计算机科学1994Vol. 21 №. 4

软件体系结构的基础研究

朱 冰 编译　梅 宏 审校

摘 要 本文讨论软件体系结构研究的基本问题。首先通过和其他体系结构的类比，给出有关软件体系结构的直觉认识，再以此给出其模型，最后介绍两个实例。

一、引言

六十年代的软件危机导致了七十年代人们对软件设计的高度重视。八十年代，软件工程研究方向又从软件设计转向在软件过程及管理的框架内考虑设计及设计过程，并且有关形式化技术和集成化 CASE 环境的研究均得到较大地进展。

九十年代则被认为是软件体系结构的年代。有关软件体系结构的研究主要在两方面展开：一是特定的软件体系结构的定义，二是研究开发软件体系结构的支撑工具。现在，有必要总结有关软件体系结构开发的技术和经验，并且探讨软件体系结构在软件过程和软件过程管理这两个领域内所起的作用。

软件体系结构可能会起如下的作用：

1)可作为满足需求分析的框架；

2)可作为软件设计的技术基础，还可作为成本估算和过程管理的管理基础；

3)有助于软件重用；

4)可作为依赖性和一致性的基础。

对软件体系结构的基础研究是为了支持软件体系结构规范说明的开发和使用，为软件结构的进一步研究打下基础。

二、对软件体系结构的直觉认识

有许多不同种类的软件体系结构，但目前还只是一些直觉，没有对软件结构进行形式化或规范化。为了更好地研究软件体系结构，先考察一下硬件体系结构、网络体系结构和建筑体系结构。

1. 硬件体系结构

硬件体系结构的实例有 RISC 机，流水线机和多处理器机等。RISC 机强调机器的指令集，流水线机和多处理器机强调硬件的体系结构成员的组成方式。

从组合的角度来看，硬件体系结构有两个特点：1)组成硬件体系结构的设计元素的数目较小；2)通过复制这些设计元素可组成较大规模的硬件体系结构。

可以通过类比硬件体系结构的构成方法来组织和配置软件体系结构，但软件体系结构的潜在设计元素之数量非常大，并且软件体系结构的规模不仅可由复制设计元素而获得，也可通过增加更多的不同的设计元素来获得。

2. 网络体系结构

网络体系结构的实例有星形网络，网状网络等。网络体系结构的特点：

1)网络体系结构有两个元件——结点和联系；

2)只考虑少量的拓扑体系结构。

与网络体系结构相类比，软件体系结构有其相应的两个元件——进程和进程间的通讯。但是，软件体系结构中，不是少量拓扑体系结构，而是相当大量

Specifications, Lecture Notes in Computer Science 618, 1992

[3] J. Widom et al., Set-Oriented Production Rules In Relational Database System, Proc. ACM SIGMOD Int. Conf. on Management of Data, 1990

[4] S. Ceri et al., Deriving Production Rules for Incremental View Maintenance, Proc. of 17th Conf. on VLDB, 1991

[5] Oscar Diaz et al., Rule Management in Object Oriented Database--A Uniform Approach, Proc. of 17th Conf. on VLDB, 1991

[6] 王珊(主编), SYBASE 关系数据库系统原理和指南，中国科技大学出版社，1993. 5

的可能拓扑体系结构，并且一般不可能对那些拓扑体系结构进行命名。另外，软件体系结构强调的也不仅是拓扑体系结构。

3. 建筑体系结构

建筑体系结构中的一些观点对软件体系结构很有帮助，如：多视图、体系结构风格、体系结构风格与工程的关系、体系结构风格和材料的关系等。

一个建筑体系结构有许多不同的视图。视图从建筑物的某些具体方面来介绍建筑物。例如，立体视图给顾客一个建筑物的整体风貌；楼层计划给出"自顶向下"视图，它给建筑工人介绍了相同的楼层规划，外加一些附加的体系结构视图，这些视图提供大量的细节，如电线、水管、暖气和空调的设计。

对不同的应用系统和不同的用户，也需要提供软件体系结构的大量的不同视图。目前，我们仅仅只有一个视图，即实现。实现就象是提供给建筑工人的详细视图，其所有的细节都可见。这样，很难将系统的设计和体系结构从所有的细节中抽象出来。

体系结构风格的概念有助于描述和说明视图。体系结构风格定义了设计元素特有的组合形式；限制了体系结构元素的种类。

工程规划和建筑体系结构以及体系结构风格有着密切的关系。例如，一个人不能从罗马式的建筑工程中获得正交风格的轻松自由的感觉。需要不同的工程规划，才能从罗马式建筑的宏伟感移至正交建筑的轻巧感。

建筑风格和材料之间的关系也是极为关键的。提供一个特定的建筑风格时，要利用建筑材料的某种特性。人们不可能用木制的柱子和横梁修建摩天大楼。设计元素的材料给一个体系结构提供了美学和工程的基础。

这样，在建筑体系结构中，我们找到了有关软件体系结构的基础点：通过提供多视图以理解体系结构的不同方面；体系结构风格可用于描述和说明软件体系结构；工程规划和材料特性在开发和支持特定的软件体系结构和软件体系结构风格时非常有用。

三、软件体系结构的研究背景

我们认为软件产品应具有如下特点：

1）需求方面：涉及信息和处理的确定，以及用户系统所需的信息和处理的特性。

2）体系结构方面：涉及体系结构元素的选择，体系结构元素的相互作用以及对体系结构元素的约束。

3）设计方面：涉及模块化，设计元素的详细接口，支持软件体系结构和满足需求的算法，过程和数据类型。

4）实现方面：涉及满足以上三个方面的算法和数据类型的表示形式。

具体的软件产品不可能这样简单。软件的模型、抽象、转换和表示之间是连续的，我们将这一连续的过程简化成四个离散的部分，给出软件体系结构与软件系统的需求方面和设计方面的关系。

四、软件体系结构的研究动机

软件费用增长的因素很多，其中与软件体系结构密切相关的因素是演化和产品化。软件系统通过不断演化来适应新的应用。但演化常会导致软件系统的脆性增大，对软件系统的修改越来越困难。在软件的产品化过程中，由于软件的软件体系结构的不成熟，因而存在许多与软件体系结构有关的问题。在建立软件系统时，还得为每个新的软件体系结构重新创建它所需的设计元素。将来要对体系结构风格进行标准化，对不同的体系结构风格，要有体系结构模板，配制专门的体系结构元素和形式。现在，每一个软件系统本质上还是新的体系结构，新的体系结构风格。

基于以上原因，在进行软件体系结构的规范说明时要考虑以下几点：

1）以软件体系结构进行约束说明，指出哪些是必要的，哪些是可有可无的。

2）将基于美学的考虑和基于工程的考虑分开，即将软件体系结构中的"支撑物"和"装饰物"区别对待。

3）通过合适的视图来描述软件体系结构的不同方面。

4）进行依赖性和一致性分析。依赖性包括软件体系结构、需求分析和软件设计之间的互依赖性，软件体系结构不同部分之间的互依赖性。一致性指体系结构风格之间、风格和体系结构之间，以及体系结构元素之间的一致性。

五、软件体系结构的模型

1. 模型

软件体系结构是具有特定形式的体系结构元素或设计元素的集合，亦即：

软件体系结构＝｛元素，形式，推理｝

软件体系结构的元素包括三类：处理元素，数据元素和联接元素。

处理元素指能对数据元素进行转换的部件；数据元素指包含使用和转换信息的元素；联接元素是将软

件体系结构的不同碎片粘在一起的粘合剂。如过程调用、共享数据和共享信息都是起"粘合"体系结构元素作用的联接元素的实例。联接元素在区别一个结构与另一个体系结构时起着极其重要的作用，并且能影响一个特定体系结构或体系结构风格的特性。

软件体系结构的形式包括加权、特性和关系。通过加权，可以形式化地区别"支撑物"和"装饰物"，还可以对软件体系结构进行约束，使之具有可选性。特性定义对体系结构元素的约束，有助于选择不同的体系结构元素。关系用于约束体系结构元素的"布局"，它反映不同元素的相互作用关系，将不同的元素安排在软件体系结构中。

软件体系结构的推理说明如何满足软件系统的约束。通过推理来选择体系结构风格、结构的元素和体系结构的形式。

2. 体系结构风格

软件体系结构是对体系结构元素的形式化组织，而体系结构风格则是对体系结构元素的抽象，以及对不同的特有软件体系结构的形式方面的抽象。一个体系结构风格一般比一个特有的软件体系结构少些约束，也少些完备性。但体系结构风格和软件体系结构之间没有硬性的区别。一个人的软件体系结构可能是另一个人的体系结构风格。这取决于使用的情况。

体系结构风格将约束的重点放在体系结构元素和体系结构元素之间的关系上。通过体系结构风格来约束软件体系结构，增加了软件体系结构的可视性。

3. 处理元素、数据元素和联接元素间的互依赖性

软件体系结构有三个重要的视图：过程视图、数据视图和联接视图。处理视图的重点放在数据流上；数据视图的重点放在过程流上。数据视图对联接元素不象过程视图那样重视。

过程视图和数据视图是相互影响的。在数据元素和处理元素所具有的某些重要特性上，彼此依赖于对方。在联接视图中，联接元素是从处理器到处理器进行数据移动的机制，而为能实现这种数据移动，联接元素必会有某些为数据元素所需的特性，同时，处理元素也必须有某些为数据元素所需要的特性。

研究软件体系结构时，这三种视图都是非常重要的，它们之间是可相互移动的。

六、软件体系结构的实例

多阶段编译器是软件体系结构风格的一个实例。我们以此为例来说明软件体系结构的描述方法。本节中，先介绍多阶段的体系结构风格，再讨论两个多阶段风格的编译器体系结构：串行形式的编译器和并行处理形式的编译器。由于篇幅的限制，也为了表示的方便，我们对例子进行简化，忽略细节问题。

1. 多阶段的体系结构风格

编译器有五个阶段：词法分析、语法分析、语义分析、优化和代码生成，优化阶段是可选的。

多阶段风格识别下列体系结构元素：处理元素包括词法分析器、语法分析器、语义分析器、优化器和代码生成器；数据元素包括字符、单词、子句、相关子句、注释子句和目标代码；联接元素待定。

体系结构风格的形式由加权特性和体系结构元素的相关性表示。例如，优化器和注释子句在一起，它们都是可选的。另外，程序正文的字符与字符之间具有线性关系，词法分析器产生单词，语法分析器产生子句，单词由字符串组成，子句由单词串组成。子句和相关子句之间存在非线性关系。每个处理元素都有一组特性集，该特性用以对处理元素进行约束。例如，词法分析器输入字符，输出单词串；而且，在单词之间存在一定的次序。

2. 串行体系结构

古典的多阶段编译器体系结构是串行体系结构。它的每一阶段在下一阶段开始之前完成其功能，并且直接将数据元素从一个处理元素传至另一个处理元素。

串行体系结构的联接元素是过程调用和参数。

将单词中的标识符放进名字表(NT)中，将子句放入抽象语法树中(AST)，相关的子句形成抽象语法图(ASG)，优化时形成带注释的语法图(AASG)。图1给出了串行体系结构的过程视图，并显示了系统的数据流。

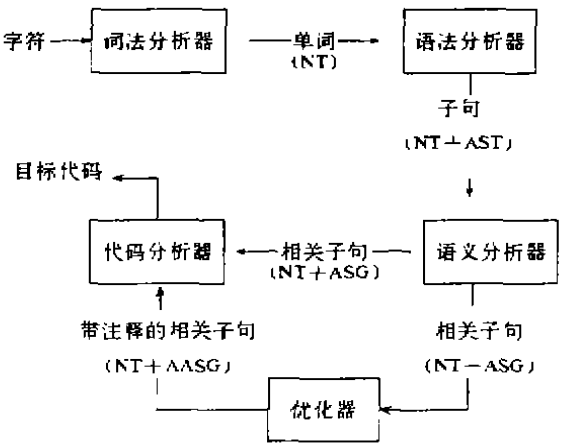

图1 串行编译器体系结构的过程视图

注意：从语义分析器到代码生成器有两条路径，仅有其中一条经过优化器。这说明本软件体系结构中，优化器是可选元素。

本例中，面向应用的特性如下：

has-all-tokens：是词法分析器分析源程序后产生的状态，是语法分析器开始工作的必要条件；

has-all-phrases：是语法分析器产生的状态，是语义分析器开始工作的必要条件；

has-all-correlated-phrases：是语义分析器产生的状态，是优化器和代码生成器开始工作的必要条件；

has-all-optimization-annotations：是优化器产生的状态，是代码生成器工作的可选前提。

串行编译器体系结构的数据视图如图2所示：

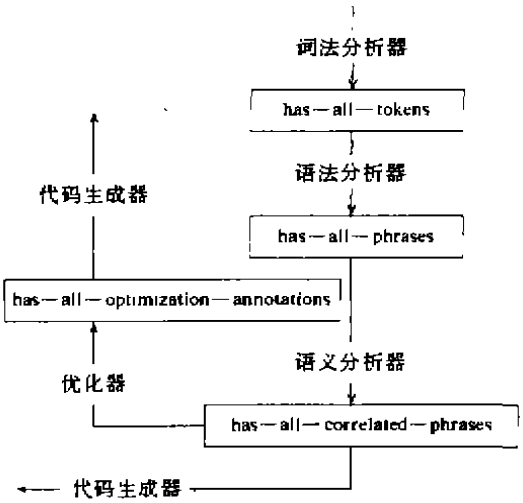

图2 串行编译器体系结构的数据视图

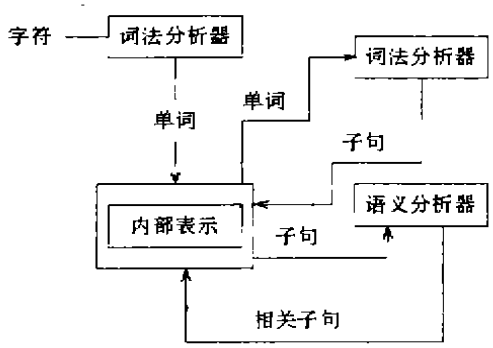

图3 并行处理的具有共享数据的编译器体系结构的部分过程视图

数据视图着重于特定的面向应用的特性，这对于每个数据体系结构的描述很重要，同时，过程视图着重于每个处理元素的功能特性。这两个视图是互补的。

3. 并行处理的具有共享数据的编译器体系结构

为尽可能地优化编译器的速度，提出了并行处理的具有共享数据的编译器体系结构：将处理元素作为独立进程对待，这些进程共享内部表达式；联接元素是共享数据，而每个处理元素处理特定的值。

图3说明并行编译器体系结构的部分过程视图。

内部表示具有的基本特性有：

no-tokens，
has-token，
will-be-no-more-tokens，
no-phrases，
has-phrase，
will-be-no-more-phrases，
no-correlated-phrases，
have-correlated-phrases，
all-phrases-correlated。

其中，单词和子句是提交的，而相关子句是累积的。

处理元素是并发的。本例中，采用并行路径表达式来描述基本特性间的关系。在并行路径表达式中，逗号表示串行，+号表示一次或多次重复，*号表示0次或多次重复，子表达式括在括号内。不同的并行路径表达式的同名特性处，就是同步点所在位置。

并行体系结构的联接视图（本例中，也为数据视图）：

并行体系结构的数据元素（单词，子句，和相关子句）的路径表达式：

1）(no-tokens，has-token$^+$)*，will-be-no-more-tokens，has-token*，no-tokens

2）(no-phrases，has-phrase$^+$)*，will-be-no-more-phrases，has-phrase*，no-phrases

3）no-correlated-phrases，(have-correlated-phrases)*，have-correlated-phrases

面向应用的特性（提交单词产生子句，再寻找相关子句，直至处理完所有子句）的路径表达式：

4）will-be-no-more-tokens，will-be-no-more-phrases，have-correlated-phrases

5）has-token$^+$，has-phrase

6）has-phrase$^+$，have-correlated-phrases

并行体系结构的过程视图：

词法分析器：(no-tokens，has-token$^+$)*，will-be-no-more-tokens

语法分析器：no-phrases，(has-toden$^+$，has-phrase)*，will-be-no-more-tokens，(has-token$^+$，has-phrase)*，no-tokens，will-be-no-more-phrases

语义分析器：no-correlated-phrases.(has-phrase$^+$.have-correlated-phrases)*.will-be-no-more-phrases.(has-phrases$^+$.have-correlated-phrases)*.no-phrases.have-correlated-phrases

串行体系结构和并行体系结构的有关应用的特性的对应表：

串行体系结构	并行体系结构
has-all-tokens	will-be-no-more-tokens
has-all-phrases	will-be-no-more-phrases
has-all-correlated-phrase	have-correlated-phrases

上面对应表中，可理解串行体系结构和并行体系结构间的可重用性。并行体系结构的主要要点总结如下：

1)处理元素与串行结构的很相似，但有不同的控制特性；

2)并行结构的形式比串行结构复杂；

3)并行结构的过程/数据/联接视图的相互依赖关系更为复杂；

4)面向应用的特性很适合于相似的体系结构。

七、软件体系结构的应用

1. 软件体系结构和需求分析与系统设计

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。

2. 软件体系结构和分析

一般而言，软件的规范说明除提供清晰和精确的文档外，还要对文档进行自动分析，暴露隐藏的各种问题。对软件的分析主要有两类：一致性和依赖性分析。

软件体系结构的规范说明支持以下几种分析：

1)体系结构风格约束的一致性分析。

2)一个软件体系结构对体系结构风格的满足情况。

3)软件体系结构约束的一致性。

4)软件设计对软件体系结构的满足情况。

5)软件体系结构和软件设计，软件体系结构和需求说明之间的依赖性分析。

6)软件体系结构和体系结构风格中软件设计和需求的非明显变化的测量，或是软件设计和需求中软件体系结构和体系结构风格的非明显变化的测量。

3. 软件体系结构和重用

设计者和程序员提高生产率的主要途径之一就是利用别人的工作，即应用和重用别的系统的组成部件，或标准部件。这样，问题的关键就在于重用部件的查找和部件的理解。

软件体系结构的焦点在于特性的识别和关系的识别。基于这点，在软件体系结构、软件设计、软件实现这三个层次上，可以选择合适的体系结构元素、设计元素或实现代码来满足各自的需要，来实现不同层次上的重用。例如，某软件系统的设计部件满足当前软件体系结构的某一特定的体系结构元素，而该设计部件的实现部分不满足要求，则重用设计，而不用实现。

软件体系结构级的部件可重用性大，实现级的部件可重性小，软件体系结构为重用问题开拓了一条道路。

八、结束语

通过对大型复杂的软件系统开发的软件过程的研究，发现软件体系结构在软件过程中起着极为重要的作用。但目前软件体系结构的研究尚处于初级阶段，我们以上的讨论是为软件体系结构的进一步研究打下基础。

软件的开发和软件系统的演化速度缓慢的原因也许在于：我们业已训练的是木匠和包工，而不是建筑师。

主要参考文献：Dewayne E. Perry & Alexander L. Wolf, "Foundations for the Study of Software Architecture", ACM SIGSOFT SOFTWARE ENGINEERING NOTES, Vol. 17 No. 4, Oct. 1992, pp40-52.