智能化软件系统的
数据驱动式测试方法与技术

参照以下提纲撰写，要求内容翔实、清晰，层次分明，标题突出。**请勿删除或改动下述提纲标题及括号中的文字。**

（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）：

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。智能化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，智能化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，智能化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，智能化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。广泛应用的智能化软件系统为人类带来了极大的便利，然而，其质量仍然存在诸多问题，如谷歌无人驾驶系统出现严重故障并造成人员伤亡[4]。特别是在安全有关领域，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。**

软件测试是一种重要的软件质量保障手段，是软件生命周期不可或缺的一

批注 [付1]: 补充国家相关政策的资料

还有其他国家在这个方向的进展

个环节，一直是软件工程领域研究的热点与难点问题之一。在软件测试中，测试人员使用有限的测试用例执行待测软件，并将测试用例的输出与其预期输出进行比较。若两者不一致，则说明待测软件中潜藏的故障（存在缺陷）。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。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**[5]：

- **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[6]。符号执行技术由 King 等人[7]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[8]、STP[9]、choco[10]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[11]、动态符号执行[12]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[13]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[15]通过在符号执行的约束表达式中引入符号变量在执行中应满足条件的描述，提升了白盒测试中用例生成的有效性。
- **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[15]。基于有限状态机的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，形成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[5][16]。基于 UML 模型的方法使用 UML 中的建模元素为待测程序建立测试模型以支持测试用例生成，例如，Kansomkeat 和 Rivepiboon [17]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。
- **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 [18]及 in parameter order 算法[19][20]. 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[21]提出的基于遗传算法的技术及 Bryce 和 Colbourn[22]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[23][24]。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[25][26][27].

- **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，以此让测试用例在软件输入域上更加均匀地分布。Chen 等人[28]在此想法的基础上提出了适应性随机测试以改进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[28]从候选的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[29]通过镜像法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，提出了镜像适应性随机测试（MART），减少了适应性随机测试的计算开销。Chen 等人[30]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，实现最远距离测试用例的选取，提出了基于随机分区的适应性随机测试。
- **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[41]，如分支覆盖[42][43]、判定覆盖[44]、数据流覆盖[45][46]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[47][48]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[49]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[50][51][52][53].
-

批注 [U2]: 用这么多篇幅说这些的目的？

测试结果判定[31]（待补充）。

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

- 智能化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；
- 智能化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；
- 智能化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，智能化软件系统的测试面临诸多新的问题与挑战。

- **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。
- **智能软件系统的模拟测试平台难以搭建**：智能化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、

批注 [付3]: 测试结果判定方面的工作进行补充

批注 [U4]: 传统软件什么样？

批注 [U5]: 深度学习的特点？

批注 [U6R5]:

批注 [U7]: 不理解，从下面的意思中看出是数据跨界融合。建议改为：数据跨界融合

批注 [U8]: 这句话的意思是：前一句叙述的特点让系统适应外界变化。表达的意思感觉有点偏：好像研究的是自适应软件一样。建议改为：不断满足新的需求

批注 [U9]: 整段表达的意思是：在运行过程中，智能软件通过。。。。实现。。。是不对的。智能软件只有在开发过程中具有上述特征，一旦正式运行将不再演化。

批注 [U10]: 很多服务具备啊

批注 [U11]: 分析不够，感觉很空，能否进一步详细分析

批注 [U12]: 这是一个实用的挑战

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。

- **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。
- **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；智能化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

批注 [U13]: 在哪里看的？

批注 [U14]: 怎么样才算是宏？

批注 [U15]: 怎么才算是微？

批注 [U16]: 根据你的叙述，宏观测试就是系统测试，基于功能的测试本身就不是针对系统的，怎么能拿一种技术的缺陷来说明整个领域存在的问题？

批注 [U17]: 额，这种方式为什么耗时耗力

批注 [U18]: 我不确定智能软件系统（我打算认为智能软件系统就是指基于学习的系统）中能不能说白盒。。。

批注 [U19]: 根据你的论述，这里的智能软件是与外界有交互的，也就意味着智能软件的输入包含周围的环境。因此，不同的应用场景与同一输入相互矛盾。如果不同意请举例子

智能软件系统测试结果验证的困难性。

- **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。

批注 [U20]: ？质量评价指标？

批注 [U21]: “特有的人工智能是啥？”

综上所述，智能软件系统测试方面的研究工作仍存在诸多不足，**亟待探索有效且高效的新型智能软件系统的测试方法与技术。本课题旨在针对智能软件系统测试的测试方面的问题与挑战，探索智能软件系统的测试关键技术，以系统有效的方式检测与评估智能软件系统的质量，为开发可靠的智能软件系统提供新型测试理论与工具支持。**

参考文献：

[1] 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.

[2] Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge[J]. *Nature*, 2017, 550(7676): 354-359.

[3] IBM. Deep Blue–Overview [EB/OL]. [2018-11-18]. http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/.

[4] Ziegler Chris. A Google self-driving car caused a crash for the first time [EB/OL]. [2018-11-18]. https://www.theverge.com/2016/2/29/11134344/google-self-driving-car-crash-report.

[5] 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.

[6] 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.

[7] King J C. A new approach to program testing[C]. In *Proceedings of the International Conference on Reliable Software*. ACM, 1975, 228-233.

[8] 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 2008). Springer, 2008: 337-340.

[9] 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 2007). Springer, 2007: 519-531.

[10] Choco Solver [EB/OL]. [2018-11-18]. http://www.choco-solver.org/.

[11] 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.

[12] Godefroid 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 2015). ACM, 2005, 213-223.

[13] 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 2008). 2008, 209-224.

[14] 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 2008). 2008, 151-166.

[15] 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 2007). ACM, 2007: 31-36.

[16] 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.

[17] 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 2003). South African Institute for Computer Scientists and Information Technologists, 2003: 296-300.

[18] 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.

[19] 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 2007). IEEE Computer Society, 2007: 549-556.

[20] 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.

[21] Ghazi S A, Ahmed M A. Pair-wise test coverage using genetic algorithms[C]. In *Proceedings of the Congress on Evolutionary Computation* (CEC 2003). IEEE Computer Society, 2003, 1420-1424.

[22] 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 2007). ACM, 2007: 1082-1089.

[23] Williams A W. Determination of test configurations for pair-wise interaction coverage[M]. Testing of Communicating Systems. Springer, Boston, MA, 2000: 59-74.

[24] 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.

[25] 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 2015). ACM, 2015: 1797-1802.

[26] 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 2013). IEEE, 2013: 249-256.

[27] 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* (ICSE2016). ACM, 2016: 643-654.

[28] Chen T Y, Leung H, Mak I K. Adaptive random testing[C]. In *Proceedings of the 9th Annual Asian Computing Science Conference* (ASIAN 2004). Springer, 2004: 320-329.

[29] 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.

[30] 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 2004). IEEE, 2004: 79-86.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] 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 2017). ACM, 2017: 1-18.

[38] Argall B D, Chernova S, Veloso M, et al. A survey of robot learning from demonstration[J]. *Robotics and autonomous systems*, 2009, 57(5): 469-483.

[39] Kuefler A, Morton J, Wheeler T, et al. Imitating driver behavior with generative adversarial networks[C]// In *Proceedings of IEEE Intelligent Vehicles Symposium* (IV). IEEE, 2017: 204-211.

[40] Tian Y, Pei K, Jana S, et al. DeepTest: Automated testing of deep-neural-network-driven autonomous cars[C]. In *Proceedings of the 40th International Conference on Software Engineering* (ICSE 2018). ACM, 2018: 303-314.

[41] Harman M, Mansouri S A, Zhang Y. Search based software engineering: A comprehensive analysis and review of trends techniques and applications[R]. Technical Report TR-09-03, Department of Computer Science, King's College London, 2009.

[42] Ahmed M A, Hermadi I. GA-based multiple paths test data generator[J]. *Computers & Operations Research*, 2008, 35(10): 3107-3124.

[43] Alshraideh M, Bottaci L. Search-based software test data generation for string data using program-specific search operators[J]. S*oftware Testing, Verification and Reliability*, 2006, 16(3): 175-203.

[44] Xiao M, El-Attar M, Reformat M, et al. Empirical evaluation of optimization

algorithms when used in goal-oriented automated test data generation techniques[J]. *Empirical Software Engineering*, 2007, 12(2): 183-239.

[45] Girgis M R. Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm[J]. *Journal of Universal Computer Science*, 2005, 11(6): 898-915.

[46] Ghiduk A S, Harrold M J, Girgis M R. Using genetic algorithms to aid test-data generation for data-flow coverage[C]//In *Proceedings of the 14th Asia-Pacific Software Engineering Conference* (APSEC 2007). IEEE, 2007: 41-48.

[47] Li H, Lam C P. An ant colony optimization approach to test sequence generation for state-based software testing[C]// In *Proceedings of the 5th International Conference on Quality Software* (QSIC 2005). IEEE Computer Society, 2005, 255-264.

[48] Li H, Lam C P. Using anti-ant-like agents to generate test threads from the UML diagrams[C]//In *Proceedings the 17th IFIP International Conference on Testing of Communicating Systems* (TestCom 2005). Springer, 2005: 69-80.

[49] Lefticaru R, Ipate F. Automatic state-based test generation using genetic algorithms[C]//In *Proceedings of the 9th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (SYNASC 2007). IEEE, 2007: 188-195.

[50] Emer M C F P, Vergilio S R. GPTesT: A testing tool based on genetic programming[C]//In *Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation* (GECCO 2002). Morgan Kaufmann Publishers Inc., 2002: 1343-1350.

[51] Masud M, Nayak A, Zaman M, et al. Strategy for mutation testing using genetic algorithms[C]//In *Proceedings of Canadian Conference on. Electrical and Computer Engineering*. IEEE, 2005: 1049-1052.

[52] Baudry B, Fleurey F, Jézéquel J M, et al. Automatic test case optimization: A bacteriologic algorithm[J]. *IEEE Software*, 2005, 22(2): 76-82.

[53] Baudry B, Fleurey F, Jézéquel J M, et al. From genetic to bacteriological algorithms for mutation-based testing[J]. *Software Testing, Verification and Reliability*, 2005, 15(2): 73-96.

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）；

1、研究目标

本课题的研究目标是为开发可靠的智能软件系统提供新型、高效的测试理论，探索智能软件系统的数据驱动式测试方法与技术，开发支持智能软件系统测试的工具原型。

2、主要研究内容

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：

（1） 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：

- **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。
- **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。
- **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。

（2） 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：

- **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

批注 [U22]: 个人感觉很大程度上可以借鉴传统软件待测功能的描述方法。

批注 [U23]: 站不住脚，参照最开始的第一个特点的评论。

批注 [U24]: 这个可以

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。

- **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。

（3） 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：

- **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。
- **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。
- **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。
- **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

批注 [U25]: 没有看出来局限，研究动机不明确

批注 [U26]: 结合动机？

估技术，以满足测试用例集充分性评估的需求。

（4） 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：

- **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。
- **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。

（5） 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：

- **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。
- **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。

（6） 面向智能软件系统的测试支持工具与实例验证

- 设计与实现一个面向智能软件系统的测试支持工具。

批注 [U27]: 除了蜕变测试还有很多测试结果验证技术，作为国重，我感觉不要局限于一种方法。

- 采用多个智能软件系统及应用场景验证面向智能软件系统的数据驱动式测试方法与技术的有效性与效率。

3. **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）；

4. **本项目的特色与创新之处；**

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）。

（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）；

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）；

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）；

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）。

（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）。

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。

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。

4. 其他。