****

经 验 软 件 工 程

近年来软件体系结构取得的成绩和面临的挑战

小 组 成 员 陈鸿超（SY1806214）

刘 颖（SY1806418）

目录

[1 前言 1](#_Toc8666898)

[2 调研方法 2](#_Toc8666899)

[3 调研结果 3](#_Toc8666900)

[3.1 软件体系结构描述研究 3](#_Toc8666901)

[3.2 软件体系结构设计研究 3](#_Toc8666902)

[3.3 软件体系结构模型研究 4](#_Toc8666903)

[3.4 软件体系结构分析与评估研究 5](#_Toc8666904)

[3.5 软件体系结构验证研究 5](#_Toc8666905)

[3.6 软件体系结构恢复研究 6](#_Toc8666906)

[3.7 软件体系结构重构研究 6](#_Toc8666907)

[3.8 软件体系结构面临的挑战 6](#_Toc8666908)

[3.8.1 软件体系结构的描述 6](#_Toc8666909)

[3.8.2 软件体系结构的记录 6](#_Toc8666910)

[3.8.3 软件体系结构的侵蚀 7](#_Toc8666911)

[3.8.4 软件体系结构的恢复 7](#_Toc8666912)

[4 结果分析 8](#_Toc8666913)

[5 调研结论 9](#_Toc8666914)

[6 参考文献 10](#_Toc8666915)

# 前言

软件体系结构是一系列抽象的概念，是软件项目的开发蓝图，是对软件设计层次的再一次提示。基于软件体系结构的软件开发方法是满足当今软件开发所面临的众多苛刻要求的必经途径，逐渐被广泛采纳，在软件工程领域有着广泛的应用。许多专家学者从不同的角度和不同的侧面对软件体系结构进行了刻画，对软件体系结构的定义、描述方式、记录方式、设计模式、评估验证等各方面都进行了大量的研究，但是至今还没有一套被大家所公认的规范。因此，本文对近年来软件体系结构方面取得的各项成绩和依旧面临的挑战进行了广泛的调研，以促进对软件体系结构的理解与研究。

# 调研方法

本文采用的是文献调研方法，首先在相关网站上对软件体系结构相关的论文进行了检索与查阅；然后根据论文摘要对论文进行了进一步的筛选，并整理和汇总了各选中论文的主要内容；最终对调研结果进行了讨论与分析，得出结论。

本次调研所查阅的论文主要来自IEEE和ACM的会议&期刊杂志、软件学报、中国科学，发表时间主要集中在近两年，都是比较新的研究成果。论文的主题主要是与软件体系结构相关的理论研究、方法改进与创新等，但不包括教程以及具体的应用实例。

具体的论文检索与筛选方式如表2.1所示。

表格 2.1 论文检索与筛选

|  |  |
| --- | --- |
| 论文检索网站 | 1. https://ieeexplore.ieee.org/Xplore/home.jsp  2. https://dblp.uni-trier.de/  3. http://www.jos.org.cn/jos/ch/reader/key\_query.aspx  4. http://www.cnki.net/ |
| 检索项 | 主题、名称、关键词 |
| 检索关键词 | 软件体系结构、Software Architecture |
| 筛选准则 | 1. 与软件体系结构相关的研究  2. 非体系结构工具或设计教程  3. 非某领域或系统的体系结构设计和分析  4. 非综述类文章  5. 发表时间较近 |

# 调研结果

本次调研共选取了21篇论文，涉及到了软件体系结构的描述、设计、模型、分析与评估、验证、恢复、重构等方面，以下将详细介绍每个方面的调研结果。

## 软件体系结构描述研究

软件体系结构是一个非常抽象的概念，目前仍然没有一个统一且完善的标准进行描述。2017年，Yuan E[1]提出了一种借助Web脚本语言OWL来描述软件体系结构中元数据的表示、交互等信息的框架，它不仅可以用于体系结构的描述和维护，还可以用于体系结构关键点的自动推理和分析，有利于整个软件行业标准的统一。

## 软件体系结构设计研究

软件体系结构的设计需要软件相关的各方人员不断沟通、讨论和尝试，最终才能确定的，是一个长期且复杂的任务，高度依赖于人类的决策。

Wang Q[2]在2017年提出了以元数据驱动的体系结构设计模式，该模式可以有效地减少业务与软件实现之间的耦合关系，提高设计的灵活性，提高系统的可重用性，摆脱了因业务的变化导致软件重组的问题。该文研究了元数据驱动设计体系结构的方法，分析了元数据驱动的本质，给出了元数据驱动设计体系结构的软件实现模型，对广泛的软件工程体系结构实践具有指导意义。Ali N[3]等人提出了一个自适应的体系结构设计过程，系统地定义了用于开发自适应体系结构的那些活动，以满足不断变化的软件需求。该过程引入了一个可变性监控代理来分析可变性模型中的变化，决定是否根据变化重新配置体系结构。

为了方便和辅助设计师的工作，Van der Werf[4]在2017年提出了一种用于捕获和记录软件体系结构设计过程工具Video Wall的设想。该工具作为讨论的白板，能够自动记录讨论过程中设计的各种方案，以生成体系结构的基本文档，减轻设计人员的工作。Gopalakrishnan A[5]提出了一种软件体系结构设计领域的智能决策支持系统，为开发人员提供明智的决策建议，克服自身的局限性，提高开发人员设计速度与创新能力。同时，该系统还可以不断的学习和完善。

在软件体系结构的设计过程中，由于人类思维认知的局限性，架构设计师不可避免的会受到认知偏差的影响，做出一些不合理的决策。Manjunath A[6]对体系结构设计决策过程中的认知偏差问题进行了详细的研究，详细记录了33个重要的认知偏差，以帮助架构设计师更好的避免这些偏差。Pretorius C [7]则提出了一种比较新颖的想法，作者提出在进行软件体系结构的设计阶段，设计师往往会倾向于做出比较理性的选择。但是在设计过程的某些阶段使用直觉会产生更多的创意，并可能在设计活动中产生更好更快的决策。因此，作者认为在体系结构的设计时，应该将直觉与理性相结合，并给出了相应的案例。

## 软件体系结构模型研究

研究软件体系结构的首要问题是如何表示软件体系结构，即如何对软件体系结构建模，根据建模的侧重点不同，我们对软件体系结构模型进行了以下调研。

首先是体系结构模型正确性验证的研究。根据统计，软件测试中发现的70%以上的错误是由需求获取或者体系结构设计引起的。因此，软件体系结构在设计阶段的正确性验证很重要，乌尼日其其格等人[8]提出了一种高阶类型化的软件体系结构建模和验证语言，同时还提供了体系结构建模和验证方法，并通过一个实际案例完成了较大规模软件体系结构的体系结构建模和验证。

随着计算机硬件能力的不断提高以及计算机软件的规模化，软件的复杂性也在不断增加，并成为限制软件质量和生产力提高的关键因素之一。为了解决这个问题，Hong Mei等人[9]提出了一种控制软件设计的高级结构复杂性的模型，并提出了一种以软件体系结构为中心的软件开发方法。

最后，我们调研了运行时软件体系结构建模的相关研究。运行时体系结构是系统运行时刻的一个动态、结构化的抽象，描述系统当前的组成成分、各成分的状态和配置以及不同成分之间的关系。运行时体系结构允许开发者以读写体系结构的方式实现系统的检测和调整。构造运行时体系结构的关键是针对不同的目标系统和体系结构风格实现合适的基础设施，以维护二者之间的因果关联。宋晖等人[10]提出了一种模型驱动的运行时体系结构构造方法，开发者只需针对目标系统、体系结构以及两者之间的关系分别建模，根据这些模型，支撑框架自动构造合法而高效的运行时体系结构基础设施。该方法具有广泛的适用性，并显著提高了运行时体系结构构造过程的效率和可复用性。

## 软件体系结构分析与评估研究

对软件体系结构进行分析和评估可以帮助相关人员理解软件的架构，确定软件的可行性和可靠性，是软件开发流程中重要的一环。

在分析方面，Altoyan N [11]提出了一种使用半形式化的软件体系结构描述语言ADL和基于约束的关系模型构建的软件架构的框架，可以用于判定一个软件体系结构的性能。针对软件项目中的文档，Shumaiev K等人[12]使用自然语言处理技术对件体系结构文档中不确定性因素进行自动检测，并通过对实例的人工分析，证明了该设想对于体系结构分析的价值，并说明了可能会遇到的困难。陈湘萍等人[13]对软件体系结构分析过程中得到的各种分析结果又进行了一次集成分析，提出了一种结果集成框架，弥补了之前软件分析结果对集成度关注不足的缺陷。

在体系结构的评估方面，仿真模拟是一个重要的手段。但是由于体系结构描述的不规范性与架构仿真正确执行的形式化要求，使得架构仿真对工程师而言仍然十分困难。因此，Bogado V[14]提出了一种自动化评估框架，工程师只需在前端提交软件架构的UCM(Use Case Map)，后端就会自动生成相应的仿真模型进行评估，同时也支持工程师自定义UCM与仿真模型的对应关系。同时，Antonino P O [15]也提出一种半自动化方法实现对体系结构的仿真模拟，该方法缩小了体系结构与仿真模型之间的差距。除了仿真方面的研究之外，Zhang C等人[16]在现有的可靠性评估模型的基础上增加了与时间相关的可靠性分析，同时使用状态图方法，建立起部件状态与系统可达状态之间的状态转移矩阵，并通过具体实验证明了本方法的有效性和准确性。

## 软件体系结构验证研究

软件体系结构验证方面的研究主要针对的是一致性验证，即软件具体的实现与设计的体系结构是否一致的问题。

在软件体系结构的一致性进行验证领域，一般都是对软件源代码进行分析，很少有研究涉及到代码的注释信息。因此，Bandara V[17]等人开发了一种从代码注释中提取信息验证体系结构一致性的方法，并通过实验证明，该方法可以有效的减轻体系结构的侵蚀。而在自动化方面，Olsson T等人[18]实现了一种针对MVC模式软件的自动一致性检查工具，该工具基于启发式的方法实现，会在源码的每次更新时生成一个检查报告，帮助开发人员确保软件的一致性。

## 软件体系结构恢复研究

对于很多的软件而言，保持其体系结构文档的实时更新是一件非常困难的事，因此很多的软件都没有相应的体系结构文档或者文档已经过时。这时，对软件的体系结构进行恢复就是一件非常有必要的工作。Lutellier T[19]等人在现有的恢复技术基础上，研究了使用准确的符号(函数名或变量名)依赖性进行软件体系结构恢复对恢复结果准确性的影响，并通过与现有的九种恢复技术进行比较，证明了该方法的有效性和普适性。

## 软件体系结构重构研究

由于软件需求的变更，原有的体系结构可能已经无法满足要求，这时就需要对软件的体系结构进行重新设计。Pretorius C 等人[20]介绍了一种由性能驱动的软件体系结构自动化重构技术，该技术会通过性能分析模型计算当前体系结构与备选方案中体系结构的性能值，从中选择最优的体系结构。Arcelli D 等人[21]提出了一种更简单的、基于性能的多目标软件体系结构重构的进化方法，该方法更关注架构优化的可操作方面，而不仅仅是搜索一组备选方案，具有更好的有效性和适用性。

## 软件体系结构面临的挑战

通过对这些论文的整理与分析，本节整理了软件体系结构领域依旧存在的一些挑战。

### 软件体系结构的描述

软件体系结构是一个抽象的概念，对于相关人员来说很难表示、理解、沟通与评估。虽然软件行业在标准化体系结构描述和建模语言方面取得了巨大进展，但对于实际的开发者而言，描述体系结构仍然是一个复杂、困难并且成本高昂的工作。如何更好的描述软件体系结构，仍然是软件体系结构领域亟待解决的问题。

### 软件体系结构的记录

软件体系结构是软件相关人员之间进行沟通的主要工具，清晰和完整的描述软件的体系结构对于体系结构的分析、项目的管理、新成员的引入都有很大的促进作用。在现有的描述方式下，表达软件体系结构的主要方法是使用自然语言文本(软件体系结构文档)的形式对其进行记录。尽管已经提出了许多辅助方法和形式化方法，但是维护软件体系结构文档的过程仍然很费时费力。

### 软件体系结构的侵蚀

软件体系结构侵蚀指的是实现的体系结构与预期设计的体系结构不符的情况，是当今软件开发行业面临的主要问题之一。在软件的开发过程中，导致软件体系结构侵蚀的原因有很多，比如开发人员没有按照设计实现、对体系结构了解不足、体系结构文档未实时更新等。而从源代码层次判断软件的实现是否满足预期的设计又十分困难，因此，至今仍未有一种合适的方法来避免这种问题。

### 软件体系结构的恢复

软件体系结构对于程序员理解和维护软件至关重要，但是由开发人员去记录并实时维护软件的体系结构，代价过于昂贵并且十分困难。因此，对于许多软件项目而言，文档化的软件体系结构要么不存在，要么已经过时。因此，如何有效的恢复和获取软件的体系结构是一个重要的挑战，现有的恢复技术还存在很大的提升空间。

# 结果分析

本次调研的论文分布情况如表4.1所示。

表格 4.1论文分布

|  |  |
| --- | --- |
| 研究领域 | 论文篇数 |
| 软件体系结构描述 | 1 |
| 软件体系结构设计 | 6 |
| 软件体系结构模型 | 3 |
| 软件体系结构分析与评估 | 6 |
| 软件体系结构验证 | 2 |
| 软件体系结构恢复 | 1 |
| 软件体系结构重构 | 2 |

从本文调研论文的分布来看，目前学术界在软件体系结构的设计与分析评估的方向研究偏多，这两类涵盖的文章占到调研文章的半数之多；而软件体系结构的描述与恢复则比较冷门，分别只有一篇论文。

从每个细分领域来看，目前软件体系结构在不同子领域的进展和侧重点各有不同：

在体系结构的描述领域，目前的研究仍然着眼于寻找一种简单完善的统一规范，促进整个软件行业的发展；在体系结构的设计方面，研究者致力于平衡人的活动对软件体系结构设计造成的影响，包括过程管理工具，常见的认知偏差与人机合作智能决策等；在软件结构的建模领域，研究内容集中在不同问题情景下软件体系结构建模语言的创造与开发上；在体系结构分析与评估领域，目前的研究方向主要集中在自动化分析与评估方法的设计以及分析与评估方法的改进；在体系结构验证领域，目前的研究集中在体系结构一致性的检查，以评估与减轻软件体系结构的侵蚀问题；在体系结构重构领域，目前的研究都着眼于性能驱动的体系结构自动化重构技术；而目前软件体系结构的恢复仍然出于起步阶段，主要的困难集中于软件体系结构文档的不规范管理、高维护成本、设计的抽象性与实现设计不一致的问题。

# 调研结论

目前，制约软件体系结构发展的主要原因在于体系结构概念的抽象性，至今仍然没有一个被学界公认的统一定义，使得体系结构各子领域发展或多或少的受到了一定的限制。

比如在体系结构的描述领域，虽然发展了这么多年，但由于体系结构的抽象性和定义的多样性，仍然没有一种统一的描述规范，并且现有的描述方法在实际应用中仍然十分复杂、困难并且成本高昂。也正是因为体系结构描述领域的困境，使得软件体系结构文档的撰写与实时维护成为了一项难以完成的工作，进一步的影响到了体系结构的设计、实现、评估、验证、恢复等各领域。

而目前的主要研究方向，一方面是针对各种现实问题情景，对各领域已有的方法进行完善与改进，比如新的设计模式、评估模型，改进后的建模方法、验证技术和恢复技术等；另一方面就是将自动化技术应用到体系结构的各领域，减轻相关人员的工作，为开发人员提供决策建议。这些成果都促进了软件体系结构的进一步发展与应用，但是在定义与描述这些基本领域，仍需要更多的研究与投入，以促进整个软件行业统一规范。

# 参考文献

[1] Yuan E. Towards ontology-based software architecture representations[C]//2017 IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE). IEEE, 2017: 21-27.

[2] Wang Q, Liu N, Zhang Z, et al. Architecture methodology researchment of metadata driven design[C]//2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN). IEEE, 2017: 1388-1392.

[3] Ali N, Hong J E. Creating adaptive software architecture dynamically for recurring new requirements[C]//2017 International Conference on Open Source Systems & Technologies (ICOSST). IEEE, 2017: 67-72.

[4] van der Werf J M E M, de Feijter R, Bex F, et al. Facilitating Collaborative Decision Making with the Software Architecture Video Wall[C]//2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017: 137-140.

[5] Gopalakrishnan A, Biswal A C. Quiver—An intelligent decision support system for software architecture and design[C]//2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon). IEEE, 2017: 1286-1291.

[6] Manjunath A, Bhat M, Shumaiev K, et al. Decision making and cognitive biases in designing software architectures[C]//2018 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, 2018: 52-55.

[7] Pretorius C, Razavian M, Eling K, et al. Towards a dual processing perspective of software architecture decision making[C]//2018 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, 2018: 48-51.

[8] 乌尼日其其格, 李小平, 马世龙, 等. 高阶类型化软件体系结构建模和验证及案例[J]. 软件学报, 2019, 30(7): 0.

[9] Mei H, Huang G, Zhang L, et al. ABC: a method of software architecture modeling in the whole lifecycle[J]. Scientia Sinica Informationis, 2014, 44(5): 564-587.

[10] 宋晖, 黄罡, 武义涵, 等. 运行时软件体系结构的建模与维护[J]. 软件学报, 2013, 24(8): 1731-1745.

[11] Altoyan N, Perry D E. Towards a well-formed software architecture analysis[C]//Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings. ACM, 2017: 173-179.

[12] Shumaiev K, Bhat M. Automatic uncertainty detection in software architecture documentation[C]//2017 IEEE International Conference on Software Architecture Workshops (ICSAW). IEEE, 2017: 216-219.

[13] 陈湘萍, 黄罡, 宋晖, 等. 基于 MOF 的软件体系结构分析结果集成框架[J]. 软件学报, 2012 (2012 年 04): 831-845.

[14] Bogado V, Gonnet S, Leone H. DEVS-based methodological framework for multi-quality attribute evaluation using software architectures[C]//2017 XLIII Latin American Computer Conference (CLEI). IEEE, 2017: 1-10.

[15] Antonino P O, Jahic J, Kallweit B, et al. Bridging the Gap between Architecture Specifications and Simulation Models[C]//2018 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, 2018: 77-80.

[16] Zhang C, Ma Y, Wang X, et al. Software Architecture Modeling and Reliability Evaluation Based on Petri Net[C]//2017 International Conference on Dependable Systems and Their Applications (DSA). IEEE, 2017: 51-56.

[17] Bandara V, Perera I. Identifying Software Architecture Erosion Through Code Comments[C]//2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer). IEEE, 2018: 62-69.

[18] Olsson T, Toll D, Ericsson M, et al. Evaluation of an architectural conformance checking software service[C]//Proccedings of the 10th European Conference on Software Architecture Workshops. ACM, 2016: 15.

[19] Lutellier T, Chollak D, Garcia J, et al. Measuring the impact of code dependencies on software architecture recovery techniques[J]. IEEE Transactions on Software Engineering, 2018, 44(2): 159-181.

[20] Arcelli D, Cortellessa V, Di Pompeo D. Performance-Driven Software Architecture Refactoring[C]//2018 IEEE International Conference on Software Architecture Companion (ICSA-C). IEEE, 2018: 2-3.

[21] Arcelli D, Cortellessa V, D'Emidio M, et al. EASIER: an Evolutionary Approach for multi-objective Software archItecturE Refactoring[C]//2018 IEEE International Conference on Software Architecture (ICSA). IEEE, 2018: 105-10509.