软件体系结构质量评价概述

周 欣 黄 璜 孙家骕 燕小荣
(北京大学计算机科学技术系 北京 100871)

An Introduction to Software Architecture Quality Evaluation

ZHOU Xin HUANG Huang SUN Jia-Su YAN Xiao-Rong
(Department of Computer Science and Technology, Peking University, Beijing 1000871, P. R. China)

Abstract Software quality is one of the most important characteristics of software system and impacts on the system's effect, cost and efficiency. As is well known, it's better to improve the quality as early as possible, which can reduce the cost in following development and maintenance. Software architecture is the first activity from problem space to software solution space, therefore, the decisions made during this process are significant to software quality. Software architecture quality evaluation analyzes and predicts quality from architecture level, helping make proper architectural decisions and detecting derivation during following development. This paper summarizes the researches in this area, introducing the purpose, content, state of the art and application status, analyzing the difficulties and discussing the future directions.

Keywords Software architecture, Software quality, Software architecture quality evaluation, Scenario, Software metrics

1. 引言

随着软件规模和复杂度的不断增大,对软件质量、成本、进度的要求越来越严格。软件质量是指软件对预期的一系列质量属性组合的满足程度[14]。目前,人们已经普遍认识到软件质量控制在软件特别是大型软件开发过程中对开发效率、成本有重要的影响,甚至关系到开发最终是否成功。高质量的软件在维护和测试阶段的开销较低,复用的潜力大。因此,预测和控制软件质量的成熟技术成为迫切的需要。

经过长期的研究,人们已经认识到在软件生命周期的越早阶段对软件质量进行评价越能降低整个开发的成本,在越早阶段对质量进行控制的效果也越好。AT&T 的报告显示,在早期阶段对软件质量进行评价可提高 10%的开发效率[20]。软件体系结构设计是从问题域空间到软件解空间的第一项活动,在体系结构设计阶段的决策对软件质量有至关重要的影响,正是因为人们已经普遍认识到好的体系结构设计是高质量软件的必要条件,我们迫切需要对软件体系结构质量评价的一系列问题进行深入研究,以回答"什么是合乎系统需求的软件体系结构?","哪种侯选体系结构更加适合系统需求?","如何在体系结构设计中做出权衡?","采用某种体系结构,系统未来的质量将会怎样?"……软件体系结构质量评价已经成为软件体系结构领域和软件质量度量领域里一个重要的研究方向。

目前,如何通过软件体系结构的分析评价来确保和提高软件质量成为学术研究和工程实践普遍关注和重点研究的问题之一。CMU/SEI 的软件风险评估过程[19]、CMU/SEI 的软件体系结构分析方法 SAAM[2]和体系结构权衡分析法 ATAM[7]、赫尔辛基大学提出的基于模式挖掘的面向对象软件体系结构度量技术、Karlskrona/Ronneby 大学提出的基于面向对象度量的软件体系结构可维护性评价技术[3]、西弗吉尼亚大学提出的软件体系结构度量方法[25]等反映了这一领域内的成果。

本文介绍了到目前为止学术界和工业界对软件体系结构质量评价所进行的研究和实践,阐述了软件体系结构对软件质量的影响及软件体系结构质量评价的目标;介绍了软件体系结构质量评价的主要方式和几种主要的评价技术;探讨了软件体系结构质量评价存在的困难。最后做出总结并提出一些解决途径的建议。

2 软件质量与软件体系结构

2.1 质量属性

正如前面所说,软件质量是指软件对预期的一系列质量属性组合的满足程度[14]。它也是面向用户的三大产品特性(质量、成本和进度[12])之一。质量属性可分成两类:第一类质量属性可在运行软件并观察其结果的时候进行度量,例如性能、安全性、可靠性和功能性等都属于这一类。第二类质量属性不能够在运行软件并观察其结果的时候进行度量,但可通过对开发或维护过程的观察进行度量。这一类质量属性包括可移植性、适应性、可复用性等[1]。随着软件系统复杂度的不断提高,人们不再单纯地关心系统能完成哪些功能,可移植性、可修改性、可扩展性等非功能性质量属性成为软件体系结构质量评价需要分析的主要目标。

2.2 软件体系结构

尽管软件体系结构自 1993 年以来直到现在都是一个热点研究问题,关于软件体系结构一直都没有唯一确定的定义。现有的各种定义从不同的角度对软件体系结构进行了诠释。虽然形式各异,但都从一定程度上反映出软件体系结构的本质特征。简单地说,软件体系结构问题包括软件系统总体组织和全局控制、通信协议、同步、数据存取,给设计元素分配特定功能,设计元素的组织、规模和性能,在各个设计方案间进行选择[6]。这些都是软件体系结构层次上的设计。

目前,各种软件体系结构质量评价技术对软件体系结构

的理解没有根本上的差异，但不同的技术可能从不同的角度和侧重点去看待软件体系结构，比如，Bass 等人在研究 SAAM 时就提出软件系统的体系结构设计可以由三方面描述：功能划分、结构和功能到结构的分配[2]。而 Helsinki 大学所提出的面向对象软件体系结构评价技术认为 UML 的 Use Case、类图、状态图、活动图、顺序图等就代表了一个面向对象系统的体系结构[8]。

2.3 软件体系结构对软件质量的影响

软件质量受到开发过程中的多项活动的影响。精确的需求定位、正确的设计决策、先进的编程技巧等都将对软件质量产生影响。软件体系结构的设计是从问题域到软件解空间的第一步，在体系结构设计阶段所做的决策很大程度上影响了系统的质量。一些体系结构方面的问题对软件的质量属性有重大的影响，例如，模块的层次可能影响系统的可修改性；功能的划分和封装可能影响系统的可扩展性；构件间的通信协议可能影响系统的效率等。从另一方面看，良好的软件体系结构并不能确保系统的功能性需求及质量需求。后阶段的设计、实现等同样可以使系统的质量发生变化甚至损坏[1]。毕竟，软件生命周期的任何阶段都对软件质量有或大或小的影响。因此，好的软件体系结构是良好系统质量的必要但不充分条件。

2.4 软件体系结构质量评价的目标

众所周知，软件质量在系统开发的后期不可能发生由坏到好的根本性转变，因此在软件体系结构设计阶段对软件质量进行评价的意义尤其重大。软件体系结构质量评价的根本目的是通过对体系结构的分析，一方面了解该体系结构对质量需求的满足程度，发现潜在的质量问题；另一方面预测未来系统的质量特性。

对系统的开发人员而言，软件体系结构质量评价应该帮助他们尽早发现体系结构设计中的偏差和失误，分析软件系统是否满足各方面的质量需求，同时还辅助他们在多个候选体系结构之间进行比较和权衡并选择一个最合适的体系结构作为接下来各个开发过程的指导。

对系统的用户而言，通常他们都希望尽早对系统的质量有更多的了解，传统的质量评价方式主要是针对代码进行分析，用户往往要到较晚时候才可以真正了解到系统的质量属性，但如果在此时用户发现不满意的地方，做任何的修改都很可能是困难且开销巨大的。软件体系结构质量评价必须在尚未生成代码，甚至是尚未进行设计的条件下，对用户所关心的质量属性进行预测，满足用户尽早了解系统的需求。

对于系统的管理人员而言，降低开发成本和控制开发风险是至关重要的。进行软件体系结构质量评价应该帮助他们区分系统中的关键部分和非关键部分，以便合理地分配资源和调整开发进度。

3 软件体系结构质量评价的主要方式

从目前已有的软件体系结构评价技术来看，某些技术通过与经验丰富的设计人员交流获取他们对待评估软件体系结构的意见；某些技术对针对代码的质量度量进行扩展以自底向上地推测软件体系结构的质量；某些技术分析把对系统的质量的需求转换为一系列与系统的交互活动，分析软件体系结构对这一系列活动的支持程度等。尽管看起来他们采用的评价方式都各不相同，但 基本可以归纳为采用了 3 类主要的评价方式：基于调查问卷或检查表的方式、基于“场景”的方式、基于度量的方式。

3.1 基于调查问卷或检查表的评价方式

SEI 的软件风险评估过程[19]采用了这一方式。调查问卷是一系列可以应用到各种体系结构评价的相关问题，其中有些问题可能涉及到体系结构的设计决策；有些问题涉及到体系结构的文档，比如体系结构的表示用的是何种 ADL；有的问题针对体系结构描述本身的细节问题，如系统的核心功能是否与界面分开。检验表中也包含一系列比调查问卷更细节和具体的问题，它们更趋向于考察某些关心的质量属性。例如，对实时信息系统的性能进行考察时，很可能问到系统是否反复多次地将同样的数据写入硬盘。

这一评价方式比较自由灵活，可评价多种质量属性，也可以在软件体系结构设计的多个阶段进行。但是由于评价的结果很大程度上来自于评价人的主观判断，因此不同的评价人可能会产生不同甚至截然相反的结果，而且评价人对领域的熟悉程度、是否有丰富的相关经验也成为评价结果是否准确的重要因素。尽管基于调查问卷与检查表的评价方式相对比较主观，但由于系统相关的人员的经验和知识是评价软件体系结构的重要信息来源，因而它仍然是进行软件体系结构质量评价的重要途径之一。

3.2 基于“场景”的评价方式

“场景”(Scenarios)是一系列有序的使用或修改系统的步骤[17]。基于“场景”的方式由 SEI 首先提出并应用在 SAAM 和 ATAM 中。这种软件体系结构质量评价方式分析软件体系结构对“场景”也就是对系统的使用或修改活动的支持程度，从而判断该体系结构对这一“场景”所代表的质量需求的满足程度。比如说用一系列对系统的修改来反映“易修改性”方面的需求，用一系列攻击性操作来代表“安全性”方面的需求等。这一评价方式考虑到了包括系统的开发人员、维护人员、最终用户、管理人员、测试人员等等在内的所有和系统相关的人员对质量的要求。基于“场景”的评价方式涉及到的基本活动包括确定应用领域的功能和软件体系结构的结构之间的映射，设计用于体现待评估质量属性的“场景”以及分析软件体系结构对“场景”的支持程度。

不同的应用系统对同一质量属性的理解可能不同，比如，对操作系统来说，可移植性被理解为系统可在不同的硬件平台上运行，而对于普通的应用系统而言，可移植性往往是指该系统可在不同的操作系统上运行。由于存在这种不一致性，对一个领域适合的“场景”设计在另一个领域内未必合适，因此基于“场景”的评价方式是特定于领域的。这一评价方式的实施者一方面需要有丰富的领域知识以对某一质量需求设计出合理的“场景”，另一方面必须对待评估的软件体系结构有一定的了解以准确判断它是否支持“场景”描述的一系列活动。

3.3 基于度量的评价方式

度量是指为软件制品的某一属性所赋予的数值，如代码行数、方法调用层数、构件个数等。传统的度量研究主要针对代码，但近年来也出现了一些针对高层设计的度量[4,5]，软件体系结构度量即是其中之一[3]。代码度量和代码质量之间存在着重要的联系，类似地，软件体系结构度量应该也能够作为评判质量的重要依据。赫尔辛基大学提出的基于模式挖掘的面向对象软件体系结构度量技术、Karlskrona/Ronneby 提出的基于面向对象度量的软件体系结构可维护性评价、西弗吉尼亚大学提出的软件体系结构度量方法等都在这方面进行了探索，提出了一些可操作的具体方案。我们把这类评价方式称作基于度量的评价方式。

上述基于度量的评价技术都涉及三个基本活动:首先需要建立质量属性和度量之间的映射原则,即确定怎样从度量结果推出系统具有什么样的质量属性;然后从软件体系结构文档中获取度量信息;最后根据映射原则分析推导出系统的某些质量属性。因此,这些评价技术被认为都采用了基于度量的评价方式。

基于度量的评价方式提供更为客观和量化的质量评估。这一评价方式需要在软件体系结构的设计基本完成以后才能进行,而且需要评价人对待评估的体系结构十分了解,否则不能获取准确的度量。自动的软件体系结构度量获取工具能在一定程度上简化评价的难度,例如 MAISA[8]可从文本格式的 UML 图中抽取面向对象体系结构的度量。

3.4 比较

经过对三类主要的软件体系结构质量评价方式的分析,我们用下表从通用性、评价者对 SA 的了解程度、评价实施阶段、评价方式的客观程度、工具支持程度等方面对这三种方式进行简单的比较。

评价方式	调查问卷或检查表		场景	度量
	调查问卷	检验表		
通用性	通用	特定领域	特定系统	通用或特定领域
评价者对 SA 的了解程度	粗略了解	无限制	中等了解	精确了解
实施阶段	早	中	中	中
客观性	主观	主观	较主观	较客观

4 主要技术

4.1 SAAM&ATAM

软件体系结构一直是 CMU/SEI 的研究重点,在这一过程中,研究人员逐渐发现了体系结构分析的意义和重要性并进行了进一步研究。他们希望寻求一种方法来描述和分析软件体系结构,从而证明这种体系结构能够满足某些非功能性的质量属性要求。1993 年,SEI 和 Texas 大学的研究人员提出了名为 SAAM[1,2]的软件体系结构分析方法。

SAAM 的一个假设前提是单个质量属性之间是独立不相干的,这样的假设当然可以简化分析过程,但实际上多个质量属性之间并非独立,而是相互影响。比如说,如果一个系统对安全性要求很高,它在性能方面就会有所损失。因此,SEI 随后在 SAAM 的基础上提出了名为软件体系结构权衡分析方法的 Architecture Tradeoff Analysis Method(ATAM)[7],该方法与 SAAM 最大的差别就在于考虑了各种质量属性之间的联系和冲突。SAAM 和 ATAM 都是基于"场景"的软件体系结构描述和分析方法,针对某个特定领域内的系统,分析和该领域相关的各种"角色"(roles,包括用户、开发人员、维护人员、管理人员等)的需求,导出一组反映功能需求的直接"场景"或反映非功能性质量属性的间接的"场景"。通过综合分析候选软件体系结构是否满足每个"场景"推导出该软件体系结构对质量属性的满足程度。

SAAMTOOL[11]是一个软件体系结构设计和分析工具,支持软件体系结构的质量评价是它所提供的功能之一。它通过对"场景"和软件体系结构元素之间的联系以及对整个系统的体系结构全部或者部分的可视化来达到分析系统的体系结构的目的。通过这些分析,我们可以看到系统的一些基本特性,比如体系结构的复杂程度、模块间的耦合程度和体系结构模板的匹配程度。同时分析者可以通过分析系统体系结构的不同视图和"场景"之间的关系来得到系统在某些特定状态下的信息。

目前,SAAM 和 ATAM 已经成功地应用到了软件工具、空中交通控制、财政管理、电话、多媒体等多个领域[7]。

4.2 面向对象软件体系结构度量技术

由于面向对象开发范型被人们普遍认为能很好地保证系统的可复用性以及降低维护时的开销,越来越多的系统,特别是大型系统,都采用面向对象的体系结构以获得更大的灵活性,从而降低升级和维护的开销。赫尔辛基大学采用了基于度量的方式进行了这方面的研究。该技术从系统的设计文档 UML 图中获取信息,结合度量模型和识别出来的设计模式来评价和预测系统的性能、复杂度和易理解性等质量属性[8]。该技术所采用的度量模型考虑进了较为全面的类信息和类之间关系。同时,设计模式对软件质量的影响也被考虑在内,研究者希望通过识别设计模式对质量进行预测。

MAISA 度量工具是面向对象体系结构度量技术的辅助工具,其功能是根据从 UML 图中获得的信息计算度量值。MAISA 本身并不能识别 UML,它需要借助于外部 CASE 工具从 UML 中获得信息,然后从生成的 Prolog 和 FAMIX[23] 文件中读取信息并进行度量计算。目前,MAISA 还称不上一个十分完善的工具,它只能以文本的方式提供度量结果,还不能实现计划中的性能估计、模式识别和基于模式的质量预测等功能。不过 MAISA 正在进一步的完善中,可以预想这将是一个十分有用的工具。

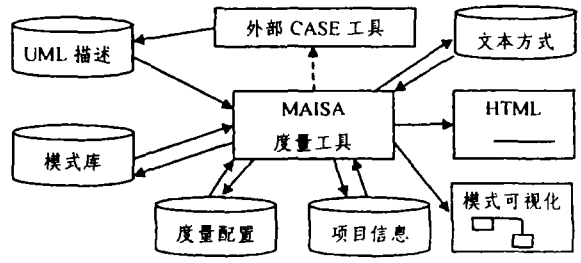

图 1 MAISA 体系结构图

目前,该技术已经成功应用到诺基亚(NOKIA)公司 DX200 交换系统的呼叫控制(Call-control)软件体系结构设计中。

5 难点

根据对现有的评价方式的分析,我们发现尽管基于软件体系结构的质量评价已经初成体系,有了较大的发展,但还存在许多困难有待进一步研究解决,其中最主要的一些有:

★不同的应用领域通常都有特殊的质量需求,例如:电信系统对性能和可靠性十分重视,而用户界面系统可能对易修改性考虑得更多。如果将电信系统的衡量标准套用到用户界面系统,将会产生不合适的结果。

★对同一质量属性的认识也不一样,这可能导致混淆。例如,操作系统的开发人员对"可移植性"的理解是系统可在不同的硬件平台上运行,而应用系统的开发人员认为"可移植性"是指系统可以在不同的操作系统或用户界面工具上运行。

★不存在统一有效的针对质量评价的软件体系结构表示法,SAAM 采用块-线图来表示待分析的软件体系结构,这种表示法过于简单,很难有效地表示足够的体系结构信息。

UML 对于面向对象体系结构的表示能力比较合适，但不能用于表示其它风格的体系结构。

★诸如易修改性、可维护性等一些质量属性涉及到系统未来的需求，而这些需求目前很难准确地获得，因此很难进行有效的评价。

★大多数质量属性都很难用一个简单的尺度去衡量。比如某个用户界面系统在设计的时候充分考虑新手的使用方便，但对于经验丰富的人来说，这样的系统使用起来可能会显得十分烦琐，这样的系统"可用性"是好是坏没有绝对的答案。

★对于基于度量的评价方式，要建立合理的度量模型及获得足够的度量信息。前者需要和实践相结合，不断获取经验数值对模型进行修改和扩充，这是一个比较长期的过程；后者很大程度上依赖程序理解技术对模式识别、软件体系结构恢复等目标的支持，但事实上这两个问题目前还处于研究的初步阶段，并未取得突破性的进展。

总结 软件体系结构质量评价虽然取得了不小的进展，但还面临许多困难，我们认为未来应该针对这些存在的困难进行进一步研究：

★清晰定义软件质量属性：正如 Fenton 所指出的"对许多软件属性，我们还只有非常粗糙的经验关系系统"[24]。许多质量属性，尤其是外部属性，如复杂性、可理解性、可维护性以及质量等等都缺乏严格的定义；有关的经验关系系统还有待研究。尽管给出一个普适的通用定义不太可能，但针对某个领域清晰定义软件属性是可行的。

★研究度量模型：目前基于度量的体系结构级评价方式所采用的大多是传统度量模型的改良，传统度量模型主要考虑来自源代码的度量信息，而体系结构级评估的度量值除了来自代码，还可能来自体系结构描述、各种抽象级别的设计文档等，因此需要研究专门的度量模型。

★实践应用：将软件体系结构质量评价技术应用到软件工程实践中去，一方面为系统的软件体系结构设计提供支持，另一方面让实践来检验其有效性、合理性和实用性。

★自动化支持：目前针对这一问题的支持工具发展还远不成熟，需要我们进一步开发出实用的工具，来辅助评估者做出正确判断。工具应该针对所采用的评价方式，提供不同的功能。

参 考 文 献

1 Clements P, Bass L, Kazman R, Abowd G. Predicting software actuality by architecture-level evaluation. In: Proc. of the Fifth Intl. Conf. on Software Quality. Oct. 1995

2 Kazman R, Bass L, Abowd G, Webb M. SAAM: A method for analyzing the properties software architectures. In: Proc. of the 16th Intl. Conf. on Software Engineering, May 1994. 81～90

3 PerOlof Bengtsson. Towards Maintainability Metrics on Software Architecture: An Adaptation of Object-Oriented Metrics. Firsrt Nordic Workshop on Software Architecture (NOSA'98), Ronneby, Aug. 1998

4 Harrison R, Counsell S, Nithi R. Coupling Metrics for Object Oriented Design. In: Proc. of the 5th Intl. Symposium on Software Metrics, 1998. 150～157, S. Benlarbi and W. Melo: Polymorphism Measures for Early Risk Prediction. In: Proc. of the 21th Intl. Conf. on Software Engineering, 1999. 333～344

5 Chidamber S, Kemerer C. A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, 1994, 20(6): 476～493

6 Garlan D, Shaw M. An introduction to software architecture: [Technical Report: CMU/SEI-94-TR-21, ESC-94-TR-21]. Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 1994

7 Kazman R, Klein M, Barbacci M, Longstaff T, Lipson H, Carriere J. The architecture tradeoff analysis method. In: Proc. of ICECCS'98, 1998

8 Nenonen L, Gustafsson J, Paakki J, Verkamo A I. Measuring object-oriented software architectures from UML diagrams. University of Helsinki, Department of Computer Science, 2000

9 Gustafsson J, Nenonen L, Paakki J, Verkamo A I. Performance-modeling in UML. University of Helsinki, Department of Computer Science, 2000

10 Clements P. Coming attractions in software architecture: [Technical Report: CMU/SEI-96-TR-008]. Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 1996

11 Kazman R. Tool support for architecture analysis and design. In: Proc. of the ACM SIGSOFT'96 Workshops. San Frar, Oct. 1996. 94～97

12 Mario R, Barbacci, Mark H, Klein, Charles B. Weinstock. Principles for evaluating the quality attributes of a software architecture: [Technical Report: CMU/SEI-96-TR-036]. Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 1996

13 Abowd G, et al. Recommended best industrial practice for software architecture evaluation: [Technical Report: CMU/SEI-96-TR-025]. Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1996

14 IEEE Standard 1061-1992. Standard for a Software Quality Metrics Methodology. New York: Institute of Electrical and Electronics Engineers, 1992

15 Mayrand J, Coallier F. System acquisition based on software product assessment. 18th ICSE, Berlin, March 1996

16 Taoxie. Object Oriented Software Quality Evaluation Technology: [Technical Report for Ricoh]. Peking University Software Engineering Institute, 2000

17 Kazman R, Jeromy Carrière S, Woods S G. Toward a discipline of scenario-based architectural engineering. Annals of Software Engineer, 2000. 5～33

18 Bosch J, Molin P. Software architecture design: evaluation and transformation. University of Karlskrona/ Ronneby, Department of Computer Science, 1999

19 Joseph, Sujoe&Sisti, Frank. Software risk evaluation method, Version 1.0 (CMU/SEI-94-TR-19, ADA290697). Pittsburgh, Pa: Software Engineering Institute, Carnegie Mellon University, 1994

20 AT&T. Best current practices: software architecture validation. Internal report. Copyright 1991, 1993, AT&T

21 Kazmen R, Bass L, Clements P. Scenario-based analysis of software architecture. IEEE Software, 1996, 13(6): 47～55

22 Laguě B. Assessing Risks Related to Software Source Code using Datrix™. QAMC Workshop 1997, Ojay, Canada, May 1997

23 FAMIX format. http://iamwww.unibe.ch/～famoos/FAMIX/

24 Fenton N E. Software measurement: a necessary scientific basis. IEEE Trans. Software Eng., 1994, 20(3): 199～206

25 Yacoub S, Ammar H H, Robinson T. A Methodology of Architectural-Level Risk Assessment using Dynamic Metrics. In: Proc. of the 11th Intl. Symposium on Software Reliability Engineering, ISSRE'2000, San Jose, CA, Oct. 2000. 210～221