80-81,封底 软件开发 JSD法 程序转换 10

计算机科学 1994 Vol. 21№. 1

JSD的系统化转换与实现技术

TP311.52

冯 昕
（中山大学软件所 广州 510275）

徐永森
（南京大学 南京 210008）

A

摘要 本文主要介绍了我们开发的NUJSDS系统中关于JSD的规格说明的转换与最后实现技术。

关键词 JSD方法，规格说明，转换，实现，状态向量

一、引 言

由Jackon与Cameron于1978～1981年间共同创造出来的著名软件开发方法—JSD方法，与传统的软件开发方法相比有以下几个方面的特点：1)它不是从功能出发而是基于环境模型来开发软件。2)特别适用于实时系统的开发。3)将规格说明与实现严格分开，实现的考虑被推迟到很后的阶段。用JSD方法开发的软件具有可维护性和坚定性。欧洲一些组织利用它开发了不少实时系统，现在该方法越来越多地为人们所接受，利用。

目前关于JSD方法的支撑工具不少，但绝大多数都是支持JSD规格说明生成阶段，很少涉及最后的实现阶段，而这一阶段至关重要，它关系到开发的系统能否实际应用。这也是支撑工具的一个难点，现在关于JSD的支撑工具的研究重点已转移到实现阶段，在国内外已有的JSD支撑工具的基础上，我们开发了NUJSDS系统，它是依赖于JSD方法的支撑工具，该系统的最大特点就是不仅支持JSD规格说明的生成，而且支持最终代码的自动生成。

二、JSD转换技术

JSD的开发过程由规格说明阶段和实现阶段组成，规格说明阶段得到系统的规格说明，它由若干并行进程组成，每个进程都可以用结构树来描述出来，进程之间存在一些数据流关系，这种关系可以用系统结构图来描述。实现阶段将规格说明转换成可直接运行的代码。

由于每个实体及其模型都与一个进程相对应，而通常处理器数目有限，不可能有足够多的处理器使每个进程都能单独占有一个处理器，同时由于这些并行进程之间存在读写等待关系，进程运行的大部分时间都花在等待上，处理器利用率不高。基于处理器和运行效率方面的考虑，必然有若干进程共享一个处理器，所以转换是必然的。通过转换，将若干进程转换成一个顺序进程。

1. 转换中处理的主要问题

在JSD实现中，主要考虑数据库、程序运行环境的设计及代码的生成。重点在于代码的产生，即是将并行进程组成的规格说明转换成可执行的顺序语言程序，所以对于实现语言及操作系统，必须提供以下的手段：

(1)JSD进程控制机制：

- 组成单元的顺序执行机制；
- 循环执行机制；
- 带有返回的two-part选择；
- 带有quit的循环；

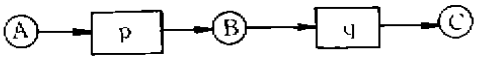
(2)存取和处理状态向量的手段。

(3)处理数据流联结的手段。

(4)程序转换机制。

2. 转换中的两个主要技术

(1)程序转换。例如有这样一个系统结构图：



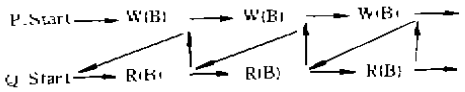
p、q 是进程程序，A、B、C 是数据流。由图中可以看出，P、Q 进程之间是通过数据流 B 联接起来，P 向数据流缓冲中写数据，Q 从中读数据。所以，在进程 P 中有若干写语句 write(B)，而在 Q 中有相应读语句 read(B)。P、Q 各自执行序列如下：

P：Start→W(B)→W(B)→W(B)→…

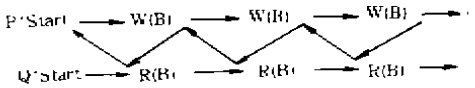
Q：Start→R(B)→R(B)→R(B)→…

有三种可能的转换：

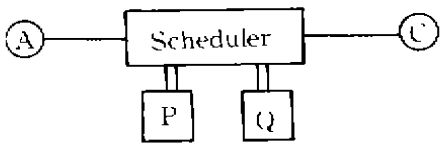
·Q 为 P 的子程序



·P 为 Q 的子程序



·使用专门调度程序



在第三种调度方法中，子程序 P、Q 的执行互为协同子程序，每次都在上次挂起的地方开始执行，协同机制的实现如下：

(a)在文本开始处有语句：

goto textpointer(文本逻辑指针)

textpointer 初值为 1。

(b)在读语句或写语句处：

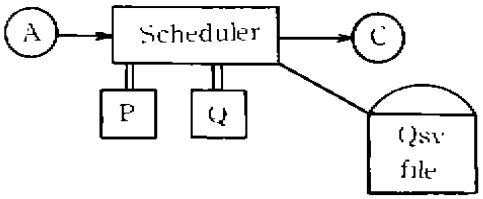
textpointer＝textpointer＋1；

return；

textpointer；

(2)状态向量分离。在现实世界中，有很多实体对应于相同的实体进程，例如一个银行的所有顾客实体都具有相同的进程序列。这些实体可以看成是实体进程的实例，即它们具有相同的文本，但具有不同的状态向量，这时没有必要让每个实例都拥有一个单独文本而占据存储空间。通过进程文本的单个拷贝以及状态向量的多个拷贝来实现状态向量的分离。

对于如下系统结构图：



在调用层中，必须执行以下操作：

Load sv of O-id INTO Qsv

Call Q(Rec,Qsv)

Q-id 用来标识进程实例。

在 Q 的子程序挂起之前，执行以下操作：

Store Q-sv as sv of Q-id

三、JSD 实现技术

由于我们希望代码的生成尽可能由支撑系统自动实现，就采用了第三种转换方法，即设置专门调度程序来调度各进程。实现中最主要的任务是(1)调度程序的生成，(2)变量的自动生成。

1 **调度程序的生成。**

在执行中的任意时刻，所有进程可分为三个集合：

1)一个正在执行的进程。

2)因为等待读数据而未得到满足的进程的集合。

3)从未执行过的进程以及挂起于写操作的进程的集合。

调度规则如下：

(1)当某进程运行到写数据操作时：

·如果读该数据的进程 B 在集合 2)中，且因为等待读该数据而挂起，则释放该进程

B 为运行进程，进程 A 放到集合 3)中。

· 如果读该数据的进程 B 在集合 3)中，则进程 A 放到集合 3)中，进程 B 被激活运行。

· 如果读数据进程 B 在集合 3)中，则进程 A 放到集合 3)中，进程 B 被激活运行。

(2)当某进程 A 运行到读数据操作时：

· 如果该数据流缓冲中不为空，则该进程在读数据后继续运行。

· 如果该数据流缓冲为空，且写该数据流的进程 B 在集合 3)中，则调度进程 B 来运行，并把进程 A 放到 2)中。

· 如果该数据缓冲为空，且写该数据流进程在集合 2)中，则说明存在死锁，提醒用户规格说明语义有错，需修改规格说明。

调度程序生成步骤：

(1)由 SD(STRUCTURED DIGRAM)→RWSD (READ-WRITE STRUCTURED DIGRAM)。

(2)扫描各进程的 RWSD，结合 SSD 得到调度结构图。

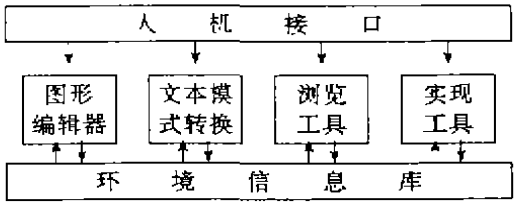
(3)转换调度结构图得到调度程序。

2. 程序变量的自动生成

从规格说明到源代码的转换中，NUJSDS 能实现变量的自动生成。

前面曾经提到，每个进程实例由进程文本和状态向量组成，状态向量是文本运行到某一时刻各变量的值的集合，其中包括文本逻辑指针。我们将每个进程的状态向量用一数据结构来表示，结构中包含若干域，域名可通过扫描结构图编辑过程中形成的.act 文件得到。结构类型的变量用进程文件名+sv 而得。这些结构变量都作为全程变量，在进程转换成的子程序内部，用状态向量中的状态向量名+sv 作为局部变量。

四、NUJSDS 系统框架图



该系统特点：

(1)该系统不仅支持规格说明的生成，而且支持从规格说明到可执行代码的自动转换。

(2)NUJSDS 不仅能保证规格说明语法正确，而且能进行部分语义检查。

(3)具有良好的用户界面。

(4)能指导用户正确使用 JSD 方法。

五、结束语

我们开发的 NUJSDS 系统在 286 微机上实现，于今年二月份通过鉴定，获得好评。目前正在对该系统进一步改变。期待将 JSD 方法和面向对象技术结合起来，使该方法更完美，并提供其支撑系统。

计 算 机 科 学

(1974 年 1 月创刊)
第 21 卷第 1 期 (双月刊)
1994 年 2 月 23 日出版
国内统一刊号：CN51—1239/TP
定价：3.00 元 国外定价：5 美元
邮发代号：78—68

编 辑 者：中国科学技术信息研究所重庆分所
顾 问：《计算机科学》审编委员会
出 版 者：中国科学技术信息研究所重庆分所
重庆市市中区胜利路 132 号
邮政编码：630013 电话：(0811) 350828
印 刷 者：中国科学技术信息研究所重庆分所印刷厂
总发行处：四 川 省 重 庆 市 邮 政 局
订 购 处：全 国 各 地 邮 政 局