软件架构核心问题与关键决策探讨

王志刚，胥 茜

（湖南师范大学 数学与计算机学院，湖南 长沙 410081）

摘 要：架构是实现软件相关服务和基本业务系统的重要阶段，而架构决策又是这一阶段的核心。为了帮助理解软件架构的本质和架构实现，概述了架构的基本过程，介绍了系统分解原则，重点讨论了架构决策的范围及主要决策内容。在架构决策框架中解释了实现架构决策的3个层次，以及各层次可以使用的UML图表等工具。掌握好这些方法能有效减少软件架构的盲目性，为软件设计与实现提供规范与指导。

关键词：软件架构；架构决策；系统分解

DOI：10.11907/rjdk.172989

中图分类号：TP301 **文献标识码**：A **文章编号**：1672-7800(2017)012-0057-03

Discussions of the Core Issues and Key Decisions of Software Architecture

WANG Zhi-gang, XU Qian

(*College of Math and Computer Science, Hunan Normal University, Changsha* 410081, *China*)

Abstract: Architecture is an important stage for implementing software-related services and basic business systems, and architectural decisions are the core of this stage. To help understand the full essenceof the software architecture and its implementation, this paper summarizes the basic process of architecture, introduces the system decomposition principle, and focuses on the scope of architectural decisions and major decisions. The three levels of architectural decision are explained in the decision framework, as well as tools such as UML diagrams that can be used at all levels. In summary, mastering these methods can effectively reduce the blindness of software architecture and provide specifications and guidance for software design and implementation.

Key Words: software architecture; architecture decision; system decomposition

0 引言

软件在人类社会活动中发挥着不可估量的作用，软件工程旨在研究软件系统架构、开发、运行、维护、演化的创新方法以提高效率和质量[1]。软件并非开发商为客户提供的第一样东西，但它的好坏会直接影响客户的感受。无论公司大小，对软件的依赖程度都在急剧增加。不管公司关注的战略重点是什么，软件架构已经成为一种能力，掌握得好将极大提升公司的实力和水平，否则会严重削弱项目或系统构建[2]。本文通过视图帮助架构师解决软件架构的关键问题，从而使其概念具有可操作性。

1 软件架构概述

以往软件架构所关注的核心问题是系统的复杂性，然而今天许多系统具有与摩天大楼媲美的复杂性。因此，软件设计问题超出了算法和数据结构，设计和指定整个系统结构就是软件架构级别的设计。很明显，复杂性是软件架构必须解决的一个关键问题[3]。主要体现在两方面：①知识难驾驭，如构建系统的复杂性、规模、依赖关系和采用技术等；②管理难驾驭，如构建系统的组织和流程、参与构建系统的人数、项目的依赖关系、外包、地理分布的团队等[4]。

分解系统可以解决复杂性问题，但是怎么分片？好的分解满足组件之间松散耦合的原则，通过干净接口，合理地将其划分为可单独处理的独立部分以简化问题。该结构必须支持系统所需的功能或服务。因此，必须考虑系统的动态行为，还要有必要的基础设施支持这些服务[5]。

如何将切片结合在一起是界面与各个部分之间的关系问题。保持系统完整性的同时，也需要保证这些切片组成的系统具有良好适应性。这些品质或特性对系统具有分散或系统性影响，因此是交叉分割的关注点。这不是一

收稿日期：2017－10－25

作者简介：王志刚(1962－)，男，湖南沅江人，硕士，湖南师范大学数学与计算机学院教授、硕士生导师，研究方向为软件工程、计算机网络；胥茜(1993－)，女，湖南岳阳人，湖南师范大学数学与计算机学院硕士研究生，研究方向为软件工程。

个孤立的问题，因为分解是由其它问题驱动的，无论怎样分割，多个部分都必须合作解决这些关注点。为了有效地解决跨部门关切的问题，必须首先在范围更广的层面上处理问题。许多系统质量（非功能性要求或服务级协议）具有这种性质，包括性能、安全性和互操作性要求。例如，为了使图像更加复杂，可能会与系统质量发生冲突，因此要考虑到系统质量的相对优先级，必须在备选方案之间权衡。架构的另一个关键是方案能否适应环境，这关系到一致性与和谐，也是业务战略和用户目标一致的问题。

架构不仅要适应原有系统的环境，不破坏以往投资的价值，而且应该对已经被证明有效的东西进行规格化，避免不必要的重复。除整合经验教训外，还应该识别与利用系统内部及跨系统复用。此外，应该预见趋势和可能的未来情景[6]。

2 架构决策核心问题

这需要从广泛范围或系统的角度进行。任何于狭义局部进行的决策，都不是架构。架构决策即使不对整个系统，至少对系统的不同部分会产生影响，因此需要从全局视角考虑这种影响，并作出必要的权衡[7]。

例如，对于单个应用程序，组件设计者作出的任何决策都应该服从架构。如果架构的范围是一组应用程序，则任何与单个应用程序相关的决策都应该服从系统架构。那些对系统影响不大的决策不是架构。所以，架构决策应该关注高影响、高优先级且与业务策略紧密结合的领域，如表 1 所示。

表 1 决策范围与影响

范围	低影响	高影响
系统性	非架构	架构决策焦点
局部性	非架构	一般不是架构

基于软件架构所涉及的核心问题，架构决策包含：优先级设置、分解与合成、属性特别是交叉切割关注点、适应环境和完整性。

2.1 优先级设置

在设计大型复杂系统时，关键要把重点放在明确的事情上，这样就可以把注意力集中在高优先级领域，从而可以合理地作出取舍，并且在商定的优先事项上作出合理的决策。架构师需要负责系统技术方面的优先级设置。这是一个高度战略化过程，涉及：①业务，包括业务策略和方向、核心竞争力和资源；②市场，包括客户、竞争对手、供应商；③渠道技术，包括趋势和机会；④约束，包括现有技术投资、现存系统挑战和系统成功障碍、系统开发和业务[8]。

2.2 分解与组合

软件架构的基础是系统结构，即系统的主要结构元素或组件以及它们的相互关系。在分离特定的关注点时，必须确保系统的功能或服务可由协作中的组件交付。因此对于每个组件的职责要有一致的假设；还必须文档化，形成适当的上下文，以便相对独立地开发；能在不了解内部构件的情况下使用组件。这些是组件的外部可见属性。

2.3 属性与交叉分割

系统分解针对一些关注点进行特定分解，以便能够独立处理。不同的分区选择倾向于隔离不同的关注点。交叉分割会影响系统的各个部分，因此必须将协作组件集合在一起，以正确地处理每一个关注点。例如，性能通常需要在架构所允许的交互模式下加以考虑，而不仅仅是优化各部分的性能。有些关注点，比如互操作性，需要关注特定架构，并且设计一种解决机制。该机制可以被设计为一组协作组件，集中于解决交叉分割关注点。例如，接口可置于非核心内聚集的组件上。

2.4 系统适应上下文

系统适应上下文的关键技术是与外部系统的互操作性、一致性和接口。然而，适合开发机构的文化和功能，也是架构决策和选择的考虑因素。

2.5 完整性

完整性是指应具有统一的整体设计、形式或结构，无论片大小都是一致的。此外，还包括内部平衡、兼容性和各部分之间的和谐，以及适应环境与目标[9]。

首先，在高层结构设计时，需要为系统创建完整概念，确保组件、属性和关系能够加入并为完整性打下基础。这里有一个关键的角色，即视觉、架构风格、原则和概念为高层结构到详细设计提供一致方法。其次，一些决策虽然与高层结构无关，但与架构的完整性有关，也要认定属于架构。然而，系统完整性为架构师打开了一扇门，让他们作出了本应该是组件设计和实现者应该作的决策。因此，简约架构精神应该得到发扬。

2.6 简约架构

可以应用 3 个原则实现简约架构：①若决策是在一个较窄的范围内进行，则交给负责该范围的团队；②只在重大结构性需求上强调架构决策，包括战略目标、重要且独特的服务以及系统性或对系统有交互影响的特质与属性；③当决策被引入时，应该就它们对组织采用架构的整体能力所产生的影响进行评估。一个决策可能会解决一个高度关注的问题，若它会导致架构被破坏，那么将其放在一边。作为简约架构规程的一部分，每个决策都该有一个合理理由和说明。决策应该在全局范围内进行优化，但在局部可能是次优的[2]。因此那些只有局部视野的人必须理解任何局部优化对整个系统的影响[10]。

另外，提供基本原则允许在架构上进行检查和平衡，在不牺牲高优先架构需求的情况下，可以更好地实现重大需求，并且可以合理地与架构决策相冲突。

3 架构决策框架

架构决策实际上处于不同的抽象级别。核心关注点是架构的结构元素和它们的外部属性及关系。然而，还有更高层次的决策可以指导与约束系统分解和结构化决策，也有一些低级的决策指导与约束下一个级别的设计和实

现。这在图 1 所示的软件架构分层模型中得到了体现。模型提供了高度有效的关注点分离方法，帮助架构师组织决策过程，为行动提供定位。它由模型、描述、解释等组成，通过捕获架构决策，帮助相关人员对架构进行可视化，并了解如何处理关注点。架构描述指导系统的创建。

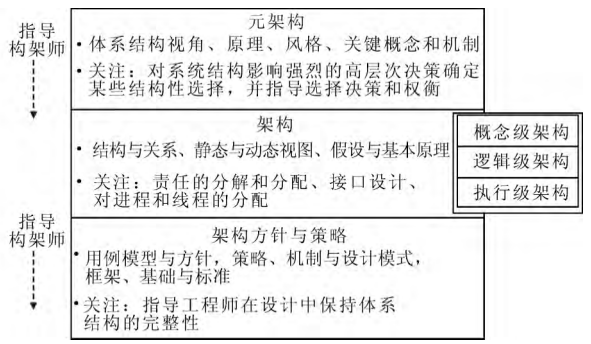

图 1　软件架构分层模型

3.1　元架构

这是一组高层决策，它强调影响系统的完整性和结构，但其本身不是系统结构。通过风格、组合或交互模式、原则和哲学，元架构规定某些结构性选择，并指导选择决策和权衡取舍。通过反复选择跨架构的通信或协调机制，确保方法一致，从而简化架构。这种方法也可以帮助找到对系统有用的隐喻或组织概念。它还可以帮助考虑系统应该具备的质量，甚至帮助选择需要的组件，最终使架构更加生动和易理解[11]。

3.2　架构的 3 个级别

这是分层决策模型的核心，也是架构活动的中心。通过设置系统优先级和约束，确保系统能实现目标和需求。该工作由元架构中的决策提供信息和约束，使用不同的视图增强架构的可理解性，并单独关注特定的关注点。架构由概念、逻辑和执行 3 个级别组成，如图 2 所示。

概念级	• 架构图表，CRC卡片 • 关注点：组件的标识和组件责任的分配
逻辑级	• 更新的架构图表（展示接口）接口规格化，组件规格化与使用说明 • 关注点：组件交互设计，连接机制与协议；接口设计与规格化；为组件用户提供上下文信息
执行级	• 流程视图（通过协作图表展示） • 关注点：运行时组件实例到进程、线程和地址空间的分配；如何沟通和协作；如何将物理资源分配给他们

图 2　3 级架构

(1)概念架构。确定系统的高级组件以及之间的关系，在不关注细节的情况下，注意力集中在系统的适当分解上。此外，还提供一种用于架构与非技术人员之间的通信工具，由架构图(无接口细节)和组件的非正式规范组成。

(2)逻辑架构。在逻辑架构中，组件的外部属性通过定义良好的接口和组件规范实现精确定义，并详细描述关键的架构机制。它提供一份详细的蓝图，使组件开发者和用户可以相对独立地工作。它包含详细的架构图、组件和接口规范以及组件协作图，还有关于机制、原理等的解释。

(3)执行架构。它是为分布式或并发系统创建的。流程视图是组件到物理系统过程的映射，关注的重点是吞吐量和可伸缩性等问题。部署视图是执行系统中组件到物理系统节点的映射。

3.3　架构的结构与行为视图

结构和行为视图对于思考和表示架构都很重要。如果接受架构是由组件组成的系统高层结构、组件的相互关系和外部属性，则结构视图是其核心。它由架构图(原型 UML 类图)、组件和接口规格组成。

将系统分解为组件并设计接口，在设计解决关键交叉分割关注点的机制时，行为视图和它的组件协作或序列图(固定的 UML 序列和协作图)可以帮助理解这个过程[12-13]。

结构和行为视图都适用于概念、逻辑和执行这三级架构。它们之间的关系如表 2 所示。

表 2　结构/行为视图与三级架构的关系

三级架构	行为视图	结构视图
概念级	协作轨迹	架构图，非正式组件说明
逻辑级	协作图	带接口和规格化的架构视图，接口规格
执行级	标识流程的协作图	标识活动组件的架构视图

4　架构指导方针

为了保持系统的完整性或解决分割关注点，架构师可以指导或约束架构决策集的底层设计与实现。也即合理地帮助设计和开发者应用架构并注意问题的正确特征，这样他们的决策就是明确的。

一般来说，除非有严格的架构理由限制设计和开发者自由发挥，否则应避免使用规定。

(1)通信架构决策。除非记住、接受和理解，否则架构决策是无能为力的。架构文档的主要目标是记录决策过程。为了达到目标，文档必须完整且明确。

(2)通信架构。为了达成目标，还必须考虑每个相关人员需要知道什么，以及如何更好地向他们传达必要的信息。

(3)架构驱动。虽然不是体系结构的一部分，但是塑造架构的驱动因素对于明确和分享是很重要的。它们包括：架构远景，即架构的期望状态；架构需求，即相关人员目标和架构功能需求，以及系统质量和约束；假设、势力和趋势，即当前业务、市场和技术环境的断言以及架构规划的范围。

(4)构架视图。图 1 和表 2 展示了一组标准化视图。这些是指导决策时的有用东西，所提供的思考工具帮助决策和选择。它们也是架构规范的基础，是抽象、特异和精确性选择级别中的完整架构决策集。

(5)架构文档。有了架构驱动和各种视图等原始资料，也就可以针对不同的人员编写文档和报告。文档包括所有架构规格书、管理概要和组件文档。

(6)管理概要。为了易于管理，需要创建一个高层次概要，包括远景、业务驱动、架构图及将业务策略与技术策略联系起来的基本方法。

(下转第 63 页)

适应度曲线。因为目标函数是求运输成本，所以适应度越小越好。从图中可以看出，本文提出的算法适应度最小。

表 2　计算结果比较

算法	1	2	3	4	5	6	7	8	9	10	平均
GA	142.0	133.2	140.1	138.0	139.8	135.2	145.5	141.2	147.4	138.3	140.1
HGA	119.6	121..3	125.2	124.5	119.8	119.4	121.5	118..5	122.5	123.4	121.6
KQGA	118.9	118.5	117.5	119.7	118.4	117.3	120.4	119.3	117.8	119.1	118.7

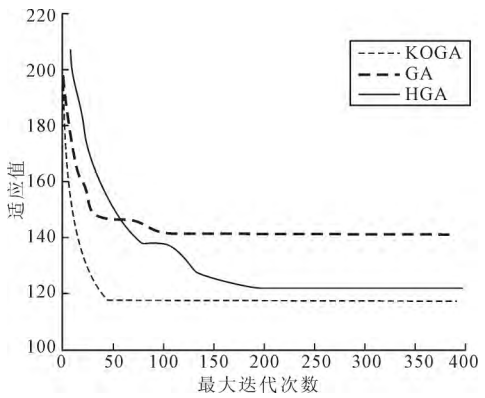

图 4　3 种算法适应值比较

4　结语

本文针对传统的量子遗传算法存储量大且易陷入局部最优等问题，提出了一种改进的量子遗传算法。采用动态调整旋转角机制对量子步长实现自适应搜索，从而加快了算法收敛速度，提高了搜索深度。引入量子自适应变异操作防止了早熟问题。局部搜索采用客户节点重置和 2-opt 法结合对线路进行优化，使解的质量明显提高。通过与传统的 GA 和 HGA 对比，证明了该算法具有较强的寻优能力。目前，KQGA 只是针对静态的带容量车辆调度问题进行研究，今后将对动态车辆调度进行研究，并验证其有效性。

参考文献：

[1] DANTZIGC RAMSERJ. The truck dispatching problem[J]. Management, Science, 1959(6): 80-91.

[2] 赵燕伟，吴斌，蒋丽，等. 车辆路径问题的双种群遗传算法求解方法[J]. 计算机集成制造，2004，10(3)：303-306.

[3] WANG C H, LU J Z. A hybrid genetic algorithm that optimizes capacitated vehicle routing problems[J]. Expert Systems with Applications, 2009, 36(2): 2921-2936.

[4] DORRONSORO B, ARIAS D, LUNA F. A grid-based hybrid cellular genetic algorithm for very largr scale instances of the CVRP [C]. High Performance Computing & Simulation Conference, 2007: 759-765.

[5] 蔡蓓蓓，张兴华. 混合量子遗传算法及其在 VRP 中的应用[J]. 计算化仿真，2010，27(7)：267-270.

[6] WANG Y. Solving the capacitated vehicle routing problem by cellular ant algortithm[EB/OL]. http://comnection. ebscohost. com/c/articles/74249351.

[7] HARYNANAN A, MOORE M. Quantum-inspired genetic algorithms[C]. Proceeding of IEEE International ConferenceonEvolutionary Computation, Nagoya, Japan, 1996: 61-66.

[8] 郎茂祥，胡思继. 用混合遗传算法求解物流配送路径优化问题的研究[J]. 中国管理科学，2002，10(5)：51-56.

（责任编辑：杜能钢）

（上接第 59 页）

（7）组件文档。对于每个组件所有者，最好提供一个系统级视图、组件和接口规范，以及组件协作图。

5　结语

本文讨论了软件架构的重点以及如何表达软件架构。但是，世界上所有的文档和演示都是不够的，除非软件架构是好的、技术上是合理的，能清晰地表明它是符合关键利益相关者的需求和目标，并且能成功地用于开发具有战略优势的系统。

参考文献：

[1] 刘璘，周明辉，尹刚. 大数据时代软件工程专题前言[J]. 软件学报，2017，28(6)：1327-1329.

[2] TNO, EINDHOVEN. Resources for software and systems architects[EB/OL]. http://www. bredemeyer. com/.

[3] 白金. 软件架构模式在信息系统开发中的应用分析[J]. 通信世界，2017(5)：249-250.

[4] 王志刚，高磊. 软件发布规划遗传算法探讨[J]. 软件导刊，2016，15(11)：56-58.

[5] 王炜，陈未如. 软件架构切片在软件可靠性评估中的应用[J]. 微计算机信息，2008，28(1)：290-292.

[6] FEA PRACTICE GUIDANCE. Federal segment architecture methodology (FSAM) practitioner's training version 1. 0. [EB/OL]. http://www. fsam. gov. 2010-08-02.

[7] 李卫华，傅晓东. 可拓创新软件体系结构研究[J]. 广东工业大学学报，2016，34(2)：2-5.

[8] 杨波，于茜，张伟，等. GitHub 开源软件开发过程中影响因素的相关性分析[J]. 软件学报，2017，28(6)：1330-1342.

[9] 林巴斯. 软件构架实践[M]. 北京：清华大学出版社，2002.

[10] 王志刚，高磊. 软件发布规划的遗传算法实现与解释[J]. 现代计算机，2016(12)：3-6.

[11] 俞析蒙. 基于验证的软件架构演化分析与评估[D]. 南京：东南大学，2015.

[12] 王智超，王敏，熊燕. 软件架构设计之多视角分析[J]. 现代计算机，2014(10)：35-37.

[13] NORMAN HENDRICH, HANNES BISTRY, 张建伟. 助老服务机器人系统设计及软件架构[J]. Engineering, 2015(1): 26-34.

（责任编辑：何　丽）