软件体系结构 软件体系结构 软件开发

计算机科学1998 Vol. 25№.1

83-86

软件体系结构概念

The Concepts of Software Architecture

汪 琼

(北京大学计算机系 100871)

TP311.52

摘 要 Software architecture as a method of software reusability has been gotten more attentions now. To identify the definition and the research contents will help us do more research in this area.

关键词 Software architecture, Software reusability

在软件领域关于体系结构的提法可以追溯到1969年Fred Brooks提出的概念结构。他和Ken Iverson称体系结构是"程序员看到的一个计算机系统的概念结构",是对若干实际系统的概括。几年后Brooks定义体系结构为"用户界面完整详细的说明,对于计算机,这是程序设计手册,对于编译器,这是语言手册,对于整个系统,则是用户完成整个工作所用到的所有手册"。在此,体系结构和实现是两个不同的概念,体系结构告知什么事发生,而实现告知是如何发生的。这种区别至今仍旧存在,在面向对象的程序设计中更是如此。

尽管体系结构一词现在还有人是作为"一个系统的用户视图"来使用的,但是软件体系结构决不是一个系统的用户视图,而是那些对用户来说相对隐蔽的系统结构。"体系结构是一类系统的共同描述"仍旧是这个概念的中心。

软件体系结构是软件结构研究的继续。1968年,Edsger Dijkstra指出,为了产生正确的结果,相对于简单编程,应该更加注意软件是如何分解,如何结构化的。当时Dijkstra正在写一个操作系统,他提出层次结构的概念,即程序分层组织,一层中的程序只能与其相邻层的程序通讯。Dijkstra认为这样的组织所展示的优雅的概念完整性可以使开发和维护更为方便。

David Parnas发展了软件结构研究,他提出了信息隐蔽模块概念[1972]、软件结构概念[1974]和

图5显示了在256个处理机Ncube-2上重构一个二维图象时,上述三个并行化版本的性能。由图5可见,当处理机的个数小于50时,是否采用分解变换技术对加速比的影响不大,这是由于较少的处理机个数无法充分利用分解变换所开发出的额外并行机会。但当处理机的个数增加时,仅利用循环并行所获得加速比的增长幅度很小,这是由于循环并行无法使所有的处理机处于忙状态,处理机的利用率不高。在对程序使用了分解变换技术后,获得了进一步的性能改进。

结束语 在串行代码的并行化过程中,传统采用的各种循环变换技术旨在开发循环并行,因而忽略了许多可利用的并行机会,在其基础上进一步采用分解变换技术,可以获得更大并行效益。

参考文献

[1] Sharp O. J., Transforming for parallelism using symbolic summarization. PHD thesis, Department of Computer Science, University of California at Berkeley, 1995

[2] Allen J. R., Kennedy K., Automatic loop interchange. In: Proceedings of the SIGPLAN Symposium on Compiler Construction, 1984

[3] Ted G. L., Hesham E. R., Introduction to parallel computing. Prentice-Hall, Inc, 1992

[4] Zima H, et al., SUPERB: A tool for semi-automatic MIMD/SIMD parallelization. Parallel Computing, 1988, 6

[5] Hwang K., Advanced computer architecture: parallelism, scalability, progrommability. McGraw-Hill, inc, 1993

[6] Callahan D, et al., ParaScope: A parallel programming environment. Int. J. Supercomputer, Appl, 2 (4), 1988

程序家族概念[1975]。一个程序家族是一组程序的集合(这些程序并不要求都已建立),划分原则只是因为将这些程序视为一组会带来好处,或有一定的用途,并不一定是"类似功能"的集合,如软件工程环境和游戏在讨论辅助建立图形用户界面的工具时就可以视为是一个程序家族的。程序家族的建立和管理与现在的软件体系结构有类似之处。

软件体系结构领域的所有工作都是为了建立一个基于体系结构的软件开发范型,其动机与 Dijkstra 和 Parnas 一样:结构最重要,结构正确会带来好处。对软件体系结构研究的展开与深入,也标志着软件工程的理论和技术在不断的发展、抽象,软件工程成为科学,又向前迈进了一步。

一、基本概念

关于什么是软件体系结构,目前还没有一个标准的一致接受的定义,许多专家基于自己的阅历,从不同角度不同侧面对软件体系结构进行了刻画,下面按照时间序列出了软件体系结构方面的一些重要论文中对软件体系结构的定义,从中我们也可以看到软件体系结构概念是如何演化发展的:

Perry & Wolf, 1992[3] 在他们早期关于软件体系结构的论文中指出:

体系结构由一组具有特定形式的体系结构元素或称设计元素构成,有处理元素、数据元素和连接元素三类。

许多文献都从这个定义出发进一步发展了其内涵。

Garlan & Shaw, 1993[4] 在其一篇被誉为是软件体系结构研究的重要论文中认为:

软件体系结构是设计过程的一个层次,它处理算法和数据结构之上关于整体系统结构设计和描述方面的一些问题,如大体组织结构和全局控制结构;关于通讯、同步和数据存取的协议;设计元素的功能定义、物理分布和合成;设计方案的选择、评估和实现等。

Bass 等,1994[6]在其一篇介绍用系统固有的质量属性评估体系结构的方法的论文中写道:

一个系统的体系结构设计可以至少从以下三个方面进行描述:该系统应用领域的功能分割、系统结构和结构的领域功能分配。

Hayes-Roth,1994[2] 在 ARPA 特定领域软件体系结构(DSSA)的技术报告中说:

软件体系结构是一个由功能构件组成的抽象系统的说明,按照功能构件的行为、界面和构件之间的相互作用进行描述。

Garlan & Perry, 1995 在做 95 年 4 月 IEEE Transactions on Software Engineering 杂志的客座编辑时修正他们原先对体系结构的定义为:

软件体系结构包括一个程序/系统的构件的结构、构件的相互关系、以及控制构件设计演化的原则和指导三个方面。

(这个定义是卡内基梅隆大学软件工程研究所关于软件体系结构为期一周的讨论班的结论)

Boehm 等,1995 Barry Boehm 和他在南加州软件工程中心的学生写道:

一个软件体系结构由以下三部分组成:①一组软件系统的成分、连接和约束;②一组系统仓库管理员提出的需求;③一个理论,它证明用构件、连接和约束定义的一个系统在实现后能够满足系统仓库管理员需求。

Soni, Nord, 与 Hofmeister, 1995 这三位西门子研究公司的研究人员基于他们所研究的在工业项目的开发环境中有影响的一些流行结构,提出:

软件体系结构至少有四个不同的实例化结构,这些结构从不同的方面描述了系统:①概念体系结构按照主要设计元素和这些元素之间的关系描述系统;②模块连接体系结构包含两个正交的结构:功能分解和分层;③执行体系结构描述了一个系统的动态结构;④代码结构描述了在开发环境中源代码、二进制和库是如何组织的。

Shaw, 1995 在第一届关于软件系统体系结构的国际专题讨论会上,Mary Shaw 总结讨论会上的观点和定义,将软件体系结构定义分类为多个模型:

+结构模型认为软件体系结构是由构件、构件间的连接加上一些其他方面组成的,包括:①配置,风格(configuration, style);**②条件,语义**(constraints, semantics);**③分析,属性**(analyses, properties);**④理论,需求说明、仓库管理员需求**(rationale, requirements, stakeholders' needs)。

这方面工作的代表是体系结构描述语言(ADLS)的发展,体系结构描述语言是用于方便描述一个体系结构的构件和联系的形式化语言,通常采用图形化描述,提供某种形式的"方框和线"语法来说明构件,并将构件连接在一起。

+框架模型类似于结构观点,但主要强调整个系统的内在结构(常常是单一的),而不是它的组成

框架模型经常以特定领域或特定问题类为目

标，这方面的工作有特定领域的软件体系结构，CO-BRA 及基于 COBRA 体系结构模型，和特定领域的构件仓库，象 PRISM。

＋动态模型重在系统的行为质量，"动态"可以是整个系统的配置变化、通讯和交互通道的设置或取消，也可以是计算过程中的动态性，如数值改变等。

＋过程模型重在体系结构的构造，构造的步骤和过程。按照这个观点，体系结构是实施过程描述的结果。

这方面的主要工作是产生体系结构的过程程序设计。

这些模型互不包含，也没有本质上的冲突，它们代表了软件体系结构研究的广度，并侧重于不同的方面，如组成构件、整个实体、建立后的行为方式或者建立方式，这些模型在一起构成了软件体系结构的一致的概貌。

二、研究目的和研究内容

前面关于软件体系结构定义的讨论中已涉及软件体系结构的研究内容，概括来说，软件体系结构领域主要研究软件体系结构的选择、表示、评估和分析，总结优秀的工程经验，开发一系列工具、技术、方法和语言，帮助大型软件系统的开发人员用体系结构的概念来改进开发效率、提高产品质量，增强开发过程的可靠性。其中：

体系结构的表示：使用领域分析技术和工具产生该领域所具有的特征目录(features catalog)，用体系结构描述语言或类似的语言进行描述。如此的特征目录可以方便开发人员阅读，促使他们使用体系结构进行软件开发，也可以让语言设计人员依此设计现有语言不能表示的特征和能力。

体系结构的选择：当前软件系统结构的选择是随意的，并没有形成工程原则。目前的研究是通过大量的实例，考察体系结构的决策是如何受项目需求、组织目标和个人背景影响。

体系结构的评估：为什么选择某个体系结构，有什么理由相信这个选择是对的。这是体系结构评估所要回答的问题。美国卡内基·梅隆大学开发了一种称为软件体系结构分析方法(SAAM)，根据所需功能和质量需求比较若干候选的系统结构。SAAM 是基于场景的评估技术，即按照体系结构在特定情景下的执行情况进行比较。SAAM 方法已成功地运用到一些应用领域，正在形式化并扩大功能。

软件体系结构的基础研究包括：体系结构设计的形式化基础、模块界面语言、体系结构描述语言、特定领域体系结构、体系结构设计环境等。

三、研究倾向

软件体系结构的研究目前有两个倾向：

1. 通用体系结构研究

分析研究这些年在各个领域开发的应用系统，总结构造复杂系统已有的一些共同的方法、技术和模式，提炼而形成若干通用的软件体系结构，如"管道结构"、"面向黑板的设计"、"客户服务器系统"等。尽管这些模式术语很少有精确定义，但是这么多年来设计人员已理解这些词的内涵，了解这些术语所代表的系统的计算模型、与类似系统的联系和系统演化的过程，这些术语确实提高了设计人员描述复杂系统的抽象能力。

2. 特定领域研究方法

研究特定领域，为产品家族提供可重用框架，即提取相关系统的共同方面，这样每个新系统可以通过实例化共同的基础结构而建立。在这种方法中，软件体系结构研究可以视为事后努力以提供一个结构化的仓库，软件体系结构的工作就是编码程序家族成员的共性，这样每个家族成员内在的高级决策不必重新发现、重新验证、重新描述。

这方面的典型代表是编译器领域，经过七十年代八十年代的研究发展，编译器设计已形成一个标准化过程，现在每一个白手建立编译器的人都会重用这些总结出来的编码经验，并且可以毫无障碍地探讨词法扫描器、语法树、属性文法、目标代码生成、优化等编译术语和技术，而无论编译的语言是 LALR(1)文法语言，还是有限状态机文法；是自己写语法分析，还是用 YACC 自动生成语法分析。

软件体系结构研究与传统研究的不同之处在于：

1)所关心的问题不一样。传统的研究主要关心算法和数据结构的设计，而软件体系结构是关于一个大系统的整体结构的设计，现有的计算机科学的学科和理论主要研究前者，因此软件体系结构需要建立自己的概念、理论和工具。

2)描述的着眼点不同。传统的系统描述基于定义-使用结构，按照源代码对系统进行模块化，在定义点和使用点有明显的依赖关系；体系结构描述则基于相互作用的构件关系图，即将一个系统模块化为"构件"和"连接"组成的图，构件定义了系统的主

要计算，连接定义了构件之间的相互作用，连接类型可以是简单的过程调用和数据共享，也可以是复杂的管道、事件黑板、客户服务器协议、数据库存取协议等。

3)研究的对象不同。一个体系结构的实例是一个特定系统的体系结构，这是传统研究的，比如用带有流程图的系统文档来描述该系统的体系结构。软件体系结构重在研究体系结构和体系结构风格，比如"管道和过滤器"风格的体系结构定义了可以用增量流转换图来表示的体系结构。一个体系结构风格规定了构件和连接的名称(如过滤器和管道)、拓扑约束(如图不可以循环)、语义约束(如过滤器不能共享数据)、构件或连接的特定实例(如，系统一定有一个数据库)。体系结构风格可以是抽象的体系结构模式或定式(象客户服务器的组织)，也可以是具体的"参考体系结构"(象 ISO OSI 通信模型，传统编译器的线性分解模型)。

四、重要意义

大系统体系结构的设计对于最终系统的成败起着举足轻重的作用，传统体系结构的设计常常是非形式化的，结果难以对体系结构的设计和原则进行交流、分析和比较，当前对体系结构的研究有可能建立体系结构设计的原则，从而提高我们有效建立软件系统的能力。应用软件体系结构的原则进行软件开发对软件开发会带来重大影响：

1)提供了新的系统分析手段。体系结构描述除了提供清晰精确的文档外，还要对文档自动进行一致性和依赖性分析，暴露隐藏的各种问题。一致性分析包括分析软件体系结构对体系结构风格的一致性、体系结构和需求的一致性、设计和体系结构的一致性，依赖性分析是分析需求、体系结构和设计之间的依赖关系，以及体系结构及风格与具体软件设计和需求之间的相互影响。此外运用体系结构描述还可以进行某种风格体系结构的特定领域分析等。

2)促进对系统的理解。由于软件体系结构是对整个系统结构的抽象，简化了我们对复杂系统的理解，而且，体系结构描述在揭示系统设计的高级约束的同时，也为我们选择特定的体系结构提供了依据。

3)有利于软件重用。体系结构描述注意对特性的识别和关系的识别，这样根据体系结构关于构件及构件间关系的详细描述，可以方便地找到可重用构件，同时了解到构件使用的环境，提高了重用的准确性。软件体系结构支持多级重用，既支持较大构件的重用，如子系统的重用，也支持构件集成的可重用框架的重用。关于特定领域软件体系结构和参考体系结构的研究已证实了这一点。

4)提供系统管理依据。软件体系结构规定了系统演化的方向，通过显性说明系统的"支撑墙"，系统维护人员就可以较好地理解系统的变化，较精确地估计修改和维护的花费。

总而言之，软件体系结构提供了满足系统需求的框架，为系统的设计和实现提供了技术和管理的基础。

参考文献

[1] Gregory Abowd et al., In Proc. of SIGSOFT'93: Foundations of Software Engineering, Dec. 1993

[2] Erick Mettala and Marc H. Graham, Technical Report CMU/SEI-92-SR-9, CMU Software Engineering Institute, June 1992

[3] Dewayne E. Perry and Alexander L. Wolf, ACM SIGSOFT Software Engineering Notes, 17(4), 1992

[4] David Garlan and Mary Shaw, In Advances in Software Engineering and Knowledge Engineering, Volume I. World Scientific Publishing Company, 1993

[5] IEEE Trans. Software Eng., 21(4), 1995

[6] Len Bass et al., Technical Report CMU/SEI-94-TR-10, CMU Software Engineering Institute, August 1994

[7] Michal Young, et al., IEEE Trans. Software Engineering, 14(6) 1988

[8] Bruce Anderson et al., OOPSLA'94, pp. 356-359

[9] Domain-Analysis Guidelines (draft), DoD Software Reuse Initiative, DISA/CIM, 701S. Courthouse Rd., Arlington, VA22204-2199; (703)285-6589

[10] Will Tracz ACM, SIGSOFT Software Engineering Notes, 19(2)1994

[11] Walter A. Rolling, ACM SIGSOFT Software Engineering, 19(3) 1994