些复杂工作。所以计算机本身就是人工智能的体现，人工智能技术的发展历史和计算机科学与技术的发展是密不可分的，人工智能技术也必须依靠计算机技术的发展才能逐渐发展成熟。

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。

从目前发展情况来看，**人工智能可以使得软件工程研究更加智能化与高效化**，具体包括以下几个方面：

（1）可以利用人工智能中专家决策系统以及人工神经网络系统来对软件工程项目进行更加智能化的设计。

（2）把智能化的模块组装到某个大型的软件系统中，从而使得这个软件可以更好的适应快速改变的需求。

（3）应用人工智能技术来提高某个计算机程序的用户界面的友好性，以及把人工智能技术应用到对图形用户的接口以及面向对象的程序设计当中等。

（4）通过利用人工智能方法对代码、各种评审记录和日志等资料进行分析，可以有效预防和发现各种软件故障，特别是利用强化学习方法生成高质量的测试用例，使软件工程的能力得到大幅度的提升。

虽然取得了很多的进展，但是计算机当中应用人工智能技术在未来很长的一段发展时间中还是有很多问题需要解决的，比如说，如何基于人工智能系统来建立支持环境和软件开发的人工智能机制。

另一方面，人工智能，特别是机器学习与深度学习的发展，要求强大的系统软件支撑，**软件工程可以为人工智能研究提供架构、测试的支持以及指导**，具体包括：

（1）软件工程可以为人工智能算法提供很好的实践验证平台。人工智能算法的有效性与可靠性，不能只停留在理论推导，需要进行一定的实证，而软件工程正是这样一个连接人工智能理论与实际应用相结合的桥梁，在实践中检验真理。

（2）软件工程可以提供面向人工智能的产品架构，指导人工智能产品开发、调试、测试、维护、管理等。例如，人工智能算法需要大量处理数据，这就需要数据库技术的发展，人工智能算法的可靠性验证则需要测试技术的革新。

（3）软件工程作为一门成熟学科，可以对人工智能的发展提供一定指导。人工智能作为新兴学科，其在正确性、可靠性、应用性上还有很多困难需要解决，而软件工程经过数十年发展，已发展成一门涉及[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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、[数据库](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93/103728" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、[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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)、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" \t "https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E5%B7%A5%E7%A8%8B/_blank)等方面的成熟学科，很多研究技术和经验对人工智能发展有很大的指导作用。

智能化已成为软件工程发展的新趋势，很多软件工程的难题可以尝试用人工智能方法去解决，智能化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。

**云计算环境下的基于人工智能和大数据的软件工程**

互联网和云计算的发展，软件已无处不在，计算机软件正在重新定义和实现着整个世界，同时，智能化时代的到来，软件系统的智能化日益成为人们关注的焦点，所以，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。

### 1.1.2智能软件系统的广泛应用

在当前高度信息化与网络化的时代，人类有限的认识和解决问题的能力已无法适应信息的飞速式变化与知识的爆炸式发展。智能软件系统是一类具有类人智能行为的计算机软件系统，通常基于人工智能实现，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，智能软件系统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。目前，在超级计算技术的辅助下，智能软件系统在特定应用领域展现出了强于人类大脑的问题解决能力：谷歌旗下DeepMind公司的AlphaGo围棋智能机器人[1][2]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[2]战胜了国际象棋特级大师加里·卡斯帕罗夫。在特定领域超越人类大脑的同时，智能软件系统在人类生活中的众多关键领域也得到了广泛应用：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，拍照软件依靠智能人脸识别及场景判断实现自动美颜等等。

与传统软件系统相比，智能软件系统呈现出深度学习、跨界融合、群体智慧的特点，具体来说：

（1）智能软件系统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，适应外部变化所带来的影响，而传统软件依据既定的业务逻辑执行各种操作，难以进行演化并适应外部环境变化；

（2）智能软件系统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；

（3）智能软件系统可依托互联网或大数据无缝整合多种智能，形成群体智慧，而传统软件系统不具备这样的特点。由于上述新特点，智能软件系统的测试面临诸多新的问题与挑战。

当前，以互联网和移动通信为纽带，人类群体、大数据、物联网已经实现了广泛和深度的互联，使得人类群体智能日益发挥越来越重要的作用。越来越多的来自世界各地的开发人员都在参与开发同一软件项目。智能系统开发方法从封闭和计划走向开放和竞争。

这些项目通常具有动态变化的软件需求，团队组织松散以及开发人员自由协作等特点。在协作过程中，许多与软件开发相关的信息，包括电子邮件系统，配置管理工具，缺陷跟踪系统等，都可以记录在软件仓库中。如果能够利用机器学习技术来针对这些数据进行分析，则有助于更好地理解测试行为，改进测试自动化。

### 1.1.3智能软件系统的安全隐患

智能软件系统为人类带来了极大的便利了，然而智能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，谷歌的无人驾驶汽车和一辆公共汽车相撞，原因是它希望公共汽车在一系列罕见的条件下停车，然而实际上公共汽车不可能停止[7]；流行的亚马逊Echo被认为是“聪明”的智能音箱之一，但一个德国人的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，他们给警察打了电话，警察不得不从前门破门而去，把音箱关掉；特斯拉的一辆无人驾驶汽车和一辆拖车相撞，原因是拖车的外表颜色和天空相近并且底盘较高[8]。类似的案例还有很多，这些都是智能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领域如无人驾驶领域带来灾难性影响，软件质量缺陷造成的损失往往是难以承受的。所以如何有效地保证智能软件系统的正确性以及安全可靠性已经成为智能软件应用、开发领域的一个亟待解决的问题。

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、对软件安全构成威胁的缺陷或不足。常见的软件安全漏洞包括缓冲区溢出、内存泄漏、整数溢出等。随着软件规模的不断增大，以及软件复杂度的不断提高，健壮的软件系统已经很难被设计和实现，所以软件漏洞的存在是不可避免的。因此，如何快速而准确地发现软件安全漏洞一直以来是信息安全领域的研究热点。

对于智能软件测试而言同样如此，

可见，软件安全漏洞检测对于保证软件系统安全具有十分重要的意义。而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。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，容易被恶意攻击者所利用。

当开发者接收到具体的缺陷报告后，需要去查找相应的缺陷源代码，一般情况下通常需要重现异常行为并执行代码审查才能找到原因。但是，缺陷报告的多样性和参差不齐的报告质量可能会使这个过程变得很困难。缺陷报告中经常缺少基本信息，例如缺陷报告中的自然语言陈述与软件系统中的技术术语之间的词汇不匹配问题，这会在一定程度上限制了基于简单词汇匹配分数的排名方法的准确性。

为了定位缺陷，开发人员不仅需要使用他们的领域知识分析缺陷报告，还需要从同行开发人员和用户收集信息。尤其当文件和报告的数量很大时，比如说由数百甚至数千个文件组成的大型项目，使用人工查找和分析缺陷的产生原因可能非常耗时。因此，如果能通过解析缺陷报告自动定位到缺陷源代码，则有助于提高软件维护的效率。

## 1.2国内外研究现状及发展动态分析

### 1.2.1软件测试理论的发展

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。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1.1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 1.1 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1.2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 1.2 软件测试方法体系**

### 1.2.2智能软件系统的测试研究

智能软件系统的特点是不确定性和概率性。对大数据具有依赖性、随机性的输入/输出、难以预测所有应用场景、需要从过去的行为中不断自我学习。采用的机器学习核心算法可能在种类和复杂性上有所不同，并且随着软件系统的应用可能组合更多的算法，具有动态变化的软件需求。通常被应用于其他更大的应用程序以满足最终的业务目标，因此需要第三方系统的接口。这些因素导致了智能软件系统开发与测试的复杂性，传统的软件开发和测试方法，将不能满足这些系统的需求。

**（1）智能软件系统的待测功能难以准确描述**：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，智能软件系统具有认知、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，并呈现出智能性；目前，研究者对于智能的认识有限，没有一个准确的定义[32]。这些问题导致软件测试人员难以清晰而准确的了解与描述其待测功能。例如，在无人驾驶汽车参加DARPA无人驾驶挑战赛[33]的场景下，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，而软件测试人员难以在这个场景下清楚了解无人驾驶系统所具有的具体功能并制定测试计划与任务。在此情况下，软件测试人员难以对智能软件系统实施有组织有目标的软件测试。

**（2）智能软件系统的模拟测试平台难以搭建：**智能软件系统通常与外界环境存在密切的交互，所以智能软件系统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系统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系统的模拟环境测试平台。如何让模拟测试平台接近智能软件系统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，搭建智能软件系统的模拟测试平台存在诸多困难。

**（3）智能软件系统的测试用例生成有待改进：**目前，智能软件系统的测试主要有基于场景的和基于功能的技术。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系统的实际表现。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系统的测试，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系统，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系统进行定性评价[34]，难以从微观层面对智能软件系统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系统按功能划分子模块，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[35][36]。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。因此，基于功能的测试技术所生成的测试用例仅针对智能软件系统中的某个特定模块，而无法应用于整个系统。此外，这种技术对不同模块的测试是独立进行的，缺乏对整个智能软件系统全面且综合的测试。综上所述，智能软件系统的测试用例生成有待改进。

**（4）智能软件系统的测试结果验证存在困难：**目前，智能软件系统测试预期的构造是一个耗时耗力的工作[32]，例如，在神经网络的测试中需要以人工的方式对测试数据集中的输入进行标注[37][40]。从黑盒测试的角度讲：对于输入输出类型较为复杂且业务逻辑功能多元化的系统，测试人员难以构造待测系统输入所对应的预期输出。从白盒测试的角度讲：传统软件系统的功能逻辑体现在系统的源代码中，可以通过程序分析的手段实现测试结果的验证；智能软件系统的功能逻辑没有直接体现在程序中，而是隐含在系统的人工智能里，如神经网络，导致白盒测试技术难以实际应用，无法实现测试结果的验证。此外，在不同的应用情景及任务目标下，智能软件系统对同一输入可能产生不同的输出，加剧了智能软件系统测试结果验证的困难性。

**（5）智能软件系统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及模型，以评估系统在测试中的表现。从智能软件系统的第一个特点中，不难发现，智能软件系统与传统软件系统最大的不同是前者所特有的人工智能，然而，目前尚不存在评价一个系统是否智能及智能程度的成熟评价指标及模型。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[38] [39]，但仍然没有合适的度量标准。此外，很多智能软件系统是多目标的，例如，无人驾驶系统会考虑驾乘舒适性、燃油消耗等。智能软件系统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[32]。在缺乏针对性质量评价指标及模型的情况下，难以对智能软件系统的质量进行系统有效的评估。

### 1.2.3基于安全缺陷报告的软件安全漏洞识别及定位发展

关于安全缺陷报告识别，近些年国外开始有一些研究，而国内尚无相关的研究。目前关于安全缺陷报告识别主要集中在以下两个问题的研究：

（1）文本挖掘方面的研究

安全缺陷报告识别主要是利用缺陷报告中的自然语言信息，如何有效地挖掘缺陷报告中的文本信息是一个关键的研究问题。Gegick[8]根据缺陷报告中的自然语言描述信息，利用词袋模型将所有缺陷报告表示成一个词项-文档矩阵，并将该矩阵和相应的标签作为输入来训练统计模型，从而进行安全缺陷报告识别。Behl[9]同样利用词袋模型表示缺陷报告，并且引入了TF-IDF值作为词的权重并通过朴素贝叶斯模型来进行安全缺陷报告识别。Chawla[10]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。Zou[11]充分利用缺陷报告的文本信息和非文本信息训练模型，从而进行安全缺陷报告识别。

（2）类别不平衡方面的研究

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别不平衡的问题，即安全缺陷报告的数量要远远小于非安全缺陷报告的数量，这会使得预测模型无法实现预期的效果，因此一些研究在类别不平衡处理方面进行了探索。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）对训练集进行处理，再利用经过处理的训练集训练分类器来识别具有较大影响的缺陷报告。Zhou[13]利用NLP和机器学习技术来识别缺陷报告和提交信息中的安全问题，他们提出了K折叠加算法来集成多个个体分类器，从而缓解数据集中类别不平衡的影响。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，非监督学习克服了监督学习需要人工标注数据以及可能出现类别不平衡问题的缺点。

Peters[15]发现一些非安全缺陷报告同样包含和安全相关的关键词，这些非安全缺陷报告相当于对模型的训练引入了噪音，势必会影响模型的训练效果，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别不平衡的情况下又被扩大。因此他们提出了一个框架FARSEC用来进行安全缺陷报告识别。该框架在训练预测模型前将含有安全关键词的非安全缺陷报告从训练集中移除，从而提高模型对安全缺陷报告的识别效果。该方法从安全缺陷报告中提取TF-IDF值最高100个词作为安全相关的关键词，并利用这100个安全关键词来过滤非安全缺陷报告，同时还利用这些安全关键词将每一个缺陷报告表示成一个100维的特征向量作为模型的输入。然而，这种方法存在一定的问题。首先，这些TF-IDF值较高的词未必就是和安全相关的词，如果这些安全关键词不准确，那么对非安全缺陷报告的过滤效果就会大打折扣。其次，用提取的100个安全关键词来表示缺陷报告，会造成缺陷报告的向量表示具有很大的稀疏性，这是因为在一份缺陷报告中很可能只出现少数几个安全相关的关键词。

关于安全缺陷的定位问题主要集中在以下两个问题的研究：

（1）基于信息检索技术的故障定位。Gay等人，采用信息检索技术，根据缺陷报告自动搜索相关文件[16]。他们将初始缺陷报告视为查询，并根据与查询的相关性对源代码文件进行排名。

Ye X等人[17]结合了4种相关计算方法对可能存在缺陷的源代码文件进行排名，这四种计算方法包括1）通过简单的空间向量模型来计算源代码和缺陷报告的相似性。2）利用API描述来弥补缺陷报告和源代码之间的词汇差距。3）利用先前已解决的缺陷报告的创建时间作为辅助计算。4）采用协同过滤的方法，通过检查源代码文件之前所涉及的所有缺陷报告并将其提取摘要，然后和新的缺陷报告做余弦相似度计算。

2016年Ye X等人[18]又对之前的方法提出了改进，扩充了另外两个对源代码排名有参考价值的方法：1）将源代码文件分解为类库，方法库，变量库，评论库，然后分别与新缺陷报告的摘要和描述做相似度计算。2）通过源代码文件的依赖关系建立文件网络图，然后用类似PageRank的方法计算新缺陷报告与源文件的相似度。

（2）基于深度学习的故障定位。An Ngoc Lam等人[19]使用深度学习进行故障定位。他们使用改进过的向量空间模型（VSM，Vector Space Model）计算缺陷报告和源文件之间的相似度，并用深度神经网络（DNN，Deep Neural Networks）来学习缺陷报告的具体术语和源文件文本标记之间的关联，最后将两者相结合，给出源代码文件和缺陷报告的相似度排名。

## 1.3智能软件的数据驱动式测试存在的问题

尽管智能软件系统为人类社会带来了极大的便利，但是依旧有很多智能软件系统缺陷造成了安全隐患的案例。尽管软件测试技术和理论已经非常丰富，但是针对智能软件系统的测试研究还远远不够，传统的软件测试技术和方法还是主要针对传统软件。虽然大数据时代已经来临，但是从海量数据中挖掘出有价值的信息，甚至利用数据来驱动测试的技术还远远不够完善，故而面向智能软件系统的数据驱动式测试还面临着以下问题：

**（1）软件测试方法与技术的整合归纳问题。**软件测试理论诞生至今，形形色色的测试方法与技术相继被提出，然而每一种方法与技术的特点及适应场景仍缺乏系统总结与归纳，甚至于若干种测试方法与技术能否互补以适应新的测试场景的研究还非常缺乏。

**（2）面向智能软件系统的测试技术与方法问题。**智能软件出现时间较短，故而面向智能软件系统的测试研究还远远不足，智能软件系统的特点是什么样的？与传统软件软件的差异在哪边？传统的测试方法与技术有哪些可以借鉴或参考的？现有的智能软件测试方法是怎样的？是否需要重新提出一些新的测试技术与方法？

**（3）数据驱动与智能软件测试的结合问题。**智能软件与大数据密不可分，数据是智能的基础，更是智能的体现，我们的测试必须要充分利用大数据，挖掘数据中有价值的信息，怎么通过数据驱动来提升智能软件测试效率，建立数据驱动式的智能软件测试框架，还需要深入加强研究。

**（4）利用云平台革新智能软件测试技术问题。**云平台可以为智能化软件提供算力支持，那么我们也可以利用云平台的高并发性的特点，为智能软件测试提供高效的计算存储资源。云计算环境下，软件技术、架构将发生显著变化，软件测试工具也应工作于云平台之上，而不再是传统的本地方式;软件测试的环境也可移植到云平台上，通过云构建测试环境，测试技术本身也应可以通过云实现协同、知识共享、测试复用。

**（5）测试方法与技术能否实现智能化问题。**我们的测试对象是智能软件系统，我们也希望测试方法与技术也能向智能化发展。产品不是一成不变的，智能化产品更新换代非常快，智能化的测试方法与技术可以动态根据产品的更新与发展智能的改进和优化测试策略和测试方案，而不至于完全不适用，所以我们需要考虑和研究如何使得智能化的测试方法与技术需要有一定的学习能力，实现智能化等问题。

## 1.4主要参考文献

# 项目的研究内容、研究目标，以及拟解决的关键科学问题

## 2.1研究内容

在智能软件系统、大数据研究与软件测试等领域的前沿研究成果的基础上，围绕智能软件系统的功能需求描述、质量评价指标确立与质量模型构建、仿真测试平台搭建、测试用例生成、测试结果验证的关键问题展开研究，主要研究内容如下：

### 2.1.1软件测试方法与技术

为了有效检测软件中潜在的错误，人们在软件开发的不同阶段，从不同角度，基于不同的理论，针对不同软件类型特性及开发方法提出了很多软件测试方法，为有效度量和提高软件质量提供了重要保证。软件测试方法不仅数量众多，而且方法之间既有明显的差异，又有密切的联系，需要进行深入的分析和研究。一方面，软件测试方法间系统的比较，分清边界，有利于科学地应用；另一方面，将各种方法进行适当的组合，有可能产生意料之外的效果，甚至可以产生新的测试方法。因此，本项目拟在几个关键问题和具体实例研究的基础上，对已有的软件测试方法进行系统地分析比较，并在此基础上研究各种方法之间可能的组合作用，从而为软件测试方法以及各种方法的组合应用提供基础、实证和示范，并进一步丰富和发展软件测试理论、方法和实践。

统一普适的软件测试理论一直是软件测试领域研究的一个热点内容，围绕这个问题，人们从不同角度提出很多软件测试的理论和方法。我们将现有各种方法分成了四大类(见表2)：测试方法类 Mi（1≤i≤40）共有 40 种； 属性或方面类测试 Aj（1≤j≤36）共有 36 种；阶段类测试 Sk（1≤k≤8）共有 8 种；软件类型测试 SUTl（1≤l≤16）共有 16 种。已有的理论和方法是我们需要深入研究的对象和进一步研究的基础。

表2软件测试方法分类表



### 2.1.2智能软件系统及其测试

在智能化软件系统与软件测试等领域的前沿研究成果的基础上，围绕智能软件系统的功能需求描述、质量评价指标确立与质量模型构建、仿真测试平台搭建、测试用例生成、测试结果验证的关键问题展开研究，主要研究内容如下：

**面向智能软件系统的待测功能描述方法**

针对智能软件待测功能难以准确描述的问题，研究面向智能软件系统的待测功能描述方法。针对具有认知、学习、推理能力的智能软件系统，能够准确且清晰地描述系统中不断变化与衍进的功能。研究如下问题：

（1）智能软件系统的认知、学习、推理机制：造成智能软件系统中功能发生变化的原因是其特有的认知、学习、推理能力，因此，了解这种能力的机理及其对智能软件系统功能产生的影响是准确而清晰地描述系统中不断变化功能的前提。研究智能软件系统的认知、学习、推理机制及其对系统功能所产生的影响。

（2）智能软件系统中功能的衍进机制：在智能软件系统的认知、学习、推理机制及其对系统功能产生的影响的基础上，研究智能软件系统中功能的衍进机制，归纳智能软件系统的功能可能发生的变化及其特点。

（3）持续衍进功能的形式化描述理论与方法：在归纳智能软件系统功能的潜在变化及其特点后，研究智能软件系统中持续衍进功能的形式化描述理论与方法，为准确且清晰地描述智能软件系统中持续衍进功能提供理论支撑。

**智能软件系统的模拟测试平台**

针对真实环境下的智能软件系统测试存在高昂且难以负担的成本的问题，搭建智能软件系统的模拟测试平台。通过模拟智能软件系统与外界环境的交互，实现在接近真实的环境下对智能软件系统展开测试。需要研究以下内容：

（1）数据驱动的智能软件系统应用环境建模：针对不同类型的智能软件系统，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系统在各应用场景下的模型。

（2）智能软件系统应用环境模拟的实现机理：以智能软件系统的应用场景模型为指导，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系统的实际应用环境，为智能软件系统的测试奠定基础。

**数据驱动的智能软件系统的测试用例生成与优化**

针对现有智能软件系统测试用例生成技术的诸多局限性，考虑智能软件系统的特点，整合面向场景的测试用例生成技术与面向功能的测试用例生成技术的优势，研究新型数据驱动的智能软件系统的测试用例生成技术。在此基础上，研究测试用例的优化技术，包括：测试用例选择，测试用例优先级排序，测试用例集充分性评估（各种覆盖准则）等。需要研究以下内容：

（1）数据驱动的智能软件系统测试用例生成实现机理：将基于场景的和基于功能的测试用例生成技术相结合，弥补两者的不足之处，提出数据驱动的智能软件系统测试用例生成技术。该技术旨在将场景测试数据与系统功能测试数据相结合，实现智能软件系统自顶向下的测试。其关键在于研究一种将测试场景与系统功能相关联的技术，从而实现由场景到系统功能的融会贯通。

（2）基于组合优化的智能软件系统测试用例选择实现机理：在单一场景下，通常将不同参数取值进行排列组合以检测系统在不同情形下的表现，然而这将带来组合爆炸问题。研究通过组合优化的方式在测试用例中进行取样的同时保证覆盖到尽可能多的组合，减少智能软件系统的测试开销。

（3）基于距离的智能软件系统测试用例优先级排序实现机理：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。为此，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最不同的测试用例，并以此为基础提出基于距离的智能软件系统测试用例优先级排序技术。

（4）基于覆盖准则的智能软件系统测试用例集充分性评估技术：如何评估测试用例集对智能软件系统测试的充分程度是一个重要的问题。研究基于变异覆盖准则、神经网络覆盖准则等多种不同准则的测试用例充分性评估技术，以满足测试用例集充分性评估的需求。

**基于蜕变测试的智能软件系统的测试结果验证技术**

针对智能软件系统的测试结果验证存在困难的问题，研究基于蜕变测试的智能软件系统测试结果验证技术。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。这些必要的属性被称为蜕变关系，隐含于待测软件的功能规格说明中，是验证测试结果及判断待测软件是否满足特定的功能需求。研究基于蜕变测试的智能软件系统测试结果验证技术，需要研究如下内容：

（1）面向智能软件系统的蜕变关系识别方法：智能软件系统中隐含的蜕变关系是验证测试结果的关键。如何从智能软件系统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系统蜕变关系识别方法。

（2）基于蜕变测试的智能软件系统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，实现智能软件系统测试结果的验证，判断待测软件是否满足功能需求。

**面向智能软件系统的质量评价指标及模型**

针对智能软件系统的质量评价指标及模型存在缺失的问题，研究面向智能软件系统的质量评价指标及模型。针对智能化软件系统呈现出的独有特点，提出新型系统智能质量评价指标及模型。研究面向智能软件系统的质量评价指标及模型，需要研究以下内容：

（1）面向系统智能的质量评价指标及模型：重点分析智能软件系统智慧程度的影响因素，提炼用于描述软件系统智能程度的软件质量特征，提出评估软件系统智能成熟度的质量评价指标，并在传统软件质量模型的基础上构建新型面向系统智能的质量评估模型。

（2）多目标情景下的智能软件系统质量评估指标及模型：分析多目标情景下智能软件系统满足用户需求程度的描述性软件质量特征，提出多目标情景下的智能软件系统质量评估指标，并以传统软件质量模型为参考提出新型多目标情景下的软件质量评估模型。

### 2.1.3大数据时代下的数据驱动式测试

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。

首先，我们学会筛选数据，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，我们需要研究如何高效预处理，使得多样的原始数据处理成可利用的中间数据。

其次，我们需要利用数据挖掘方法，研究数据中蕴含的重要信息对测试的帮助，直观而言，这些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。

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，从而达到对智能软件系统的高效测试。

### 2.1.4云计算平台下的测试技术革新

云计算是通过使计算分布在大量的[分布式计算](https://baike.baidu.com/item/%E5%88%86%E5%B8%83%E5%BC%8F%E8%AE%A1%E7%AE%97" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)机上，而非本地计算机或远程服务器中，企业[数据中心](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E4%B8%AD%E5%BF%83" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)的运行将与[互联网](https://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)更相似。这使得企业能够将资源切换到需要的应用上，根据需求访问[计算机](https://baike.baidu.com/item/%E8%AE%A1%E7%AE%97%E6%9C%BA" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)和[存储系统](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E7%B3%BB%E7%BB%9F" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)，这样可以提升计算的可靠性、通用性、扩展性等。

云计算环境下，对软件开发产生重要影响，首先软件技术与架构发生显著变化，需要与云相适应。其次，软件开发的环境和工作模式也将发生变化，虽然，传统的软件工程理论不会发生根本性的变革，但基于云平台的开发工具、开发环境、开发平台将为敏捷开发、项目组内协同、异地开发等带来便利。最后，软件产品的最终表现形式更为丰富多样，从可下载的应用到可调用的服务等。

云计算环境下，由于软件开发工作的变化，也必然对软件测试带来影响和变化，故而需要对测试技术进行革新。

（1）软件技术、架构发生变化，要求软件测试的关注点也应做出相对应的调整。软件测试在关注传统的软件质量的同时，还应该关注云计算环境所提出的新的质量要求，如软件动态适应能力、大量用户支持能力、安全性、多平台兼容性等。

（2）云计算环境下，软件开发工具、环境、工作模式发生了转变，也就要求软件测试的工具、环境、工作模式也应发生相应的转变。软件测试工具也应工作于云平台之上，测试工具的使用也应可通过云平台来进行，而不再是传统的本地方式;软件测试的环境也可移植到云平台上，通过云构建测试环境;软件测试也应该可以通过云实现协同、知识共享、测试复用。

（3）软件产品表现形式的变化，要求软件测试可以对不同形式的产品进行测试，如Web Services的测试，互联网应用的测试，移动智能终端内软件的[测试](https://baike.baidu.com/item/%E6%B5%8B%E8%AF%95" \t "https://baike.baidu.com/item/%E4%BA%91%E8%AE%A1%E7%AE%97/_blank)等。

### 2.1.5测试方法与技术的智能化

我们的测试对象是智能化软件系统，但是同样的，我们也需要开发的测试方法与技术尽可能的智能化，主要因为智能化软件版本应用广泛且版本更新较快，我们开发的测试方法与技术不可能只针对某一种软件或某一版本的软件可以高效测试，而应该具有一定的学习能力，根据待测软件的不同，合理建立测试模型，动态调整测试策略。

这就要求我们前期需要广泛调研多样的智能软件系统，制定动态科学的测试框架，测试技术与方法也需要具备一定的普适性和可扩展性。从而实现测试方法与技术的智能化。

## 2.2研究目标

### 2.2.1总结现有的软件测试理论，归纳适合本课题的方法与技术

我们拟对现有的软件测试方法和理论进行调查研究，例如组合测试[8,43]、蜕变测试[19]、变异测试[22]、演化测试[9，11]、FUZZ测试等，并进行科学性总结与归纳，分析总结每种测试方法和理论的应用场景与各自特点。建立一套完整的软件测试方法体系。本课题是面向智能软件系统的数据驱动式测试，传统的软件测试方法和理论可能都不能很好适用，但可以尝试从别的角度切入，考虑传统软件测试方法和理论能否通过一定修改以适合本课题，或者考虑是否存在某几类测试方法与技术可以借鉴或综合，确定智能软件系统的测试方向。

### 2.2.2建立面向智能软件系统的普适的测试框架与体系

我们拟对现有智能软件系统进行调查研究，智能软件非常庞大，种类众多，我们不可能开发出针对每一款智能软件的测试工具，我们希望能够分析智能软件系统的特点，对智能软件系统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，继而提出一个面向智能软件系统的普适的测试框架与体系，并给出科学性论述。

### 2.2.3探索数据驱动式测试中关键方法与技术

在建立面向智能软件系统的普适的测试框架与体系之后，我们拟在框架中丰富框架中关键方法与技术，我们拟归纳出数据驱动式测试的特点，对数据进行预处理，挖掘等操作。我们希望能对智能软件系统中大量多样的数据，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、社交媒体反馈等等，进行数据预处理，数据挖掘，数据存储，数据学习等操作，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，从而达到对智能软件系统的高效测试。

### 2.2.4基于云计算理论建立高效的测试框架与体系

云计算可以提高测试的高效性与可靠性，传统测试存在测试复杂度高，测试成本巨大等问题，我们需要利用云平台计算能力显著的特点，开发出并行测试框架，其次云计算与测试相结合要求我们关注云计算环境所提出的新的质量要求，如软件动态适应能力、大量用户支持能力、安全性、多平台兼容性。测试工具也应运行在云平台之上，实现共享性与安全性，测试对象也应包括单机工具、线上工具和Web服务。

### 2.2.5基于人工智能开发智能化的测试方法与技术

测试框架与体系不是一成不变的，我们拟开发出可以根据测试对象的不同，学习不同测试场景，进而动态调整测试策略和测试评价的智能化测试方法与技术。智能化与软件工程相结合要求我们开发出智能化的测试软件，智能化意味着对于广泛意义的不同种类的智能软件，都具有一定的学习能力和调整能力。

## 2.3拟解决的关键科学问题

### 2.3.1软件测试理论的系统性总结归纳问题

软件测试理论经过数十年的发展，越来越多的测试方法与技术被提出、发展并应用。不同的测试方法有其特殊的不可替代的优点，如何通过系统的比较获得各种方法的边界是一个需要解决的重要的问题，我们需要系统梳理出软件测试理论中不同方法的适用场景、使用条件、预期结果等信息，探索不同软件测试的方法的互补性与互异性，所以我们首先要解决对传统软件测试理论的现有技术与方法系统性总结归纳的问题。

### 2.3.2面向智能软件系统的测试理论研究及应用问题

总结与归纳现有软件测试理论中测试技术与方法之后，我们需要解决通过分析智能软件系统的特点，如何借鉴并改进传统软件测试理论与方法，进而如何构建面向智能软件系统的数据驱动式测试框架，丰富测试技术与方法等问题，并能设计实验选取合理测试对象来进行实证。解决好这一系列问题之后，我们就可以通过逐步完善测试框架，来提升我们对智能软件系统的数据驱动式测试能力，并且通过实证研究与案例分析来验证测试理论的科学性。

### 2.3.3数据驱动式测试中数据处理问题

智能软件系统会产生大量的数据，如何采集、预处理、存储、挖掘、学习这些数据是一个重要难题。我们知道大数据具有数据量大、多样、高速且价值密度低等特征，在数据采集、整理、存储上需要应用云计算及并行计算技术提高数据预处理能力，而为了提取原始数据中的潜在价值，对故障进行诊断定位，提取出关于软件质量和开发动态的重要信息等工作则需要利用机器学习、数据挖掘、信息可视化以及大规模数据处理等技术。

### 2.3.4云计算环境下测试技术革新问题

测试是保证软件质量的重要一环，而测试效率是评估测试方法及技术优劣性的重要指标，传统的数据处理技术对于大数据的处理效率很低，得益于云计算技术的蓬勃发展，人类面对复杂问题的计算能力有了很大的提升，所以本项目中希望应用云计算技术，来提升智能软件系统的测试技术，提高测试效率，即测试技术革新问题，也是云计算与软件测试的结合问题。其次，数据安全及测试安全也是我们不得不考虑的问题，云平台可以搭建一个面向智能软件系统的可靠高质量的测试平台，我们需要考虑如何应用云平台高可靠性的特征来保证测试安全可靠的问题。

### 2.3.5测试方法与技术智能化问题

人工智能与软件工程的结合不仅仅停留在面向智能软件的测试方法与技术研究，还应该在软件工程活动中利用人工智能算法改进软件活动效率。我们考虑如何使得测试方法与技术智能化的问题，即如何学会智能化分析测试需求，智能化制定测试策略，智能化生成测试用例，智能化诊断测试报告以及智能化定位并修复故障灯问题。

# 拟采取的研究方案及可行性分析

### 1 广泛调研，系统梳理

我们已经在编写的教材《软件测试的概念与方法》中为每种方法整理了一  
些信息，将在此基础上，针对每种方法继续广泛收集资料，充分了解相关工作，  
认真总结出六个方面的要素，即概念、理论、方法、研究现状，优缺点和文献  
资料。并给出相应方法的支持工具和实例研究。

其次，我们也会调研目前在智能软件领域的测试以及数据驱动性测试的研究成果，梳理每项成果的优势与不足，加深对问题的理解。

### 2 全面思考，重点突破

由于传统测试方法与技术众多，适用场景及优缺点各异，我们考虑智能软件系统的数据驱动式测试方法与技术，就要充分借鉴已有的测试方法与技术，首先明确测试需求，之后尝试确定测试模型，从模型入手，寻找类似或可能适用的测试方法与技术。如果存在几个可能适用的测试方法与技术，则需要我们重点分析比较，集众家之所长，组建成一个完整的测试方法。如果技术之间存在排斥性，我们也可以保留数个测试方法，独立进行后续测试，在实证中比较诸多方法的优劣，从而进一步加深对智能软件测试问题的认识。

### 3 技术整合，综合应用

本项目是数据驱动式测试方法与技术研究，大数据也是智能软件的重要特征，科学处理数据是测试方法研究的关键，数据的处理应用包括数据采集、数据预处理、数据存储、数据挖掘、数据学习等诸多阶段，其中涉及到很多机器学习和并行计算等技术，这些算法或技术目前有迹可循，在人工智能领域和云计算领域应用广泛，我们项目需要将这些诸多技术整合在测试框架中，应用于数据驱动测试方法中。

### 4 技术革新，应用升级

智能软件与大数据密不可分，那么处理大数据显然会带来传统的数据处理方式效率低与数据安全隐患高等问题，所以我们可以利用云计算技术实现数据处理技术的革新，例如我们需要系统性研究MapReduce技术，从而应用在大规模数据的并行处理上，可以极大提升数据处理与计算能力。其次，应用Hadoop技术搭建高效可靠的分布式测试平台，可以在技术上消除软件系统及测试工具的安全隐患。

### 5 实证研究，智能测试

智能化测试是软件测试研究未来的趋势，尽可能实现智能化也是项目提出的测试方法与技术追逐的目标，在已提出的智能软件系统的数据驱动性测试方法与技术的基础上，充分考虑测试对象的多样性，归纳出普适的测试理论与方法，开发高兼容性的测试工具，并在实证中检验测试理论的科学性，坚持在智能化方向上继续努力。

# 本项目的特色与创新之处

1. 我们在前期已经做了非常充分的调研，收集整理了迄今为止，几乎所有的  
   软件测试方法和相关理论，并对这些理论和方法进行初步的分类和处理。以此为基础，期望在软件测试的理论和方面作一些新的创新。
2. 设计面向智能软件系统的测试框架是本项目的特色之一，通过比较传统软件与智能软件的共同点与差异，进一步归纳出智能软件测试的特征，丰富智能软件系统的测试理论。
3. 针对智能软件与大数据密不可分的关系，项目另一个重点在于数据驱动式测试的研究，科学利用数据，处理数据进而从数据中挖掘出软件质量和开发动态相关的信息从而服务软件测试，是本项目的一个重要创新。
4. 数据的高效处理是项目难点，我们利用云计算技术，提升数据处理整合效率，其次，从安全角度考虑，利用云平台搭建安全可靠的测试平台，为测试理论与技术的实践作准备，这些都是本项目的重要特色。
5. 测试智能软件的同时，我们还需要保证测试方法与技术本身的智能化问题，智能化是未来软件发展的趋势，我们会整理应用人工智能方法，保证测试方法与技术的生命力与竞争力，尽可能在测试方法与技术智能化领域作进一步探索与尝试。

# 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）