一种基于 JSD 方法的规格说明语言及其支撑系统的设计与开发

刘 琳 徐永森
(江苏省公安厅通讯处，南京 210024) (南京大学计算机科学系，210008)

严 明
(解放军南京政治学院，南京 210000)

DESIGN OF A JSD-BASED SPECIFICATOIN LANGUAGE AND DEVELOPMENT OF ITS SUPPORTING SYSTEM

Liu Lin
(The Public Security Department of Jiangsu Province, Nanjing 210024)

Xu Yongsen
(Department of Computer Science, Nanjing University, 210008)

Yan Ming
(Nanjing Institute of Politics Science, Nanjing 210000)

ABSTRACT

Jackson System Development method, JSD, is a famous operational software development approach in the 80's. We designed a JSD-based graphical operational specification language-NUJSDL and a supporting systemNUJSDS which supports the use of the method in the practice of software development. NDJSDL has the characteristics of understandable, analysable and maintainable etc. The results of the different phases in JSD development procedures are described by the various mechanisms in NUJSDL separately. A specification in NUJSDL may

1990 年 5 月 23 日收到，1990 年 8 月 3 日定稿. 作者 刘琳，1989 年在南京大学获硕士学位，主要研究领域为应用软件. 徐永森，南京大学教授，主要从事软件方法学、软件工程学、软件理论方面的研究工作. 严明，1986 年毕业于南京大学，现任解放军南京政治学院助教，主要从事应用软件方面的研究工作.

be regarded as a prototype of the system to be developed. NUJSDS is an interactive and extensible system, which includes the tools of editing, analysis, specification generation and transformation. It supports the incremental construction of the system specification and the transformation from specification to software procedural description.

摘 要

Jackson 系统开发方法 (JSD) 是八十年代初提出的一种很有名的操作式软件开发方法. 为了支持开发者将其应用到软件开发实践中，我们设计了一种基于 JSD 方法的图形化的操作式规格说明语言 NUJSDL, 并开发了其支撑系统 NUJSDS. NUJSDL 语言具有易理解、可分析和可维护等特性，它提供了多种机制分别刻划 JSD 开发过程中各阶段的结果，并且用它书写的规格说明还可以作为待开发系统的一个原型. NUJSDS 是编辑工具分析工具规格说明生成工具和转换工具集成起来的交互式可扩充的系统，它支持规格说明在 JSD 思想指导下的开发，并能实现从规格说明到软件过程式描述的自动转换.

§ 1. 引 言

随着计算机科学与技术的发展，计算机应用领域越来越广，“软件系统从顺序过渡到并发再过渡到实时，其复杂性的增加程度十分可观” [5]，可是软件生产率却很低下，质量和可靠性愈来愈难以保证.

实时系统具有功能复杂、性能要求高、生命期长、系统需求变化频繁、与环境之间的接口复杂、难测试等特点. 因此按传统的方法从系统的功能着手进行系统开发比较困难. 这种系统要求有一种支持重复分析和重复实现的开发模式，即在开发过程中和系统投入运行后都应可修改系统的需求规格说明，然后再加以实现.

八十年代出现的一种新的软件开发模式——基于自动化的模式 [10] 就是为适应这一要求，并且有效地利用现有的软硬件工具而提出的. 在这个模式下，用户可以用一个适当的规格说明语言来描述和维护规格说明，维护仅仅是开发过程的继续，修改后的规格说明可作为待开发系统的原型，保证系统与用户需求相一致，最后再借助于适当的工具将高级的规格说明转换为具体实现. 这种由用户自己产生和维护的规格说明作为用户与开发者之间的接口，从根本上改善了软件开发方法.

这种新的开发模式包含了原型构造、转换实现和操作式规格说明等软件开发新思想. 其中代表性的方法之一是操作式方法.

操作式方法的基本思想是为所求系统建立一个操作模型，它具有以下几个特点：

(1) 首先建立环境的模型，然后进行系统功能开发；

(2) 将面向问题的考虑与面向实现的考虑严格分离；

(3) 提供一种操作式的规格说明语言；

(4) 可实现从转换后的规格说明到具体实现的自动生成.

目前国际上关于操作式方法的代表性工作有 PASILey 系统 [7], GIST 系统 [8], METAFOR 方法 [9] 和 JSD 方法 [2].

JSD 方法是 1982 年研制出来的一种系统开发方法. 由于其覆盖了应用软件系统开发的所有主要活动, 因此从一开始它就受到很高的评价且使用得很为成功. JSD 方法将系统开发的过程分为两个阶段, 即说明阶段和实现阶段. 具体地被分为六个步骤: 实体动作步、实体结构步、初始模型步、功能定义步、时间要求步和实现步. 其中前五步构成说明阶段, 最后一步构成实现阶段. 在说明阶段, 前三步建立系统主题的计算机仿真模型, 后两步则完成系统功能的开发以及进程间数据通讯的描述. 前三步称为模型步, 后两步为网络步. JSD 方法的详细介绍参看 [2]. JSD 方法的主要成就不仅在于开创性地进行了对客观环境及它与软件间的关系的研究, 而且还在于它明确了软件系统说明决策与实现决策间的界限.

为了更好地将 JSD 方法用于软件系统开发的实践中, 目前国际国内都有不少研究人员正在致力于 JSD 方法支撑系统的研制. 在分析和总结国内外有关工作的基础上, 我们设计了一种基于 JSD 方法的 NUJSDL 语言, 并构造了其支撑系统 NUJSDS, 该系统支持开发者用 JSD 方法进行软件系统的开发.

§ 2. 规格说明语言 NUJSDL

规格说明语言 NUJSDL 是一种基于 JSD 方法的描述语言. 在语言的设计中, 主要考虑了以下几个方面: 易理解性, 可分析性, 可维护性.

一方面由于“基于图形的语言的信息传递速度和整个规格说明的清晰程度都明显优于正文语言” [5], 且 JSD 方法本身提供了一整套图形表示来描述开发过程中各阶段的结果, 另一方面由于正文语言的描述能力较强, 可以描述图形无法描述的信息, 因此我们将 NUJSDL 设计成一种图形和正文相结合的语言.

一个用 JSD 方法开发出的系统规格说明是由多个顺序进程组成的进程网络. 顺序进程是一个结构简单的程序, 没有并发、平行和多任务概念, 在任一时刻只有一个动作发生. 由于系统是由多个这样的进程组成, 因此整个系统在某一时刻则可有多个动作发生. 这样就很好地刻划了平行性和分布式.

2.1 图形版本的 NUJSDL 语言

在 NUJSDL 语言的图形版本中, 用系统规格说明图 SSD(System Specification Diagram) 来描述进程网络结构, 而组成进程网络的各顺序进程的内部结构用结构图 SD(Structure Diagram) 来描述. 这样, 一个 NUJSDL 图形规格说明由两个层次组成: 网络层和进程层. 网络层是一个 SSD, 进程层是多个 SD.

让我们来描述一个简单的银行系统.

假设该银行系统中与目标系统相关联的实体是“顾客” (customer), 而其有关的动作有“立户” (invest), “存款” (pay-in), “取款” (withdraw) 和“销户” (terminate). 图 1 描述了“顾客”所发生和承受的动作在发生时间上的先后次序关系, 即顾客进程结构图.

一般来说, 结构图是树型的, 它用动态事件序列来刻划进程的活动. 在 JSD 方法中, 实体、模型、功能同等对待, 都看着顺序进程, 用 SD 刻划.

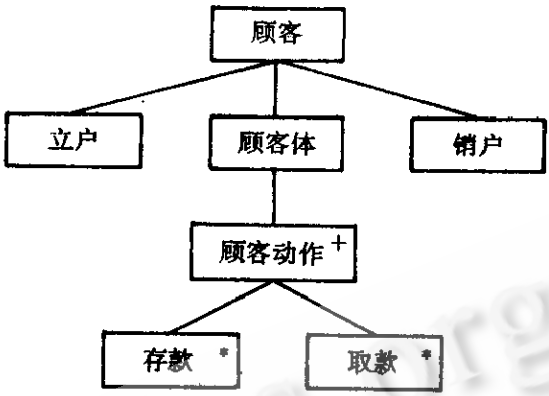

图 1 顾客进程的结构图

SD 中有两种成分：基本成分和复合成分．基本成分是不含子成分的成分，即图中的叶结点，它们是与该图对应的进程发出和承受的动作．复合成分有三种：顺序，选择和重复．每个复合成分都由若干子成分组成，子成分可以是基本成分也可以是复合成分．SD 中最大的复合成分是该 SD 本身，根结点的名是其所描述的进程的名． SD 中除根结点和叶结点外其它的中间结点的名都是为描述的目的而引入的． SD 中各成分的表示及其含义如下：

名称	图形方式	正文方式	含 义
基本成分	A	PELEM(A)	成分 A 是一个基本成分
顺序	A A1 ··· An	PSEQN(A: A1,···,An)	成分 A 由子成分 A1,···,An 组成，且按从左到右的顺序排列
重复	A A0*	PITER(A: A0)	成分 A 由零个或多个子成分 A0 组成
选择	A A1* ··· An*	PCASE(A: A1,···,An)	成分 A 在不同条件下分别由 A1,···,An 之一组成

由此表可知，图 1 的含义是实体顾客发出的第一个动作只能是立户，在此之后可多次执行存款或取款动作而不受什么限制，最后一个动作是销户．顾客体和顾客动作是为描述的目的而引入的．显然该 SD 描述了实体顾客作为银行用户的整个生命期．

该银行系统的网络结构如图 2 所示．

从图 2 中可以看出， SSD 由四种成分构成：矩形表示进程，圆形表示数据流，菱形表示状态向量，有向线段表示其它三种成分之间的关系． SSD 中允许四种联系方式，其表示法和含义如下：

名称	图形表示	正文表示	含　义
数据流联结	P — D — Q	DSCON1(P, Q, D)	进程 P 和 Q 通过数据流 D 联结起来
状态向量联结	P — P-SV — Q	SVCON1(P, Q, P–SV)	进程 P 和 Q 通过状态向量 P–SV 联结起来
输入	D — P	INPUT (D, P)	D 是进程 P 的输入数据流
输出	P — D	EMPUT(P, D)	D 是进程 P 的输出数据流

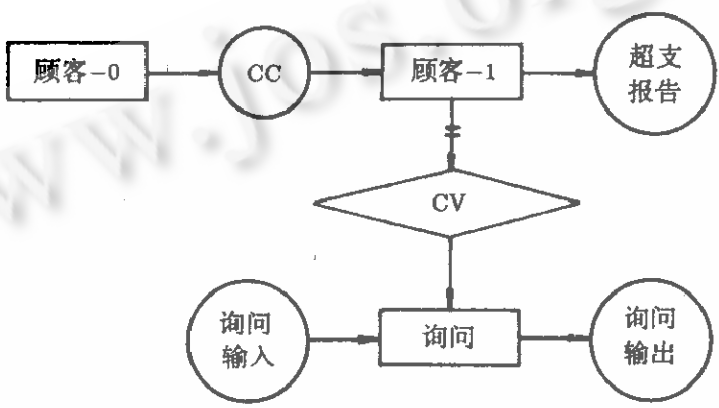

图 2　银行系统的系统规格说明图

图 2 中有的有向线段上有两条短杠. 这表示两进程间不是 1-1 对应关系. 进程间的关系有一对一、一对多、多对一和多对多关系. 这在正文表示中通过下标来描述. 在 SSD 中, 后缀为 0 的进程表示实体, 后缀为 1 的进程表示相应实体的模型, 其它进程为系统的功能进程. 图 2 中 CUSTOMER-0 和 CUSTOMER-1 通过数据流 CC 联结起来, 构成了系统的初始模型. 显然, CUSTOMER-0 和 CUSTOMER-1 都应对 CUSTOMER 做一定的修改以保证这种联结的实现.

由此可知, 图 2 的含义是 CUSTOMER-1 模拟 CUSTOMER-0 的行为, 进程 ENQUIRY 根据输入数据流 ENQUIRY INPUT 的要求查看 CUSTOMER-1 的一个或多个状态向量, 产生所求的输出, EXCESS REPORT 是 CUSTOMER-1 的输出数据流. 当某个 CUSTOMER 执行 withdraw 动作时, 若取款超出他的存款数, 则输出超支报告 EXCESS REPORT.

显然, 在初始模型基础上, 还可以开发出其它的功能. 这种功能的独立性对于系统的最初开发以及运行中的维护都是相当有利的.

图 2 加上各进程的 SD 就构成了该银行系统的完整图形版本的规格说明.

2.2 正文版本的 NUJSDL 语言

正文版本的 NUJSDL 比图形版本多了一些描述机制, 用以描述图形方式无法描述的

信息．正文版本的规格说明的结构为：

SYSTEM 系统名
  COMP 部分
  NETW 部分
  PROC 部分
END 系统名

其中 COMP 部分列出构成系统的所有顺序进程的名，它说明系统由哪些进程组成；NETW 部分对应于 SSD，刻划进程间的相互关系；PROC 部分由多个进程结构模块 PROCESS 组成，刻划各进程的内部结构，并给出必要的动作描述，COMP 部分出现的每个进程名都对应有一个 PROCESS 模块．每个 PROCESS 模块对应一个 SD，其结构为：

PROC 模块名
  ACTION 部分
  STRUCTURE 部分
END 模块名

其中模块名即为进程名；ACTION 部分用于描述实现进程间通讯或提供系统功能所必需的动作及条件；STRUCTURE 部分用于描述进程的结构．

这样用正文方式描述的该银行系统的规格说明为：

```
SYSTEM bank
  COMPONENT
    customer-0, customer-1, enquiry
  END;
  NETWORK
    DSCON1(customer-0, customer-1, cc);
    SVCONm1(customer-1, enquiry, cv);
    INPUT(enquiry input, enquiry);
    EMPUT(enquiry, enquiry output);
    EMPUT(customer-1, excess report);
  END;
  PROC customer-0
    ACTION  1: write(cc)  END;
    STRUCTURE
      PSEQN(customer-0: invest, 1, customer-body, terminate, 1);
      PITER(customer-body: movement);
      PCASE(movement: pay, withd);
      PSEQN(pay: pay-in, 1);
      PSEQN(withd: withdraw, 1)
    END
  END customer-0;
  PROC customer-1
    ACTION  1: Read(cc);  2: cc≠terminate;
        3: cc=pay-in;  4: cc=withdraw
    END;
    STRUCTURE
      PSEQN(customer-1: 1, invest, 1, customer-body, terminate);
```

```
      PITER(customer-body: 2);
      PSEQN(2: movement);
      PCASE(movement: 3, 4);
      PSEQN(3: pay-in, 1);
      PSEQN(4: withdraw, 1);
    END
  END customer-1;
  PROC enquiry
    ...
  END enquiry;
END bank.
```

§ 3. NUJSDL 的支撑系统 NUJSDS

NUJSDS 是在分析总结国内外现有工作基础上开发的一个支持 JSD 方法的系统，它不仅能作为系统开发的工具，支持用 JSD 方法来开发系统，而且能作为训练工具，训练用户学会应用 JSD 方法.

3.1 总体结构和各组成部分的功能

NUJSDS 的总体结构如图 3 所示. 其中单线箭头表示调用关系，双线箭头表示数据流向. 由图可知，该系统由人机接口部分、编辑器、分析器、规格说明生成器、转换器和数据库六个部分组成. 人机接口部分负责与用户的交互，以菜单方式供用户选择各种命令，从而调用不同的功能模块. 编辑器由图形编辑程序和正文编辑程序组成. 正文编辑程序提供了进行正文编辑时所需的各种功能. 图形编辑程序是面向 NUJSDL 的一种语法制导的编辑程序. 利用该图形编辑程序，开发者可将屏幕当作纸，将 Mouse(鼠标器) 当作笔来描述他所要开发的系统. 图形编辑程序由系统编辑模块和进程编辑模块组成，分别支持 SSD 和 SD 的编辑. 这两个模块都有编辑、删除和修改的功能. 用户可以根据需要给出图形方式或正文方式的 NUJSDL 规格说明，并在编辑过程中生成内部数据供分析器、规格说明生成器和转换器使用. 分析器对规格说明的语法语义进行分析检查，并向用户报告出错信息. 规格说明生成器把规格说明转换成与之等价的另一内部形式. 转换器把内部形式的规格说明转换为用模式逻辑写的软件过程式描述. 数据库用来存放开发过程中产生的各种信息和用户指南等文档资料.

3.2 NUJSDS 的特点

该系统提供了面向 NUJSDL 的图形化编辑程序，并具有语法制导进行编辑的功能，提高了所编辑的规格说明的语法正确性.

该系统是一个集成化环境，支持从规格说明编辑到过程式描述自动生成等一系列开发活动，提高了软件开发的自动化程度.

该系统是用模块化语言 Modular-2 实现的，具有可扩充性和易修改性.

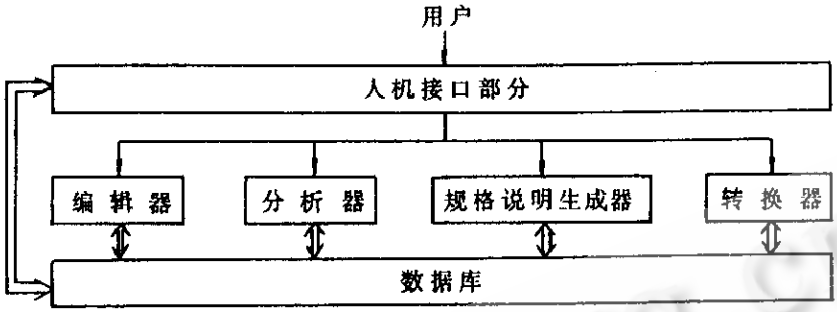

图 3 NUJSDS 的结构

§ 4. 结束语

NUJSDS 是用 Modular-2 编写的，现已在 IBM PC/AT 上完成第一阶段的工作．用该系统，我们已成功地开发了一些典型系统的规格说明．使用结果表明，该系统基本达到了预期的目标，可以支持用户用 JSD 方法进行系统开发．

目前国际上许多学者从 JSD 方法和 OOD(面向对象的设计)方法的共同原则出发，找到了两者的内在联系，即软件质量的关键在于用反映问题本身的结构的方法来构造问题的解．两者在描述客观现实环境时所用的概念也很类似，只是侧重点有所不同，各有其特点．因此将这种方法结合起来来开发应用系统是一种很有意义的尝试．有人正在尝试从 JSD 的图形方式规格说明直接生成 ADA 程序．无疑，这对提高目标系统的质量和可靠性，以及提高软件开发的自动化程度都是非常有意义的．这一目标的实现有待于功能更强的支撑系统的开发．

参考文献

[1] M. A. Jackson, Principles of Program Design, Academic Press, 1975.

[2] M. A. Jackson, System Development, Prentice-Hall, 1982.

[3] J. R. Cameron, JSP and JSD. The Jackson Approach to Software Development, IEEE Computer Society Press, 1983.

[4] J. R. Cameron, "An Overview of JSD", IEEE Trans. On S. E., Vol SE-12, 10, Feb, 1986, 222–240.

[5] J. K. Stoegerer, "规格说明语言的一种统一的处理方法"，(中译文)，计算机科学，1986, 1–13.

[6] 仲萃豪、叶农，"JSD 方法对软件工程研究的启示"，计算机科学，1988, 1. .

[7] Pamela Zave, "The Operational Approach to Requirement Specification for Embedded System", IEEE Trans. On S. E. Vol. SE-8, 5, May 1982, 250–269.

[8] R. M. Balzer, N. M. Goldman and D. S. Wile, "Operational Specification as the Basis for Rapid Prototyping", ACM SIGSOFT Software Engineering Notes, Dec., 1982, 3–16.

[9] C. L. Mcgowan and M. B. Feblowity, "The METAFOR Approach to Executable Specification", in Proc. of 3rd Int. Workshop on Software Specification and Design.

[10] R. Balzer, T. E. Chesatham, Jr., and C. Green, "Software Technology in the 1990's. Using a New Paradigms", Computer, Nov. 1983, 39–45.