**研究报告**

**基于搜索的软件工程与自适应系统**

**王璐**

**2016年6月**

目录

[1 基于搜索的软件工程 4](#_Toc456722458)

[1.1 基本定义 4](#_Toc456722459)

[1.2现状分析 5](#_Toc456722460)

[ 文献分析 5](#_Toc456722461)

[ 会议情况分析 5](#_Toc456722462)

[ Harman教授团队 7](#_Toc456722463)

[1.3相关技术 17](#_Toc456722464)

[ 遗传算法（Genetic Algorithm） 18](#_Toc456722465)

[ 粒子群算法（Particle Swarm Optimization） 19](#_Toc456722466)

[ 爬山法（Hill Climbing ） 20](#_Toc456722467)

[1.4研究方向 21](#_Toc456722468)

[ Requirements 22](#_Toc456722469)

[ Design tools and techniques 24](#_Toc456722470)

[ Software/Program verification and model checking 26](#_Toc456722471)

[ Distribution，maintenance and enhancement 27](#_Toc456722472)

[ Management 29](#_Toc456722473)

[1.5对比分析 31](#_Toc456722474)

[ 需要解决的问题 33](#_Toc456722475)

[1.6未来展望 33](#_Toc456722476)

[ Generality and Applicability 33](#_Toc456722477)

[ Scalability 34](#_Toc456722478)

[ Robustness 34](#_Toc456722479)

[ Feedback and Insight 34](#_Toc456722480)

[2 自适应系统 35](#_Toc456722481)

[2.1基本定义 35](#_Toc456722482)

[2.2核心要素 36](#_Toc456722483)

[2.3基础理论 39](#_Toc456722484)

[2.4研究现状 40](#_Toc456722485)

[2.5主要方法 45](#_Toc456722486)

[ 软件产品线 45](#_Toc456722487)

[ 软件体系结构 46](#_Toc456722488)

[ 反射方法 46](#_Toc456722489)

[ 控制论 47](#_Toc456722490)

[ 面向服务的方法 48](#_Toc456722491)

[ 多Agent方法 48](#_Toc456722492)

[ 自然激励方法 49](#_Toc456722493)

[ 形式化建模与检查 50](#_Toc456722494)

[ 学习方法 51](#_Toc456722495)

[ 需求工程 51](#_Toc456722496)

[ 其他方法 52](#_Toc456722497)

[2.6对比分析 52](#_Toc456722498)

[ 缺陷分析 53](#_Toc456722499)

[ 未来挑战 54](#_Toc456722500)

[3 多目标自适应问题 54](#_Toc456722501)

[4 总结与展望 58](#_Toc456722502)

# 基于搜索的软件工程

## 1.1 基本定义

软件工程中有许多问题是需要从若干的候选集中寻求出最佳答案的，例如：

1. 可以覆盖所有程序分支的最小测试路径集合是什么？
2. 提高系统可维护性的最佳的系统体系结构设计方案是什么？
3. 为平衡开发成本和用户体验，系统的需求集合是什么？
4. 最佳资源分配策略是什么？
5. 某一软件系统最优的重构顺序是什么？

上述问题虽然涉及到软件的测试、设计、需求工程、工程管理等多个领域，也存在很多研究，但本质上来看都属于优化问题。因此，也容易被映射为基于搜索的优化问题从而产生了基于搜索的软件工程这一个新的研究领域

搜索是指数学领域的基于搜索的优化即Search-based Optimization SBO，将非形式化的软件工程问题，映射为SBO问题。搜索这个概念，和文献或超文本搜索不同，搜索问题是一种将优化或近似优化的解决方案建立为搜索空间，成为候选集合，然后通过适应度函数，从方法中找到更好的策略

最早思考将优化技术引入SE思想在1970s就产生了，最早的研究是Miller和Spooner在1976年，利用优化方法解决软件测试问题。而SBSE最早是在2001年由Harman和Jones提出的。当然，这篇文章虽然被认为是SBSE的Manifesto，但在1994年的时候，Carl Chang就在IEEE Software editorial中提出了在SE中应该更广泛的应用进化计算的思想。

Harman和Jones在2007的文章中提出了关于在SE中应用SBO的两个关键的问题：

1.如何选择对问题的表示

2.如何定义适应度函数

## 1.2现状分析

### 文献分析

通过对近年来关于基于搜索的软件工程的文章进行分析，我们可以看出该方向的研究已成为软件工程领域的研究热点。主要研究方向侧重于软件测试领域。

表1 文献数量统计

表2 文献方向统计

### 会议情况分析

中国基于搜索的软件工程研讨会CSBSE，由中科院软件所、清华大学、南京大学、上海交通大学、东南大学、大连理工大学的多名学者参与。研讨方向涵盖：项目规划、软件测试、车间调度、特征选择。

表3 会议举办情况

|  |  |  |
| --- | --- | --- |
| **时间** | **地点** | **举办单位** |
| 2012年7月 | 北京 | 北京化工大学 |
| 2013年6月 | 大连 | 大连理工大学 |
| 2014年7月 | 徐州 | 中国矿业大学 |
| 2015年6月 | 南京 | 南京大学 |

软件学报JoS曾举办过基于搜索的软件工程研究专刊。该专刊在2015年7月征稿，2016年4月发表于第27卷第4期，收稿28篇，最终收录12篇。

图1 文章方向统计

国际会议SBSE集中于对基于搜索的软件工程进行研究，从2011年开始举办。

表4 文献数量统计

表5 会议举办情况统计

|  |  |
| --- | --- |
| **时间** | **举办地** |
| 2011年7月 | Szeged, Hungary |
| 2012年7月 | Riva del Garda, Italy |
| 2013年6月 | t. Petersburg, Russia |
| 2014年7月 | Fortaleza, Brazil |
| 2015年6月 | Bergamo, Italy |
| 2016年12月 | North Carolina, USA |

图2 会议文章方向统计

### Harman教授团队

**主要学术论文**

|  |  |  |
| --- | --- | --- |
| [An analysis and survey of the development of mutation testing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:eQOLeE2rZwMC)  Y Jia, M Harman  Software Engineering, IEEE Transactions on 37 (5), 649-678 | [608](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=7093006449542376691) | 2011 |
| [The current state and future of search based software engineering](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:u5HHmVD_uO8C)  M Harman  2007 Future of Software Engineering, 342-357 | [509](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=9802273223008121758) | 2007 |
| [Regression testing minimization, selection and prioritization: a survey](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:7PzlFSSx8tAC)  S Yoo, M Harman  Software Testing, Verification and Reliability 22 (2), 67-120 | [489](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=13748823959745241920) | 2012 |
| [Search-based software engineering](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:u-x6o8ySG0sC)  M Harman, BF Jones  Information and software Technology 43 (14), 833-839 | [476](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=9919909975533558027) | 2001 |
| [Search algorithms for regression test case prioritization](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:9yKSN-GCB0IC)  Z Li, M Harman, RM Hierons  Software Engineering, IEEE Transactions on 33 (4), 225-237 | [468](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=13603083843116261073) | 2007 |
| [Using formal specifications to support testing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:ufrVoPGSRksC)  RM Hierons, K Bogdanov, JP Bowen, R Cleaveland, J Derrick, J Dick, ...  ACM Computing Surveys (CSUR) 41 (2), 9 | [284](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=8701672188994041348) | 2009 |
| [Reformulating software engineering as a search problem](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:d1gkVwhDpl0C)  J Clarke, JJ Dolado, M Harman, R Hierons, B Jones, M Lumkin, B Mitchell, ...  Software, IEE Proceedings- 150 (3), 161-175 | [279](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=13504433044590985239) | 2003 |
| [A theoretical and empirical study of search-based testing: Local, global, and hybrid search](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:qxL8FJ1GzNcC)  M Harman, P McMinn  Software Engineering, IEEE Transactions on 36 (2), 226-247 | [217](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=16678916924423704440) | 2010 |
| [Search based software engineering: A comprehensive analysis and review of trends techniques and applications](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:LkGwnXOMwfcC)  M Harman, SA Mansouri, Y Zhang  Department of Computer Science, King’s College London, Tech. Rep. TR-09-03 | [217](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=9969815425739119142) | 2009 |
| [Pareto efficient multi-objective test case selection](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:Tyk-4Ss8FVUC)  S Yoo, M Harman  Proceedings of the 2007 international symposium on Software testing and ... | [215](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=18333885440718156994) | 2007 |
| [Search-based software engineering: Trends, techniques and applications](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:jFemdcug13IC)  M Harman, SA Mansouri, Y Zhang  ACM Computing Surveys (CSUR) 45 (1), 11 | [206](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=1454353810395296838) | 2012 |
| [Testability transformation](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:2osOgNQ5qMEC)  M Harman, L Hu, R Hierons, J Wegener, H Sthamer, A Baresel, M Roper  Software Engineering, IEEE Transactions on 30 (1), 3-16 | [193](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=17032705378732120742) | 2004 |
| [Using program slicing to assist in the detection of equivalent mutants](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:IjCSPb-OGe4C)  R Hierons, M Harman, S Danicic  Software Testing, Verification and Reliability 9 (4), 233-262 | [182](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=14340776921491883847) | 1999 |
| [An overview of program slicing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:UeHWp8X0CEIC)  M Harman, R Hierons  Software Focus 2 (3), 85-92 | [157](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=17498301528275354167) | 2001 |
| [Pareto optimal search based refactoring at the design level](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:YOwf2qJgpHMC)  M Harman, L Tratt  Proceedings of the 9th annual conference on Genetic and evolutionary ... | [152](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=3766309458164614024) | 2007 |
| [The multi-objective next release problem](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:0EnyYjriUFMC)  Y Zhang, M Harman, SA Mansouri  Proceedings of the 9th annual conference on Genetic and evolutionary ... | [148](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=11756015840448162960) | 2007 |
| [Amorphous program slicing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:qjMakFHDy7sC)  M Harman, S Danicic  Program Comprehension, 1997. IWPC'97. Proceedings., Fifth Iternational ... | [147](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=17021915737674955725) | 1997 |
| [Software module clustering as a multi-objective search problem](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:dshw04ExmUIC)  K Praditwong, M Harman, X Yao  Software Engineering, IEEE Transactions on 37 (2), 264-282 | [145](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=14141327013561504506) | 2011 |
| [An orchestrated survey of methodologies for automated software test case generation](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:D_tqNUsBuKoC)  S Anand, EK Burke, TY Chen, J Clark, MB Cohen, W Grieskamp, ...  Journal of Systems and Software 86 (8), 1978-2001 | [144](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=12654577515499039369) | 2013 |
| [Higher order mutation testing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&citation_for_view=IwSN8IgAAAAJ:k_IJM867U9cC)  Y Jia, M Harman  Information and Software Technology 51 (10), 1379-1393 | [136](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=5373424549860519324) | 2009 |
| [Using program slicing to simplify testing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:zYLM7Y9cAGgC)  M Harman, S Danicic  Software Testing, Verification and Reliability 5 (3), 143-162 | [125](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=3347136805015439333) | 1995 |
| [A multiple hill climbing approach to software module clustering](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:Se3iqnhoufwC)  K Mahdavi, M Harman, RM Hierons  Software Maintenance, 2003. ICSM 2003. Proceedings. International Conference ... | [121](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=8866307499416660002) | 2003 |
| [A survey of empirical results on program slicing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:W7OEmFMy1HYC)  D Binkley, M Harman  Advances in Computers 62, 105-178 | [118](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=6339435316617488000) | 2004 |
| [A New Representation And Crossover Operator For Search-based Optimization Of Software Modularization.](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:_FxGoFyzp5QC)  M Harman, RM Hierons, M Proctor  GECCO 2, 1351-1358 | [118](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=2411115691277445434) | 2002 |
| [Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:O3NaXMp0MMsC)  S Yoo, M Harman, P Tonella, A Susi  Proceedings of the eighteenth international symposium on Software testing ... | [114](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=3760918441100640124) | 2009 |
| [Search-based techniques applied to optimization of project planning for a massive maintenance project](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:roLk4NBRz8UC)  G Antoniol, M Di Penta, M Harman  Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE ... | [111](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=2261948400952408618) | 2005 |
| [Evolutionary testing in the presence of loop-assigned flags: a testability transformation approach](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:Y0pCki6q_DkC)  A Baresel, D Binkley, M Harman, B Korel  ACM SIGSOFT Software Engineering Notes 29 (4), 108-118 | [111](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=1260553743798669848) | 2004 |
| [Constructing subtle faults using higher order mutation testing](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:e5wmG9Sq2KIC)  Y Jia, M Harman  Source Code Analysis and Manipulation, 2008 Eighth IEEE International ... | [107](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=16947636315865964197) | 2008 |
| [Metrics are fitness functions too](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:kNdYIx-mwKoC)  M Harman, J Clark  Software Metrics, 2004. Proceedings. 10th International Symposium on, 58-69 | [107](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=10092950827395001630) | 2004 |
| [A multi-objective approach to search-based test data generation](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:9ZlFYXVOiuMC)  K Lakhotia, M Harman, P McMinn  Proceedings of the 9th annual conference on Genetic and evolutionary ... | [102](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=5159039809664896776) | 2007 |
| [Search based approaches to component selection and prioritization for the next release problem](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:4TOpqqG69KYC)  P Baker, M Harman, K Steinhöfel, A Skaliotis  Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on ... | [102](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=2123396513634173125,10255647089308201497) | 2006 |
| [How to overcome the equivalent mutant problem and achieve tailored selective mutation using co-evolution](https://g.alexyang.me/citations?view_op=view_citation&hl=zh-CN&user=IwSN8IgAAAAJ&cstart=20&citation_for_view=IwSN8IgAAAAJ:mVmsd5A6BfQC)  K Adamopoulos, M Harman, R Hierons  Genetic and Evolutionary Computation–GECCO 2004, 1338-1349 | [102](https://g.alexyang.me/scholar?oi=bibs&hl=zh-CN&cites=13268545203610586795,13617793459185312970) | 2004 |

**Harman教授团队成员**

[Dave Binkley](https://g.alexyang.me/citations?user=wY56fjQAAAAJ&hl=zh-CN)

[Loyola University Maryland](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=12377646102442628794)

[Semantics based tools](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:semantics_based_tools), [IR in SE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:ir_in_se), [Slicing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:slicing), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Yue Jia](https://g.alexyang.me/citations?user=DG4Q08UAAAAJ&hl=zh-CN)

[University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Mutation Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:mutation_testing), [Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Shin Yoo](https://g.alexyang.me/citations?user=W9ymXf4AAAAJ&hl=zh-CN)

[Korea Advanced Institute of Science and Technology](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=5340164518610934029)

[Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Evolutionary Computation](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:evolutionary_computation), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Yuanyuan Zhang](https://g.alexyang.me/citations?user=Y7if1VEAAAAJ&hl=zh-CN)

[University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Requirements Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:requirements_engineering), [Evolutionary Computation](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:evolutionary_computation)



[Phil McMinn](https://g.alexyang.me/citations?user=ll6Fc7gAAAAJ&hl=zh-CN)

[University of Sheffield](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=9829198446306616343)

[Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing), [Search-Based Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:search_based_software_engineering), [Meta-heuristic Algorithms](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:meta_heuristic_algorithms), [Agent-Based Modeling](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:agent_based_modeling)



[Zheng Li](https://g.alexyang.me/citations?user=crKr1qAAAAAJ&hl=zh-CN)

Professor, Beijing University of Chemical Technology

[SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing), [Program Analysis](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:program_analysis)



[Jens Krinke](https://g.alexyang.me/citations?user=y8MpLZwAAAAJ&hl=zh-CN)

Senior Lecturer in Computer Science, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Program Analysis](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:program_analysis), [Slicing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:slicing), [Clone Detection](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:clone_detection), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Nicolas Gold](https://g.alexyang.me/citations?user=H0jzfvwAAAAJ&hl=zh-CN)

Senior Lecturer in Computer Science, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Computer Music](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:computer_music), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Digital Humanities](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:digital_humanities), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Paolo Tonella](https://g.alexyang.me/citations?user=6n3gAUMAAAAJ&hl=zh-CN)

[Fondazione Bruno Kessler, Trento](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=8352841911256301593), Italy

[Computer Science](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:computer_science), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing)



[Kiran Lakhotia](https://g.alexyang.me/citations?user=6Cqrrn8AAAAJ&hl=zh-CN)

Research Associate, CREST, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse), [Software Testing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_testing), [Dynamic Symbolic Execution](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:dynamic_symbolic_execution)



[Justyna Petke](https://g.alexyang.me/citations?user=a6kdFxIAAAAJ&hl=zh-CN)

Research Associate, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Constraints](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:constraints), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse), [Genetic Improvement](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:genetic_improvement), [SAT](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sat), [CSP](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:csp)



[Afshin Mansouri](https://g.alexyang.me/citations?user=eHKFZJIAAAAJ&hl=zh-CN)

Professor of Operations and Supply Chain Management, [Brunel University London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=14480557975201879779)

[Humanitarian Operations and Disaster Management](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:humanitarian_operations_and_disaster_management), [Sustainable Operations Management](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sustainable_operations_management), [Supply Chain Management](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:supply_chain_management), [Management Science](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:management_science)



[Federica Sarro](https://g.alexyang.me/citations?user=nW9MDIQAAAAJ&hl=zh-CN)

Senior Research Associate, CREST, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Empirical Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:empirical_software_engineering), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse), [Effort Estimation and Defect Prediction](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:effort_estimation_and_defect_prediction), [App Store Analysis](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:app_store_analysis), [Software Measures](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_measures)



[Laurence Tratt](https://g.alexyang.me/citations?user=_iWWkmIAAAAJ&hl=zh-CN)

Software Development Team, Informatics, King's College London

[Programming languages](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:programming_languages), [domain specific languages](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:domain_specific_languages), [compilers](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:compilers), [virtual machines](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:virtual_machines)



[Massimiliano Di Penta](https://g.alexyang.me/citations?user=j6ucyOAAAAAJ&hl=zh-CN)

[University of Sannio](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=1403511252509235021), Italy

[Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Mining Software Repositories](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:mining_software_repositories), [Software Evolution](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_evolution), [SBSE](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:sbse)



[Kelly Androutsopoulos](https://g.alexyang.me/citations?user=0VGY3MQAAAAJ&hl=zh-CN)

Senior Lecturer in Software Engineering, [Middlesex University](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=12131964207698724264)

[Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering), [Modelling](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:modelling), [Slicing](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:slicing)



[David Clark](https://g.alexyang.me/citations?user=Yr7RX3MAAAAJ&hl=zh-CN)

Computer Science, [University College London](https://g.alexyang.me/citations?view_op=view_org&hl=zh-CN&org=7707954445345430443)

[Program Analysis](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:program_analysis), [Programming Languages](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:programming_languages), [Security](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:security), [Software Engineering](https://g.alexyang.me/citations?view_op=search_authors&hl=zh-CN&mauthors=label:software_engineering)



## 1.3相关技术

表6 技术统计

基于搜索的软件工程所采用的技术可以分为如下几种，其应用情况如表6所示。

* Evolutionary Algorithms (EAs)

Genetic Algorithms(GA)

Genetic Programming (GP)

Scatter Search (SS)

Evolutionary Strategies (ES)

Particle Swarm Optimization (PSO)

Estimation of Distribution

Algorithm (EDA)

* Simulated Annealing (SA)
* Hill Climbing (HC)
* Ant Colony Optimization (ACO)
* Tabu Search (TS)
* Artificial Immune Algorithm(AIS)
* Sequential Quadratic Programming(SQP)

### 遗传算法（Genetic Algorithm）

借鉴生物界的进化规律（适者生存，优胜劣汰遗传机制）演化而来的随机化搜索方法,由J.Holland教授1975年提出。

图3 遗传算法的优势

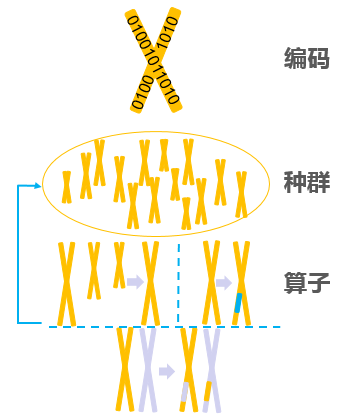


图4 遗传算法流程

### 粒子群算法（Particle Swarm Optimization）

在对动物集群活动行为观察基础上，利用群体中的个体对信息的共享使整个群体的运动在问题求解空间中产生从无序到有序的演化过程。1995 年由Eberhart 博士和Kennedy 博士提出。

+

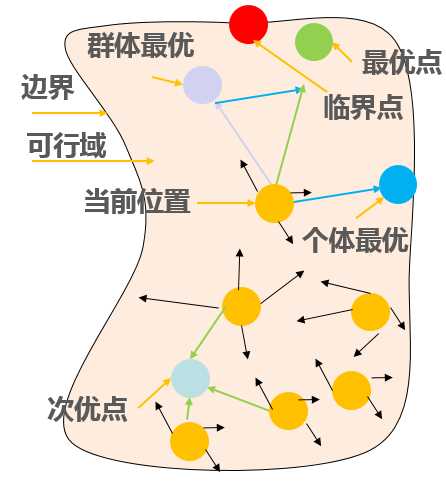


图5 粒子群算法流程

### 爬山法（Hill Climbing ）

一种简单的局部搜索技术，在SBSE领域应用广泛，为SBSE的许多新问题，提供快速，可行，可理解的初始结果

图6 爬山法优势

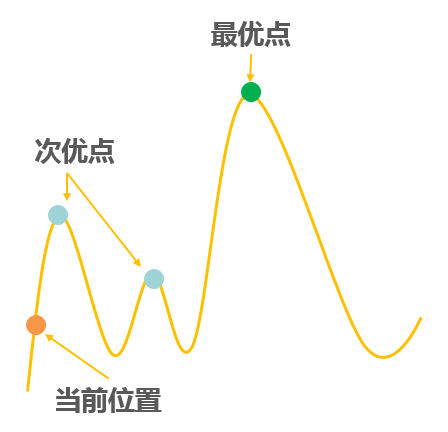


图6 爬山法算法流程

**蚁群算法（Ant Colony Optimization）**

与具体问题关系不紧密的优化算法，是一种用来在图中寻找优化路径的机率型技术，由Marco Dorigo于1992年在他的博士论文中提出

## 1.4研究方向

为研究基于搜索的技术与软件工程技术相结合的研究情况，以搜索工程的关键词与软件工程技术相结合，进行文献的分析。

表7 查询关键词

|  |  |  |
| --- | --- | --- |
| Search based | optimization | multiobjective optimization |
| Genetic Algorithms | Genetic programming | hill clmbing |
| Simulated Annealing | Tabu Search | Artificial Immune Algorithm |
| ant colony optimization | Sequential Quadratic Programming | Particle Swarm Optimization |

图7 软件工程关键技术

### Requirements

需求工程是软件过程中至关重要的环节，SBSE被用于优化需求的选择，依据需求和具体实现之间的关系，或者需求之间的优先级。

Bagnall利用Next Release Problem，NRP描述需求release 规划，描述了多种元启发式优化算法，包括贪心、branch-and-bound，SA和HC。

该文没有给出每个需求的价值属性，而是通过关联的cost进行衡量，该文的任务是找到利益相关者的子集，他们的需求将被满足。

优化目标是最大化利益相关者的利益，在公司资源的约束下，采用的策略是一个单目标的，基于规划的NRP，这是2001年，SBSE第一次用于需求领域。

Feather 和Menzies在2002年，建立了一个迭代模型去达到需求的近似最优根据实际中的NASA pilot 研究，提出了一个Defect Detection and Prevention DDP过程；DDP结合了需求交互模型和摘要工具，去提供一个近似最优，在解决风险消减和开销平衡空间中。该文是最早用Pareto最优方法解决需求的文章；但是前沿并不是如Jalali在2008提出的多目标技术产生的，而是迭代应用基于权重的单目标SA；Feather在2006年总结目前需求工程中形象化技术时，也提到了基于SA的Pareto前沿思想。

Ruhe在2004提出了基于GA的方法，即EVOLVE family 目的在于最大化需求传递带来的利益，在一个不断增加的软件版本发布计划中。他们的方法在平衡需求和可用的资源；评估和优化了不同命令冲突时的程度，根据利益相关者的优先级。他们接受需求的变化和两种类型的需求交互关系，并为一次迭代管理中的next release提供了候选集。使用单目标优化方法，将资源预算作为约束。

Carlshamre在2002，利用线性规划LP技术，将需求相关性也考虑进来。Ruhe 2005，使用整数规划方法结合了计算智能和人类协商以解决冲突的目标。Vanden Akker 2007进一步扩展了ILP并建立一个工具，整合需求选择和release planning的规划，去法相一个最优的需求集合，最大化的满足预算的收益。

Harman2006年，利用搜索方法选择构件，去包含不同的release以形成一个系统。Baker 2006认为确定next set of release of a 软件的问题是一个排序和选择候选软件构件的问题。他们使用贪心和SA方法解决这个问题。而Harman 2006年统一认为需求问题是一个特征子集选择问题，提出了一个单目标方法，并利用现实数据库Matorola进行测试。

AlBourae，2006，关注于解决需求变化问题，即重新规划产品release，建立了一个贪心的重计划方法，目的在于降低风险，增加需求个数，实现在变化下的搜索空间。

Cortellessa，2006，提出了一个优化框架提供Code Off-The-Shelf COTS和in-house components的选择提供决策支持。整数规划LINGO模型被提出用于自动满足需求，当最小化开销时。

与其他SE领域的问题一样，例如项目规划、NRP、区域测试等，特征子集选择问题和特征的排序即，优先级问题之间存在联系。因此关于Karlsson在1998年提出的需求优先级分析方法，和Greer2004提出的最优化配置需求以增值的方法，都是基于1.利用优先约束方法解决利益相关者之间的冲突排序2.平衡需求和资源，取得最大收益3.基于GA的持续版本增量式软件开发。

上述均为单目标方法，然而问题在于最大化一个变量时极有可能影响另一个变量的潜在最大化，导致搜索时存在一种偏见领导的情况。

Zhang 2007，提出了采用多目标优化方法解决NRP问题，优化价值和开销。

Saliu 2007在集成早先工作的基础上，展示了实现对象和需求对象如何利用多目标方法进行仿真优化。Zhang当时就是实现了两个目标的Pareto优化方法，考虑了实现层和需求层的目标。

Finkelsterin 2008，利用多目标方法探索需求分配产出的公平性。而对于公平性有着不同的定义，例如每个用户希望从开发者处得到相等的预算，或者他们获得了和其他人一样多的被满足的需求。他们就研究了不同的公平性定义下的不同的优化目标函数。

以上说明，SE问题一般是内在、不确定、定性和不正规的。使用SBSE可以形式化问题为定量的优化问题。

### Design tools and techniques

这是SBO另一个应用广泛的领域，是获得更好的开发设计。这里有很多被广泛接受的度量方法，例如内聚和耦合，也有很多工作可以做去优化这个问题。Doval 1999。但是先前关于内聚和耦合的并不认为是在对设计本身进行研究，他们更多的关注的是系统在实现后的，关于模型边界的重组。所以早先的工作更多的应该被分类为维护而不是设计阶段应该做的事。

软件的设计和再设计（即维护）之间必然存在联系。很多文献关于软件维护时的SBSE方法同样也适用于设计阶段。Simons 2006-2008提出了多目标GA方法实现面向对象软件的设计。和之前关于软件维护阶段的内聚和耦合的研究（Harman 2002等）一样，适应度函数的设计收到了相似的SE目标的启发。然而目标是上有的设计目标而不是下游的维护目标。O‘Keeffe 2003将OO软件的设计问题转变为一个使用SA的优化问题。提出了用来评估设计质量的度量方法。

设计模式被认为是可以也必须作为利用SBSE解决设计问题的基石。Raiha 2008 就研究过这种可能性。他提出了一种基于GA的，基于若干设计模型的软件体系结构自动生成方法。

其他的研究更加侧重于软件设计过程的研究。Feldt 199提出了一种模型去探索早期的软件开发过程的困难程度，利用GP，并描述了一个较好软件开发平台WISE的原型，它使用了生物进化算法2002。也有研究QOS的，例如Khoshgoftaar 2004提出一种基于GP的欧姆表优化模型。

Qos感知的web服务组合问题被Canfora 2005使用GA进行了解决。这是面向服务计算的核心问题，其他学者Jaeger2007也讨论过Quality-of-Service-based Web Service Selection问题，利用GA。 Ma 2008提出基于GA的服务组合和选择，以Qos作为约束。

也有的研究关注于构件选择和集成问题。构件选择问题和需求分配问题比较相近。Baker 2006，提出了基于贪心和SA解决构件选择问题，Yang 2006提出了使用GA以降低软件集成问题中的风险。经典的OR技术也被用于构件选择问题的解决中，Desons结合回溯和分支定界法实现自动构件替代问题，实现软件重用和演化。其他作者将构件的选择问题看做是一种选择优化问题。例如Cortellessa 2008提出了一种支持COTS 构件选择的框架，实现最小化系统约束开销。Vijayalakshmi2008提出了基于GC的构件优化组合，Kuperberg 2008提出了基于GA的平台无关的再工程参数行为模型，为了黑盒构件的性能预测问题。

State-base模型目前已经是常用的设计方法，并为应用SBSE提供了基础，因为目前已经存在了很多利用优化方法建立的基于状态的模型。Goldsby 2008，提出了一个基于演化的工具，为了软件行为模型的产生，以提升软件质量。AvidaMDE系统产生了一系列通信的基于状态模型，关于系统行为，使用模型推理技术，根据情况合成了有限状态自动机。Lucas 2005提出了一个EA方法，用于学习确定性有限自动机，优化分配状态标签，利用证据驱动的状态合并算法比较其性能。

Feldt，最早将SBO应用于SE的先驱，揭示了错误容忍可以利用GP加入到系统中，以解决多种不同的软件变形，1998. 这是一种新方法，N-version 计算，基于这个，高度容错的系统可以被制造多次，利用的不同的方法，以增加其鲁棒性。目标是增加质量，由于GP进化了版本，定性的不同于任何人类产生的不同版本。

在传统的N版本计算方法中，不同团队的程序员将开发不同（希望能够成为多样化的）的方案以解决同一个问题。当然，对于一个系统开发不同的版本的这种方法，是个高开销的方式去解决软件鲁棒性和容错，这只会被用于高度安全攸关的情况，在这种情况下，经验是可信的。因此Feldt的工作提供了一种解决这种高开销的方法。

将SBSE用于设计在最近三年里有了很大的发展，也浮现出很多新的设计问题。Amoui 2006，利用GA解决了面向对象度量问题和找到系统转变的最优顺序以提升转变设计的质量。这与用SBO找到重构的最优顺序一样。Barlas 2008就提出了一个基于GA的多客户端多服务器的映射客户端和服务器数据传递的问题。Bowman 2008将Strength Pareto Evolutionary Algorithms 2 SPEA2应用于解决Class Responsibility Assignment CRA问题的决策支持系统中。Cao 2005，用GA处理了开销驱动的web服务选择问题。Chardigny 2008，提出了利用搜索方法抽取基于构件体系结构，从OO系统中。Sharma 提出了在实现性能最大化上部署构件的启发式方法。

### Software/Program verification and model checking

模型检测拥有庞大的搜索空间，并存在着很多的候选度量方法去引导一次搜索。Codefroid 1997首先利用SBO解决模型检查领域的状态空间。状态空间太大而不能全部被检查到，SBO可以识别同构子图并找到反例。Alba 2007使用ACO进行模型检测，找到反例。Mahanti 2006提出了使用ACO和PSO的模型检查方法。

Johnson 2007使用模型检查去衡量有限状态自动机样的适应度。Katz 2008提出了一种基于模型检查的GA方法，去确认并从规格中综合。他们利用GA框架，将Hoare-Logic-style assertion-base specifications 和模型检查进行结合。

### Distribution，maintenance and enhancement

软件维护是一个增强或优化已经部署的软件系统或软件版本的过程，当然也包括修补缺陷。它涉及到软件的变化，包括在使用中发现的正确性上的缺陷，或者错误，或者新增的功能以提升软件的可用性和实用性。

关于这个领域有两个核心的研究问题。一个是基于搜索的软件系统模块化。利用SBSE进行自动重构。这个问题可以继续被划分为模块化和重构。

另一个工作就是程序语言的演化，Van Belle 2002，实时任务的分配，Bate 2006，基于CA度量分类的质量预测，Vivanco 2004，遗产系统的迁移，Sahraoui 2002.SBSE被用于概念分配问题。Gold 2006将GA和HC用于发现重叠概念边界，因为关于可能分配方案的空间增长的过于迅速。而利用SBSE可以解决这个问题。

#### Modularization

Mancoridis 是第一个利用SBSE方法解决软件模块化问题的人，在1998年。他们的最初工作是利用HC去聚类模块，以实现最大化内聚和最小化耦合，从1998到2008这一段时间一直都进行研究。他们早先的研究产生了一个叫做Bunch的工具。

模块聚类问题和在图里找寻临近社团的问题类似，节点代表模块，边代表模块间的依赖关系。Mancoridis 1999，将这种图称为模块依赖关系图。而bunch工具提供了这种图的层次聚类，并支持用户选择聚类规模。

在Macoridis之后，其他学者也利用SBSE解决了模块聚类问题。Harman 2002分析了特别的模块化粒度，将之作为适应度函数，而Mahdavi 2003，将multiple hill climbs和简单的HC与GA相结合。而Harman 2005也提出了模块化质量的鲁棒性，Modularization Quality 适应度函数，与替代措施内聚和耦合、Evaluation MetricEVM等进行比较，在聚类基因表现数据中进行了应用。

Bodhuin 2007利用GA集合经典聚类算法，以降低封装规模，平均下载次数。Huynh 2007利用GA聚类设计结构模型Design Structure Matrices并检测设计和源代码结构之间的一致性。

基于HC及其扩展方法的研究，例如Harman 2002，Mahdavi 2003，这种简单的方法却在解决该问题时十分有用。然而，Praditwong 2010显示了多目标优化技术可以显著地胜过HC在模块质量方面。

很多方法都研究过聚类问题，在SE中也有很多研究前景，也不仅是模块结构问题。例如Cohen 2006显示了利用基于搜索的聚类算法解决Java程序优化中的堆式分配问题。内存破碎问题也被利用SBSE解决过，Del Rosso 2006针对内存碎片问题，利用GA寻找优化配置离散的自由列表数据结构。

#### Refactoring

在重构问题中，目标是变化程序，在不变更语义的前提下调整结构。Reformat 2003，研究了代码克隆问题，并提出了一种基于GP的自动代码克隆方法。Di Penta 2005提出了一种语言无关的软件renovation框架，去除无用对象，克隆和循环依赖，并聚类大的函数库形成更加内聚，更小的函数库。Cowan 2004，提供了一种基于GP的自动编程框架。Bouktif 2006利用SBSE规划重构步骤以去除重复代码。Antoniol 2003 提出基于GA的重构过程，以减少规模并最小化耦合的函数库。Bodhuin 2007提供了基于GA的重构决策的软件度量方法。

SBSE这类的工作可以根据一个目标进行划分，即目标是否是优化程序以重构出一个自己的新版本，还是优化实施重构时的操作流程步骤。也可以按照方法是单目标（将所有的度量） Seng 2005，或者采用多目标（Pareto）Harman 2007。Bouktif 2006提出了一种在约束和优先级下的重构操作规划方法，以去除重复代码。

这个工作非常接近状态层次基于搜索的转变，Ryan 2000提升了命令式语言以识别转变，并提出了GA以确定优化转变的次序，以最小化SPMD 单指令流多数据流Fortran程序的执行时间，在并行框架下的执行时间。

Stephenson 2003使用GP改进编译器的启发性。Hoste 2008和Dubach 2007使用一个变换方法提升编译代码的性能，通过搜索编译任选项，它控制着gcc中的优化层次。有60个这样的flag（纯粹的代表gcc中的优化行为），给优化选择提供了一个非平凡的空间，特别是目标在于提升编译代码的性能时。该文考虑了编译时间和代码质量（以执行时间表示），即程序编译时的非功能性需求。

重构的目标在于重新组织程序的结构，以提升结构方面的一些优势，而不会影响整个系统的对外行为。因此很多一般方法，例如source-to-source 程序转变和SBSE相结合。Fatiregun 2003-2005和Kessentini 2008 利用转变方法降低程序规模，并自动构造非定型的切片。而第一个采用搜索方法对任意形式的source-to-source程序进行转变的是Cooper 1999，他利用搜索方法找到最优化的编译序列。他更关注于整个程序，而后期如Ryan2000，Fatirregun1998更关注于微观或声明层的转变。

上述工作更关注OO范型，不过这些原则和声明层的转变方法是一样的。

### Management

管理软件生命周期中各个复杂的事物，寻找到最优化的软件产品过程，并保证该过程的产品最优。任务和资源分配、调度、成本绩效评估都是最常见的问题。而SBSE问题可以被分类为是否关心项目的计划活动，或者是否关心开销预估的决策支持。

#### project planning

Chang 1994是最早提出用SBSE解决管理问题的学者。他们早期的工作是利用SBSE管理软件，并提出了Software Project Management Net SPMNet，用于项目调度和资源分配，并利用仿真数据进行了测试。其他的早期工作还有Aguilar-Ruiz 2001年提出的利用Software Project Simulator SPS 评估适用性，使用一种进化搜索找到项目管理的管理规则。

在项目规则中，为工作包分配工作团队可以被看做是一种装箱问题的应用。Antoniol 2004和Chicano 2005利用了GA，HC，SA方法解决人力资源分配问题，他们同时认为再造工程和项目放弃同样是SE中的重要问题。所以他们利用现实的数据Year 2000 Y2K 维护项目进行了测试，而Chicano 认为多目标问题可以应用综合数据中，多个目标结合在一个单个的适应度函数中，利用他们的权重。

Bouktif 2006 使用SBSE解决确定预期软件系统质量问题，认为这是一个预测系统。利用基于GA的质量模型提升软件质量预测，并展示了一般问题在结合质量预测后，成为一个贝叶斯分类模型，并可以通过SA算法进行定制。而在2004年利用GA提成了基于规则集的OO系统质量预测。

Alvarez-valdes 2006使用Scatter Search SS算法解决项目规划问题，以最小化项目完成时间。这是少有的利用SS的SBSE方法。Barreto 2008提出了一种基于优化的项目策划人员算法，解决人员问题。Cortellessa 2008描述了一种支持软件体系结构决策的优化框架。Hericko 2008利用简单的基于梯度的优化方法，以最优化团队规模以降低项目开销。Kapur 2008利用GA提升人员优化在项目版本中，并在时间约束下提供给用户最好的质量。Kiper 2007，利用GA和SA选择最优的Verification and Validation V&V子集，以减少在预算约束下的风险。

目前该领域仍有一些没有解决的问题：

1.鲁棒性：在早期编译时间找到一个项目计划是不充分的。也许找到一个在变化发生时更鲁棒的计划是更加重要的。这种鲁棒的计划可能是编译时的次优，因此确认一些在编译时期不好的策略是十分值得做的，这样才能保证对变化的应对。这也就是关于健壮性的折衷，在优化文献中已经被广泛研究了，Beyer 2007.

2.不好的估计：所有的项目估计都要克服poor estimates，Shepperd 2007.许多工作都假设在项目计划开始阶段可以得到可行的估计。但是这是一种不切实际的假设。许多工作需要完成，以保证按照项目计划开发时能够处理那些预估只有局部正确性的情况。

3.集成：软件项目管理是软件开发生命周期中最高层的事物。他引导设计、开发、测试和维护工作。因此，项目管理不是理想上的可以被优化的一个孤立事物。所以如果要利用SBSE，就需要集成这些各种的管理事物。

软件项目管理也不能与需求工程孤立，因为不同需求的选择会影响到计划的可行性。因此，虽然需求收集，分析阶段领先于管理计划的构想，这也不能清晰的描述一个计划阶段的概念可以成为优化问题的一个构想。早期关于集成的工作，Saliu 2007显示如何实现需求和研究对象可以仿真优化，利用MOP。然而还有更多的工作需要将设计阶段也包含进来。

#### Cost Estimation

成本估计是一个很长苛求的任务，Shepperd 2007.对所有项目不只是涉及到项目的，评估活动是困难的，因为没办法预测不可预见的和自然趋势，因此无法分配任意数量的成本去未知的必须的事件中。而对于软件预测的困难可能比对一个普通项目更大，因为：

1.软件开发过程内在的不确定性

2.软件工程学科相对来说比较年轻

3.SE解决方案可以适用的很多任务，是不同的。

Dolado 2000，是第一个研究的，他利用GP进行项目成本估计，使用一种表征回归，symbolic regression，即繁殖简单数学函数，满足项目付出的观测函数，通过测量功能点。优势在于产出不仅是个观测系统，也是以解释观测系统行为的功能。

GP被广泛应用在该领域，Evett 1999，Liu 2001都用了GP进行预测，并提出连个case study。他们的GP是经过扩展、重定义的。GP预测也被当做决策支持的基础。Huang2008利用GP和GA，联系灰关联度分析技术，提升评估的正确性。Jarillo 2001，利用GA和GP进行工作量评估，进行缺陷和可靠性预测。Lokan 2005 研究基于GP的预测模型，在不同适用度函数下的性能。

Burgess 2003采用GP进行成本估计。Shan 2002比较了语法导向的GP方法和线性回归方法在软件开发成本估计方面的能力。Sheta 2006提出两个新的模型去建模软件开发过程中的成本预估，利用了GA，并利用NASA的软件项目数据库进行测试。Shukla 2000提出了一个neuro-genetic 方法使用一个一般训练过的神经网络预测软件所需的资源，基于历史数据。

Kirsopp 2002 预测未知的软件属性，依据已经知道的属性，并寻找近邻的项目，那些共享相似的已知值的项目。也就是案例推理方法Case-Based Reasoning 并被广泛应用在预测系统。CBR在现有数据有好的质量时，效果不错，但当缺失数据时，就不是很理想。Kirsopp 2002任务确定好的预测器，可以被形式化为特征子集选择问题，他们利用HC进行处理。

## 1.5对比分析

通过对以上方法的介绍，表8分析了这些技术的分布情况。

表8 文章分析

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **名称** | **分级** | **需求** | **设计** | **编程** | **检测** | **维护** | **度量** | **管理** | **测试** | **总计** |
| **GECCO** |  | 2 | 4 | 1 | 5 | 16 | 2 | 8 | 46 | 84 |
| **TR** |  |  | 3 |  |  |  |  |  | 15 | 18 |
| **IST** | B类期刊 | 2 | 1 | 1 |  |  |  | 5 | 5 | 14 |
| **Ph.D Thesis** |  |  | 1 |  |  | 3 |  |  | 11 | 15 |
| **CEC** |  |  | 1 |  | 3 |  |  |  | 8 | 12 |
| **ASE** | B类会议 |  |  |  |  |  |  |  | 14 | 14 |
| **SBST** |  |  |  |  |  |  |  |  | 12 | 12 |
| **MSc Thesis** |  |  | 1 |  |  |  |  |  | 8 | 9 |
| **ICSM** | B类会议 | 1 | 1 |  |  | 3 |  | 2 | 2 | 9 |
| **TSE** | A类期刊 |  |  |  |  | 1 |  | 1 | 6 | 8 |
| **ISSTA** | B类会议 |  |  |  |  |  |  |  | 8 | 8 |
| **COMPSAC** | C类会议 |  |  |  |  | 1 |  | 2 | 3 | 6 |
| **COR** |  |  |  |  |  |  |  | 2 | 5 | 7 |
| **CSMR** |  |  | 1 |  |  | 5 |  |  |  | 6 |
| **SEKE** | C类会议 |  |  |  |  | 1 |  | 1 | 2 | 4 |
| **JSS** | B类期刊 |  |  |  |  | 2 |  |  | 4 | 6 |
| **STVR** | B类期刊 |  |  |  |  |  |  |  | 5 | 5 |
| **Book** |  |  |  |  |  | 2 |  |  | 3 | 5 |
| **ICSE** | A类会议 |  |  |  |  |  |  |  | 3 | 3 |
| **Others** |  | 6 | 35 | 3 | 13 | 18 | 3 | 22 | 125 | 225 |

### 需要解决的问题

1. **普遍性和可应用性**

已用于软件工程领域的多个问题的解决中解决问题表示和适应度函数，即可使用该方法

1. **可扩展性**

软件工程问题需要一个可以扩展的结果集，而搜索方法天然就是可并行的

1. **健壮性**

方案的健壮性和方案本身的作用同样重要，而搜索到了一个方案后它周围都是备用的方案

1. **反馈和洞察**

通过搜索，避免了人为操作带来的偏见；并有可能获取到意想不到的结果

## 1.6未来展望

### Generality and Applicability

第一个显著的特点就是SBSE在解决SE问题上的广泛性。虽然测试是主要研究方面，并存在54%的文章，不过其他方面也有研究。

这种广泛性和适用性是由于SE的本质。在用SBO解决问题时，首先需要解决两个问题，一个是问题的重新表述，一个是适用度函数的定义，它表征了需要被优化的目标。一旦这两个问题可以解决，那么利用SBO就可以解决问题，并取得好的成果。

对于其他工程学科，也许表述问题不是那么容易。工程制品的物理特性使得仿真是唯一合算的选择。这就导致优化方法在一定层面上无法解决这些问题。并且，对于一些领域，如何度量工程制品的优化特性也不是一件清晰的事。即使好度量，但结果也不是容易收集的，这又导致了用优化的一个鸿沟。

然而软件并不具备什么物理特性。所以，在表征软件制品，并且软件本身就是采用一些无形的材质，例如信息，性能，逻辑表征的。这种无形给SE带来了很多问题，但同时，却给SBSE带来了巨大的优势。很少有SE问题无法重表示的，并且经常是立即可用的。

并且对SE，度量方法是广泛常用的，软件度量常在会议和期刊中研究。因此，假定适应度函数时，我们必将受益。

### Scalability

结果的可扩展性对于SE来说是最大的问题。实验室中产出的优雅的结果在实际中无法适用，这就是由于缺乏可扩展性。而幸运的是，SBSE天然是并行的。HC可以在并行下运行，通过在每次从不同的点开始爬山 Mahdavi 2003. GA是基于人口基数的，天然是并行的。对于每个个体的适应度计算可以通过并行的方式计算，在最小的开支下，Asadi 2010.搜索算法，无论是普通的，还是SBSE用的，都提供了一种killer application，对浮现的范例，用户层的并行计算。

这种趋势预示着与并行计算的结合，在处理扩展的SE问题时，SBSE的天然并行表明可以通过并行的SBSE解决SE的扩展问题。

### Robustness

在某些SE领域，方案的鲁棒性与方案的同样重要。例如找到一个周围都是好的策略的满足的策略比周围都是不好策略的策略要好。

因此，搜索需要找到稳定的，充足的区域在搜索空间中，这样近邻才有可能成为提出的策略的候补，在该策略才能抵抗问题特征的少量修改。Beyer 2007。

至今，SBSE关注于预测最好的策略，但是许多应用领域可能需要应对变化，因此，也行需要关注这个特征。

### Feedback and Insight

错误的直觉是SE的主要发生的错误，导致了被误解的说明书，关于需求的不充分沟通，设计方面的隐含假定。SBSE可以处理这些问题。不同于基于人类的搜索，自动搜索技术没有那些偏见。他们自动的根据适应度还是搜索空间，适应度函数则体现了人类的意志。

这也是SBO的中央坚固度之一。也说明了为什么SBO往往能够提供意想不到的结果。自动搜索和人类意志可进行有效的联合，在一个迭代的过程中进行优化，实现出更好的封装人类的假设和直觉的适应度函数。

# 自适应系统

## 2.1基本定义

图10 自适应软件的研发需求

**Ravindranathan，1998：**一个系统在变化的环境中，通过**选择执行相关模型**或者**变换模型间的关系**，以实现自己的目标。

**Brun，2009**：系统可**调整自身**行为以适应环境Self指系统可**自主决定**如何适应。

**Salehie，2011：**当系统发现自己**无法完成既定目标**或者**有更好的策略可以采取**时，将评估自身情况及环境情况，并作出调整。

**Naqvi，2012:**系统包含一个**闭环控制系统**，负责在运行时根据变化信息，自动调整系统变化来自**用户需求、系统特性和环境特征**

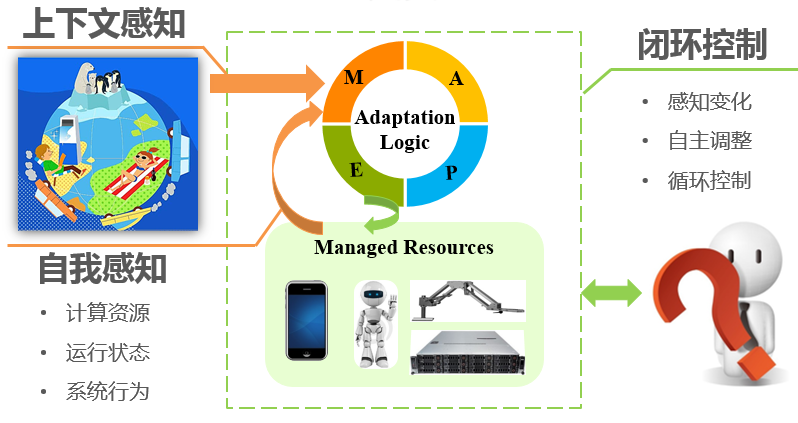


图11 自适应系统组成

## 2.2核心要素

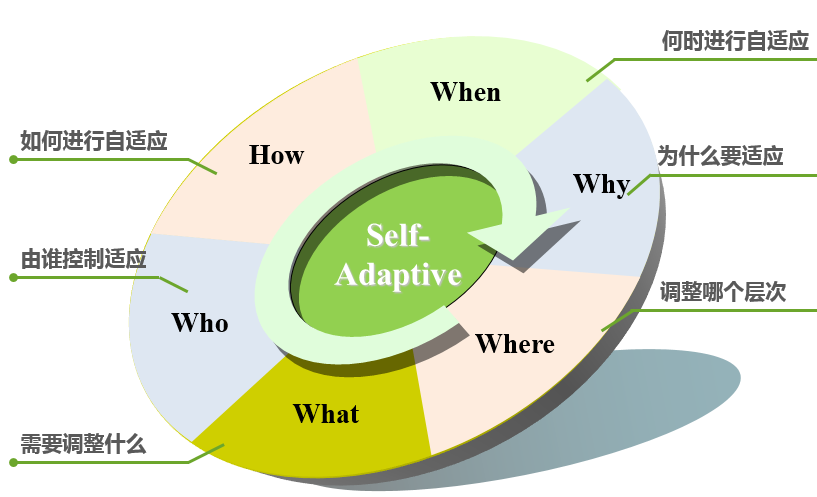


图12 自适应系统要素

时间要素反应了软件将什么时候进行自适应，一般存在如下两种方式：**反应reactive变化发生后，采取相应的自适应策略**；**主动proactive在变化发生前就修改软件，避免它的失效**。

* 用户更喜欢主动方法，但是相比于反应式算法，主动方法更加复杂
* 目前大部分方法都是反应式方法
* 主动方法的研究已经形成了一个新的方向“Self-optimization”

变化是引发自适应的原因，一般分为：

资源变化

* 硬件组件的损坏
* 软件错误
* 网络连接错误
* ……

环境变化

* 环境变量变化
* ……

需求变化

* 用户组的组成变化
* 用户喜好变化
* ……

软件可以在不同的层次实现自适应调整，一般分为：

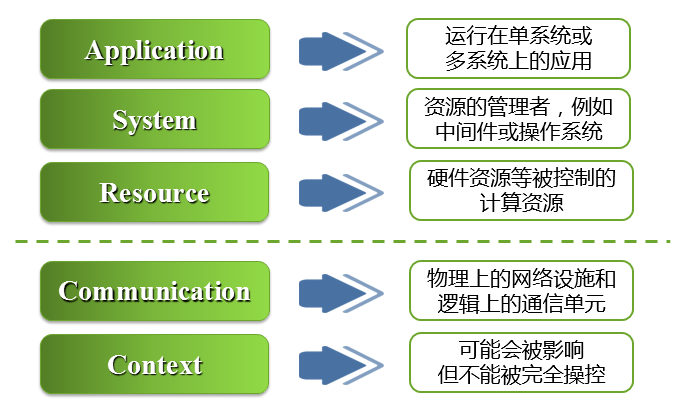


图13 软件变化层次

**软件调整什么才能实现自适应，可分为：参数**在运行过程中调整系统的参数，较为容易实现但参数之间可能会互相依赖；**上下文**属于可调整上下文的自适应软件系统的一种特殊方法；**结构**更换失效的构件，或者加入性能更好的构件，通过集成实现服务的切换或形成新的服务。

**控制自适应过程的控制者，存在如下的发展趋势：**



图12 自适应决策分类

**控制技术展现着软件具体如何进行自适应过程，分离方式包括：**

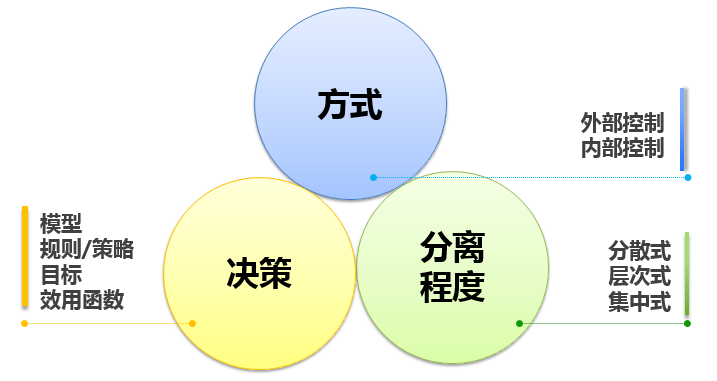


图13 控制循环分类

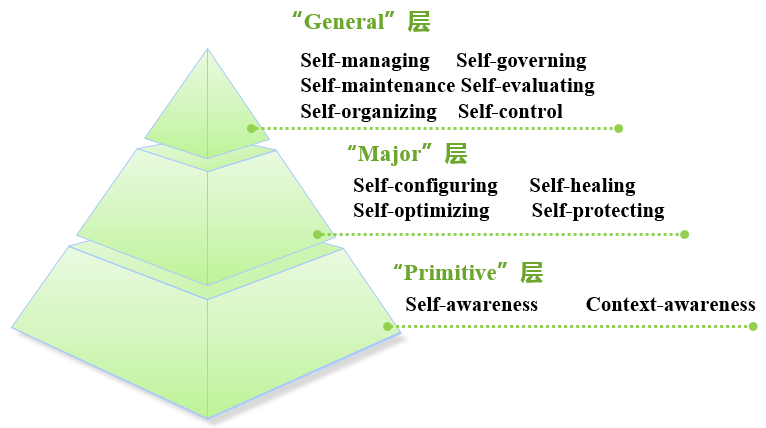


图14 自适应策略分层

## 2.3基础理论

表8 MAPE对应情况

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **time** | **reason** | **level** | **technique** |
| Monitoring | 持续进行 | 感知的对象 | 识别层次 |  |
| Analyzing | 判断采用主动方法或反射方法 | 分析的对象 |  |  |
| Planning |  | 需考虑的因素 | 待调整的层次 | 具体策略 |
| Executing |  |  | 在这些层次上执行调整 | 具体执行 |



图15 内部控制

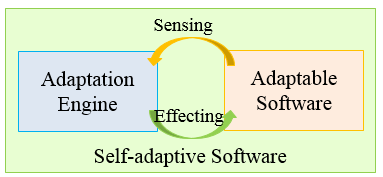


图16 外部循环

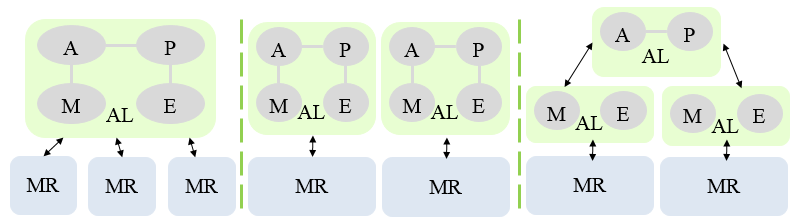


图17 MAPE分类

## 2.4研究现状

表9 工业项目

|  |  |  |  |
| --- | --- | --- | --- |
| **项目** | **内容** | **Self属性** | **MAPE** |
| KX | 运行时刻收集应用行为数据并进行适应 | 自配置 | 感知 |
| Accord | 提供编程框架，以规则指导适应过程 | 自配置 | 决策 |
| ROC | 建立递归恢复过程，支持系统撤销行为 | 自愈 | 执行 |
| TRAP | 利用AOP和反射机制建立动态应用 | 自配置 | 执行 |
| K-component | 建立基于构件交互的系统行为，并适应 | 自配置 | 决策 |
| Self-adaptive | 利用模型检测，修改软件体系结构 | 自配置，自愈 | 分析 |
| CASA | 基于合同，支持应用层和底层适应行为 | 自配置，自优化 | 执行 |
| J3 | 模型驱动的应用层适应框架 | 自优化 | 分析 |
| DEAS | 识别变化，分析系统行为，实现目标 | 自配置 | 决策 |
| MADAM | 针对移动计算，建立体系运行态模型 | 自配置，自优化 | 感知 |
| M-Ware | 建立中间件，实时感知资源变化 | 自配置，自优化 | 感知 |
| ML-IDS | 检测网络攻击，实施操作系统保护 | 自保护 | 分析 |

通过详细阅读2014年至2015年相关SCI期刊或IEEE、Springer、ACM与Elsevier出版的文章，共计54篇，以及3篇关于自适应软件的综述文章，整理了目前国际团队在该领域的研究现状与研究方向。

* 按研究方向分

表2 主要研究方向-按研究方向分

|  |  |  |
| --- | --- | --- |
| 方向 | 篇数 | 说明 |
| 通用 | 47 | 不局限于特定领域的研究方法 |
| 移动计算  领域 | 1 | 考虑其组成元素的多变和网络环境的复杂，要考虑自适应系统应对这些复杂变化时的健壮性 |
| 高性能计算领域 | 1 | 针对百亿规模以上的系统，考虑其性能优化、能耗降低、地域弹性等问题，考虑这种规模下软件的自适应反馈策略 |
| 智能交通 | 1 | 使车载系统能够自适应得应对各种车辆驾驶过程中的各种突发情况 |
| 遗产系统 | 1 | 利用自适应实现遗产系统的改造，实现软件重用 |
| 嵌入式领域 | 2 | 由于嵌入式系统代码不易获取，且不易修改，因此利用MAPE-K方法时，需要改进 |
| 云计算 | 1 | 建立云环境下的自管理软件，对云资源进行高效处理，基本是采用了MAPE方式 |

通用的方法研究最多，之前我们主要涉及到的是针对遗产系统的研究。然而随着网络技术的发展，与云计算、物联网相关的领域也开始成为自适应研究的对象。

* 按研究对象分

表3 主要研究方向-按研究对象分

|  |  |  |
| --- | --- | --- |
| 方向 | 比例 | 说明 |
| 需求工程 | 3 | 关于非功能性需求的研究，普遍采用目标驱动的需求工程方法 |
| 系统行为 | 1 | 对系统的自适应行为进行建模，从高层角度讨论自适应策略的实施 |
| 环境建模 | 1 | 主要采用形式化的方法，量化环境的改变情况，以预测环境变化对性能的影响 |
| 建模工具 | 3 | 利用UML扩展元素，定义自适应系统 |
| 逻辑建模 | 2 | 区分软件自适应逻辑和应用逻辑，对演化逻辑的变化进行研究 |
| 知识建模 | 2 | 对自适应系统中的知识进行建模，以提供自适应策略的推理 |
| 系统建模 | 10 | 采用模型驱动、SOA、软件体系结构等方法对系统进行建模 |
| 控制策略 | 31 | 研究实现自适应的各种控制策略 |
| 人类参与 | 2 | 将人对自适应系统的影响考虑到MAPE过程中，以解决对复杂环境的感知或自适应方案的决策 |

当前，随着人脑融合的研究日渐成熟，人类的干预又被重新引入了自适应循化中。这可以说是目前最新的研究点，但是并不适合我们继续深入。自适应控制策略的研究，与其他建模方法密不可分，之前我们对环境、触发事件等，仅采用元组的形式描述，这种描述方法太简单，也导致了无法利用这些信息进行推理，以支持自适应演化，如果能够继续深入下去，则自适应逻辑、知识、环境等应该采用统一的一种形式化定义方法，以实现高层的知识推理。

* 按研究方法分

表4 主要研究方向-按研究方法分

|  |  |  |
| --- | --- | --- |
| 方向 | 比例 | 说明 |
| 控制论 | | |
| 控制论 | 8 | 以反馈循环管理软件系统 |
| 软件工程方法 | | |
| 模型驱动的  软件工程 | 4 | 采用建立各个阶段的模型，驱动软件的研发与自适应 |
| 中间件 | 2 | 在传统MAPE-K循环的基础上，研究非功能与功能需要的组件自组织策略 |
| SOA | 3 | 利用服务封装系统中待调整的软件单元，通过结合控制论，实现软件单元的重新组合 |
| 需求驱动的  软件工程 | 4 | 利用案例推理，对过去历史方案进行存储和总结 |
| 软件产品线 | 2 | 实现从需求到代码层的大规模重用 |
| 软件体系结构 | 4 | 采用基于策略的方法，从整体角度重塑软件 |
| 生物启发 | | |
| 基于搜索的  软件工程 | 2 | 将非功能性需求看作是多个目标，采用NSGAII多目标优化算法解决问题 |
| 生物进化策略 | 1 | 利用达尔文生物进化策略，采用遗传算法，演化系统 |
| 基于策略的方法 | | |
| 代数图 | 1 | 利用基于代数图的迁移策略，形式化建模系统 |
| Markov  决策过程 | 3 | 属于基于策略的方法，以离散状态Markov链建模自适应软件系统，从而以概率的方式，决策系统的调整方案 |
| 概率模型 | 1 | 以概率模型的方式，建模系统 |
| Petri网 | 1 | 利用基于人工神经网络的Petri网，实现对系统的需求、行为等建模 |
| 博弈论 | 1 | 利用随机化多人博弈，建模各种人类行为对软件系统的影响 |

控制论、模型驱动和软件体系结构是目前研究自适应系统最常用的三个方法，利用Agent技术，从AOSE角度出发研究自适应软件系统仍然可以被称为是比较新颖的方法。

而基于策略的方法中，使用数学方法，形式化建模系统，实现自适应逻辑的推理方法，有助于提升方法的档次与可信度，是日后需要考虑引入的一个方向。

## 2.5主要方法

### 软件产品线

* Gomaa提出了一个可配置、演化的产品线生命周期
* Hallsteinsen利用models@run.time实现了动态产品线

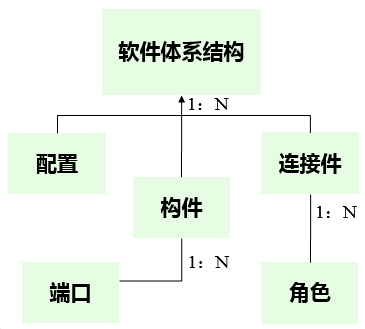
**基于构件或面向概念**

* MUSIC提供基于服务集成和构件组合的开发方法
* 设计模型描述构件及其关系；运行模型调整系统结构

**元模型**

* Vogel利用megamodels总结不同运行时模型，产生资源和控制系统间的模型转换引擎
* Lehmann提出了运行时模型的元模型

### 软件体系结构

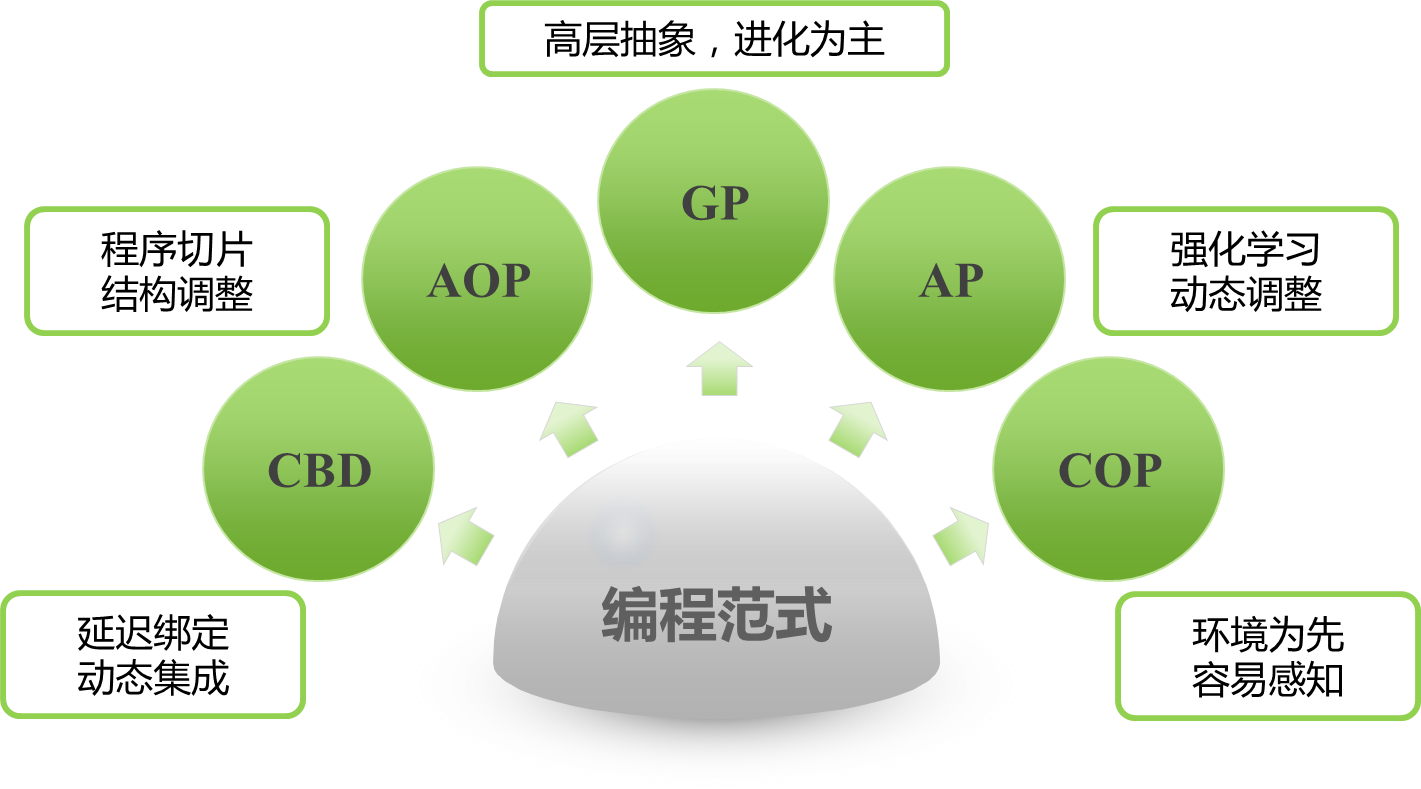


Rainbow框架外部控制体系层系统层结构调整参数调整推理变化；**Agent技术控制调整Rainbow，Manager，Arch-studio**。

### 反射方法

内省introspection对内省结果的反应，调解intercession对自我行为的感知。





### 控制论





### 面向服务的方法

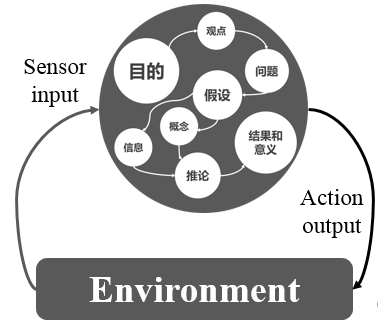
**服务更新**更新服务加载更优服务；**协作更新**服务组合组织调整

**MUSIC**利用MDE技术开发了基于服务的SAS框架模型

**SASSY**利用模型驱动技术建立SAS的基础服务包括：目录，QoS服务，适应服务

**结合其他技术**：元数据，MetaSelf，构件模型，AOP，需求工程，CARE，MAS，QoS，MOSES。

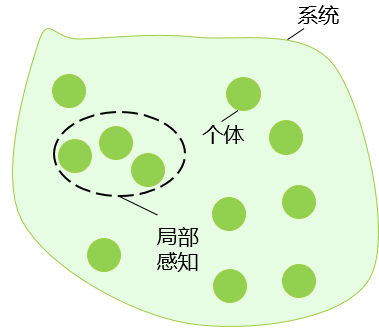
### 多Agent方法

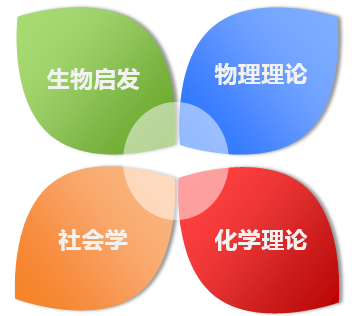




多Agent系统理论，Self-organization，De Wolf 建立具有适应性的MAS的设计模式；Unity，Tesauro 提出的分散式体系结构，可实现目标驱动，自组装，自愈，自优化等行为；AMAS，Bernon 提出的基于多Agent系统和Agent协作实现自组织；结合其他技术，SOA，结合SOA建立自适应系统；CBD，结合CBD将自主Agent设计为构件；对象模型，提出基于Agent结构的SAS设计模式。

### 自然激励方法





**社会学**，市场理论，拍卖理论，社会规范，多Agent协作；**化学理论**，化学反应，服务组合，延迟绑定，元组空间，系统协调；**生物启发**，生物行为，MAS的集体行为分析，生物行为例如群集，觅食，趋药性等已用于自组织；**物理理论**，势能场理论，路径规划中的障碍规避，机器人自主控制，自适应任务分配。

### 形式化建模与检查

**检查 Verification，**Self特性和动态性导致保证SAS的行为和结构的正确性特别是对于安全攸关的领域；**形式化建模 Formal modeling**，建立系统的形式化模型，辅助开展行为检查与结构验证。

Ortmeier提出了Restore Invariant Approach (RIA)**；**Nafz提出了Formal Reference Model for SAS**；**King提出了一个针对自主计算系统的自测试框架**；**Smith提出了自顶向下的形式化方法建立自组织系统**；**Autonomic System Specification Language可形式化描述和产生自主系统**；**Cordy使用模型检测的方法，基于转换自动机动态检测软件产品线**；**Filieri 利用离散时间的markov链进行运行时的概率模型检测。

### 学习方法



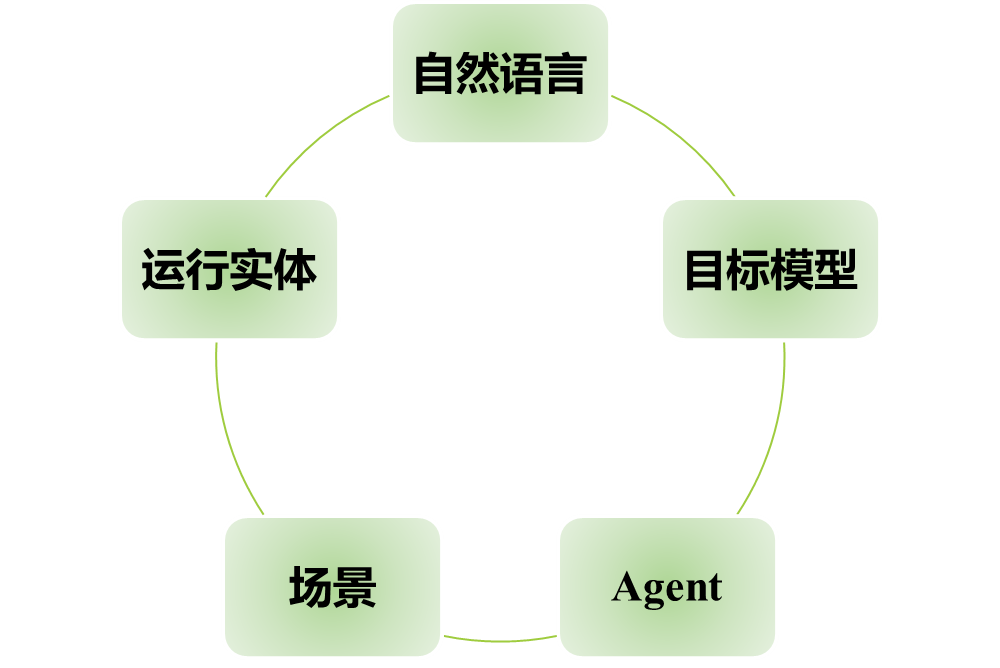


**结构：**协作式强化学习**，**进化编程**，**观察/收集模式；**参数：**进化算法**，**模型学习**；算法，**进化编程**，**基于人工智能的学习技术。

### 需求工程

* 开发人员给出系统和反应的一般定义
* 为实现适应，运行时刻采用需求工程技术
* 辅助开发人员给出关于自适应机制的决策
* 评估自适应机制

**需求描述语言**

****

自然语言RELAX可处理不确定性采用自然语言描述需求；KAOS，i\*，LoREM，FLAGS，CARE，基于目标模型，针对SAS的需求描述语言；Tropos4AS基于Agent技术建模SAS的需求；Live sequence charts 基于场景的描述语言；Runtime Entities将需求建模为运行时的实体，并集成对需求的反射能力。

### 其他方法

**基于任务，**根据用户任务的特点制定自适应策略，自适应机制不是预先定义的，而需要根据用户任务的情况推理产生，建立自适应中间件，支持结构层调整。

**中间件为中心**，简化设计，建立构件库以重用构件，建立设计空间，聚类设计方案，应用集成。

## 2.6对比分析

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **方法** | **MAPE** | **Time** | **Reason** | **Level** | **Tec** | **App** | **DC** | **DDec** |
| 模型 | All | React | Ctx/TR/U | App/Com  /TR | Par/Str | Ext | G/P/M/U | H/C |
| 体系结构 | ALL | React | Ctx/TR | App/TR | Par/Str | Ext | G/P/U | H/C/D |
| 反射 | ALL | React | Ctx/TR/U | App/Com/  TR/Sys | Par/Str | Both | P/M | H/D |
| 程序范式 | P/E | React | Ctx/TR | App/TR | Str | Ext | G/P/M/U | H/C/D |
| 控制论 | ALL | React | Ctx/TR | Sys/Com | Par/Ctx | Ext | G/P/M | D |
| 服务 | P/E/ALL | React | Ctx/TR/U | App | Str | Ext | G/P/M/U | H/C/D |
| Agent | P/ALL | React | Ctx | App/Sys | Par | Both | P/M | H/D |
| 自然启发 | P | React | Ctx | App/Sys | Par/Str | Int | **U** | H/D |
| 形式化 | A/P/ALL | React | Ctx/TR | Sys/TR | Par/Str | Ext | **M** | D |
| 学习 | P | React | Ctx/TR | App/Sys | Par | Ext | G/M/U | H/C/D |
| 需求工程 | A/P/ALL | React | U | App | Par | Ext | **G** | H/C/D |
| 其他 | A/P/ALL | React | Ctx/TR | App/TR | Par/Str | Ext | G/U | H/D |

### 缺陷分析

**上下文适应**，可被自适应逻辑控制的，上下文环境。

**循环分离**，实现自适应逻辑，即MAPE各环节的分离。

**主动适应**，主动调整，阻止不期望的情景发生。

### 未来挑战

将用户纳入自适应控制循环，建立集成其他模型的运行时模型

处理不确定因素的Verification & Validation

将设计阶段的成果转化到运行阶段

自适应过程的重用

# 多目标自适应问题

越来越多的软件系统已运行在互联网等动态、多变、开放、难以控制的新环境。这种新环境的分布性、软件及设备的多重异构性、用户需求的个性化及多变性等特点，使软件的组成逐步由固定转向多样，软件的行为逐步由静态转向动态，软件的开发逐步由封闭转向开放。运行在新环境下的复杂软件系统逐渐呈现出资源异构、动态开放、持续构造等新特征。1）资源异构：随着无线传感技术的迅速发展，各类超薄化、小体积的新型设备逐渐产生。该类软件系统不仅拥有服务器、客户机等传统计算资源，同时也拥有智能手机、平板电脑、可穿戴设备、跨界智能终端等地域分散的移动化计算资源。2）动态开放：由于计算节点的高度自主性和各节点业务功能的相对独立性，该类软件系统一般采用开放式的体系结构以便于集成不同的计算节点。这也导致了软件内部计算节点、节点间交互关系与组织结构的动态多变性。3）持续构造：巨大的规模、多变的结构导致复杂软件不可能一次性完成设计、开发和部署。在时间尺度上，该类软件持续开发、部署、更新和调整的过程可能长达数年甚至数十年。

具有新特征的复杂软件系统在运行过程中需要适应更加复杂的软件变化，如图1所示。1）异构的计算资源、多变的运行环境、动态的计算节点、用户的个性需求导致了软件变化“来源多样”。需求的变化可能影响软件整体业务流程，而计算资源的损坏可能影响某个节点的对外功能，因此不同软件变化的“影响各异”。 2）用户需求、系统结构以及计算资源的变化可能会“同时产生”。当调整同一计算资源以适应不同软件变化时，极有可能产生“互相冲突”的自适应策略。3）动态开放和持续构造等软件特征导致新的软件单元、计算资源可以自主动态地加入到系统运行过程中，使得软件系统“不可预测”这些新单元或资源的所有变化，“不可确定”所有相应的自适应策略。



图1 建立软件自适应性面临的问题

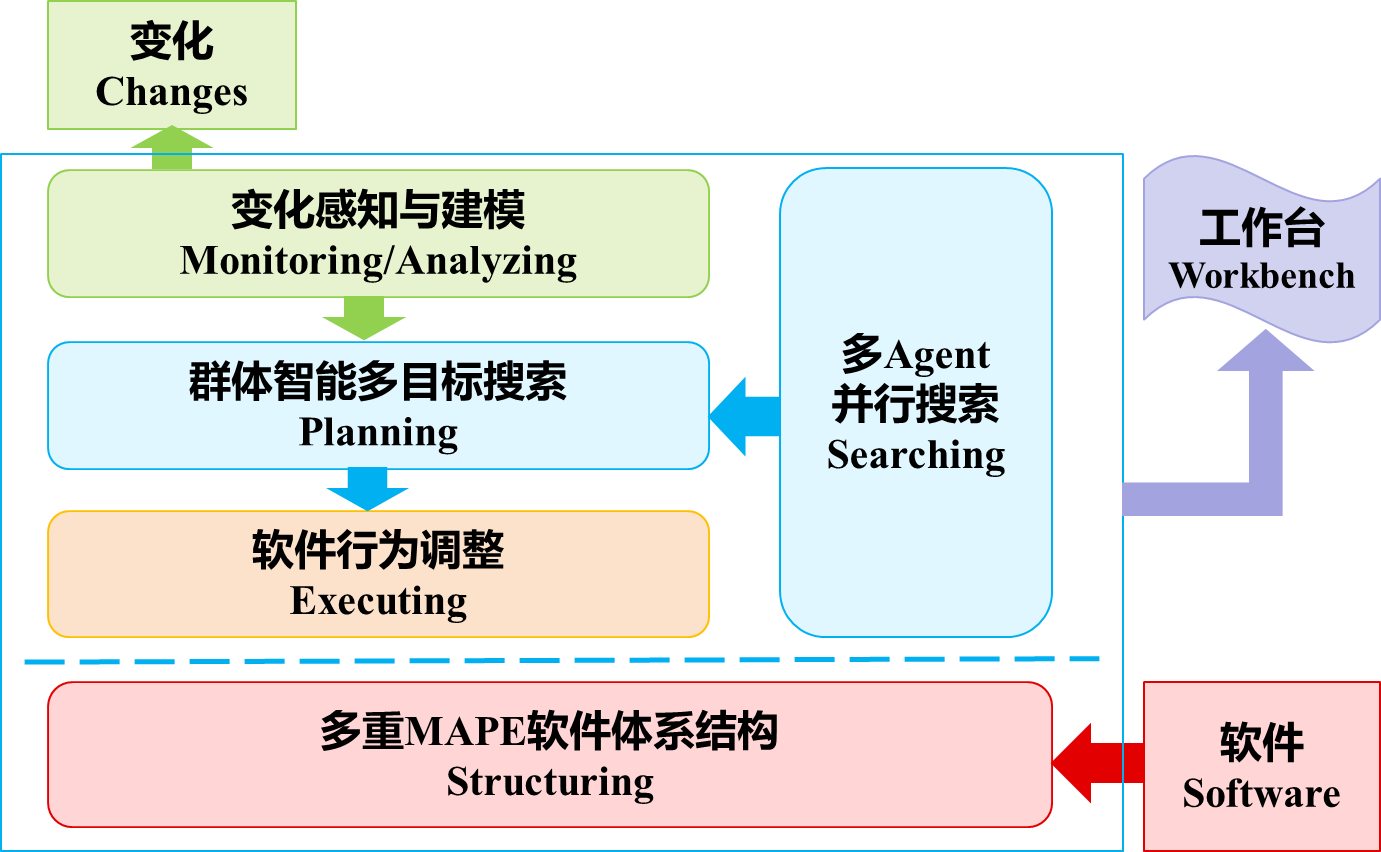
因此，为适应具有上述特点的软件变化，资源异构、动态开放、持续构造的复杂软件系统其自适应性需解决“多维调整、策略冲突、在线决策”三方面的问题。1）如何根据不同变化的影响范围，对软件系统整体或某个计算资源进行多维调整，以适应“来源多样、影响各异”的软件变化。2）如何同时处理多个软件变化，并调解不同自适应策略之间的冲突，以适应“同时发生、互相冲突”的软件变化。3）如何在不终止系统运行的前提下根据变化信息在线决策自适应策略，以适应“不可确定、不可预见”的软件变化。

根据基于搜索的软件工程的思想，不同软件变化可被建模为不同的优化目标。因此，软件的自适应问题将被映射为多目标优化问题（Multi-objective Optimization Problem，MOP）。适应不同软件变化的自适应策略将构成一个解空间。在运行过程，软件系统可通过搜索算法在解空间中选择出最优的自适应策略以调整自身行为。并且，与马尔科夫决策算法、博弈论、Petri网等技术不同，搜索算法无需关于软件变化的大量上下文信息，并且对噪声数据、错误数据有明显的容错能力，更加适用于软件自适应决策。然而，就文献分析所知，目前采用基于搜索的软件工程技术解决软件自适应问题的研究相对较少，针对该方向的方法研究和技术突破具有一定的前瞻性和创新性。

智能体（Agent）来源于分布式人工智能领域（Distributed Artificial Intelligence，DAI）是指能够完成自主完成特定任务并通过协作完成共同任务的软件。多个Agent间通过共享共同的目标进行通信和协作，将形成多Agent系统（Multi-Agent System，MAS），这与目前基于搜索的软件工程领域的粒子群、蚁群等群体智能优化算法的设计理念一致[Giacomo Cabri 05]。并且，结合多Agent系统所具有的“并行处理、智能自治”等优势，可有效解决基于搜索的软件工程技术存在的搜索效率和搜索开销问题。1）搜索效率：当需要搜索的解空间非常庞大时，如何提高搜索效率是必须解决的问题之一。Agent采用基于目标驱动的行为控制，其建模层次比一般的构件和服务更加高层抽象。Agent之间的交互信息不再是一般的数据或控制指令，而是高层的目标信息、搜索结果等，为实现对状态空间的并行搜索提供了可能。2）搜索开销：基于搜索的软件工程技术的时间和空间开销比较大。Agent的自治能力、对变化的感知能力、以及对未知情况灵活处理的智能性等特征，使Agent本身就可处理部分软件变化，避免了大量采用搜索方法而产生的时空开销。因此，将智能体技术与基于搜索的软件工程技术相结合能够有效的解决复杂软件系统自适应性面临的问题。

本项目拟建立基于多智能体并行搜索的自适应软件建模方法与运行机制。该机制封装软件单元为Agent，建立Agent结构及Agent间组织关系，以Agent感知（Monitor）同一时刻发生的软件变化，分析（Analyze）变化信息并建立搜索目标，利用Agent在自适应策略空间中并行搜索，决策（Plan）出最优自适应策略，并执行（Execute）该策略以调整软件系统行为。基于多智能体并行搜索的自适应软件建模方法与运行机制可有效解决复杂软件系统自适应性面临的问题。

首先，该机制可解决多维调整问题。依据Agent的自治性特点，Agent可依据既定目标实现对自身功能的智能调整。依据Agent社会性和协作性等特点，各Agent可通过协作协商实现对Agent间组织关系与协作关系的调整。其次，该机制通过基于搜索的软件工程技术可解决策略冲突问题。该机制可感知同时发生的不同软件变化，并采用基于搜索的软件工程技术，将每个软件变化映射为一个优化目标，利用多目标优化技术实现对“同时发生、互相冲突”的软件变化的权衡处理，从而制定出满足各个目标的最优自适应策略。并且，该建模方法与运行机制从如下四个方面解决在线决策问题：①有别于离线决策中静态绑定软件变化与自适应策略的方式，该机制通过依据变化动态搜索自适应策略的方式，实现了变化和策略间的动态绑定。②该运行机制支持在系统运行过程中在线扩展自适应策略空间以应对未知变化。③与离线决策在系统运行中无法更新自适应策略不同，该机制可在运行中动态组合现有策略，产生出数量庞大的新自适应策略，以更好地应对未知变化。④采用智能体技术实现对自适应策略的并行搜索，以保证在系统运行过程中实时产生自适应策略。



# 总结与展望



本次学术论坛主要介绍了基于搜索的软件工程与自适应系统的研究，分别从基于搜索的软件工程的研究意义，主要方法，研究问题，未来研究；及自适应软件系统的核心要素，基础理论，基本定义，主要方法等进行了介绍，给出了基于群体智能搜索的多目标自适应决策方法。

今后将继续沿着该方向深入研究，并早日在系统中践行目前的设计方案。