软件开发 抽象逻辑结构图 图形化

计算机科学 1996 Vol.23 №.6

83-86, 90

抽象逻辑结构图及其应用*)

刘建宾 龚世生
(汕头大学计算机系 广东汕头515063)

TP311.52

摘 要 ALSD, Abstract Logic Structure Diagrams, is a new diagramming technique mainly for program representation, which is CASE-oriented, like-tree diagrams with multiple-level abstract capability and high understandability, and can be used in all phases of software development life cycle. This paper presents ALSD's formal definition, discrsses its characteristics, function and application.

关键词 ALSD(Abstract Logic Structure Diagrams), Program representation, Software diagramming tool

图形化技术作为人们清晰思维的语言和良好的通讯交流工具,在软件开发中一直发挥着重要作用,是软件表现技术的重要形式之一。迄今为止,已经出现了各种各样的程序图形表示法[1,2,6,7],然而使用现有的技术,在开发的不同阶段需要在多种不同表示之间进行转换,给开发各阶段的平滑过渡和一致性维护带来很大的困难。为此,我们提出了一种新的程序图形化表现技术——抽象逻辑结构图 ALSD。

1 形式化定义

1.1 几个基本集合和函数

我们首先定义几个基本的集合和函数,作为定义 ALSD 的基础:

① 概念结点类型集合 T_{Cn} = {+, *, o, ?, !, #, &, 空格}。

②逻辑结点类型集合 T_{ln} = {SEQ, WDO, FDO, RPT, IFT, IFE, CAS, WHN, OTW, CAL, SEG, OPE, OPB, DCL, DEF, ESC, NXT, RET, PAR, UND}。T_{Cn}, T_{ln} 各元素的含义见表1。

③自然语言语句集合 S_{nn} = {s | s∈自然语言语句}。

④受限自然语言语句集合 S_{dl} = {s | s∈S_{nn} ∧ verb(s)∈VDD ∧ noun(s)∈EDD ∧ conj(s)∈CDD}。其中 VDD, EDD, CDD 分别表示命令动词词典、程序实体名词典和连接词词典; verb: S_{dl} → VDD; noun: S_{dl} → EDD; conj: S_{dl} → CDD。

⑤目标程序语言的基本操作命令和表达式集合 S_{pl} = {s | s∈目标语言表达式 ∨ s∈目标语言基本操作语句}。其中:基本操作语句不包括结构化语句,GOTO 语句和标号语句。

⑥几个树上的函数:parent(t, n)返回树 t 中 n 结点的父结点,childnum(t, n)返回 n 结点的孩结点数,brothernum(t, n)返回 n 结点的兄弟结点数,eldest(t, n)为判断 n 是否为最长兄弟结点的逻辑函数。最长兄弟结点是兄弟结点中最后画出的结点。

1.2 ALSD 的定义

抽象逻辑结构图是一个四元组:ALSD = (t = (A, d), t_C = (A_C, dc), t_l = (A_l, dl), t_i = (A_i, di))。其中:t 表示由处理结点构成的集成 ALSD, t_C, t_l, t_i 分别表示集成 ALSD 的概念、逻辑和实现视图,A 是处理结点的集合,A_C, A_l, A_i 分别表示概念结点、逻辑结点与实现结点的集合。下面我们给出它们的定义:

结点分类	概念类型	逻辑类型	说 明	结点分类	概念类型	逻辑类型	说 明
顺序	上	SEQ	概括结点	未定义	空格	UND	未确定结点
可选	o	IFT	ifthen	块结点	!	CAL	模块调用
并发	&	PAR	并行处理			SEG	过程段
循环	#	WDO	while 循环	跳转	)	ESC	退出
		FDO	for 循环			NXT	继续循环
		RPT	repeat 循环			RET	返回
选择	?	IFE	ifthenelse	终结点	:	OPE	基本操作
		CAS	case			OPB	操作块
		WHN	情况分支			DCL	数据声明
		OTW	否则分支			DEF	数据定义

表1: ALSD 的处理结点类型

*)广东省高教局重点学科研究课题。刘建宾 讲师,主要从事软件工具与环境,应用软件的研究工作。龚世生 教授,主要研究方向:软件工程,图形图象处理等。

(1)集成 ALSD　是用 t=(A,d)表示的,由处理结点构成的树,A 是满足处理结点构成规则集 R_n 的处理结点的集合,$A\subseteq T_{cn}\times S_{nn}\times T_{ln}\times S_t\times S_t$,$S_t=S_{df}\cup S_{nn}$,$S_t=S_{pl}\cup S_{rn}$,$d:A\to 2^A$ 是满足下面定义的处理结点分解条件的树结点分解函数。

·处理结点分解条件:对 $a\in A$,d 满足如下的条件:

c1:若 $a[T_{cn}]\in\{:,),!\}\vee a[T_{ln}]\in\{OPE,OPB,DCL,DEF,CAL,SEG,ESC,NXT,RET,UND\}$,则 $|d(a)|=0$。

c2:若 $a[T_{cn}]=$"空格"$\wedge a[T_{ln}]=$null,则 $|d(a)|=0$。

c3:若 $a[T_{cn}]=$"空格"$\wedge a[T_{ln}]\neq$null,则 $|d(a)|\geqslant 0$,且 $a[T_{ln}]\in T_{ln}$。

c4:若 $a[T_{cn}]\in\{+,*,o\}$,则 $|d(a)|\geqslant 1$,且对 $a_j\in d(a)$,有 $a_j[T_{cn}]\in T_{cn}(j=1,\cdots,k)$。

c5:若 $a[T_{ln}]\in\{SEQ,WDO,FDO,RPT,IFT,WHN,OTW\}$,则有 $|d(a)|\geqslant 1$,且对 $a_j\in d(a)$,有 $a_j[T_{ln}]\in T_{ln}-\{WHN,OTW\}(j=1,\cdots,k)$。

c6:若 $a[T_{cn}]=$"?"$\vee a[T_{ln}]\in\{CAS,PAR\}$,则 $|d(a)|\geqslant 2$。

c7:若 $a[T_{cn}]=$"?"$\wedge|d(a)|=2$,则对 $a_1\in d(a)$,有 $a_j[T_{cn}]\in T_{cn}(j=1,2)$。

c8:若 $a[T_{cn}]=$"?"$\wedge|d(a)|>2$,则对 $a_1\in d(a)$,有 $a_j[T_{cn}]\in\{\perp,o,$空格$\}(j=1,2,\cdots,k)$。

c9:若 $a[T_{ln}]=$"IFE",则有 $|d(a)|=2$,且对 $a_j\in d(a)$,有 $a_j[T_{ln}]\in T_{ln}-\{WHN,OTW\}(j=1,2)$。

c10:若 $a[T_{ln}]=$"PAR",则有 $a_j\in d(a)$,有 $a_j[T_{ln}]\in T_{ln}-\{WHN,OTW\}(j=1,2,\cdots,k)$。

c11:若 $a[T_{ln}]=$"CAS",则对 $a_j\in d(a)$,有 $a_j[T_{ln}]=$"WHN"$(j=1,2,\cdots,k-1)$,$a_k[T_{ln}]\in\{WHN,OTW\}$。

c12:若 $(a[T_{cn}]\neq$"空格"$\wedge a[T_{cn}]\neq$null$)\wedge a[T_{ln}]\neq$null$\wedge|d(a)|\neq 0$,则对 $a_j\in d(a)$,有 $a_j[T_{cn}]\neq$null$\wedge a[T_{ln}]\neq$null。

(2)处理结点构成规则集 R_n　由 $T_{cn}\to T_{ln}$ 的映射规则集 R_{cl},$T_{ln}\to T_{cn}$ 的抽象规则集 R_{lc} 以及结点语义描述一致性规则集 R_s 组成,$R_n=R_{cl}\cup R_{lc}\cup R_s$,$R_{cl}=\{R_{cl}(x)|x\in T_{cn}\}$,$R_{lc}=\{R_{lc}(x)|x\in T_{ln}\}$。设 $a\in A$,若 $a[T_{ln}]\neq$null,$a[T_{ln}]$ 应满足 $R_{cl}(a[T_{cn}])$;若 $a[T_{cn}]\neq$null,$a[T_{cn}]$ 应满足 $R_{lc}(a[T_{cn}])$;若 $a[S_t]\neq$null,并且 $a_j[T_{ln}]\in\{SEQ,OTW,NXT,ESC,UND\}$,则 a 应满足 R_s。

①概念结点类型到逻辑结点类型的映射规则集 R_{cl}

·顺序结点映射规则集 $R_{cl}(\perp)=\{r1,r2,r3\}$

r1:parent(t,a)$[T_{cn}]=$"?"$\wedge$brothernum(t,a)$\geqslant 3\wedge$not eldest(t,a)$\to a[T_{ln}]=$"WHN"。

r2:parent(t,a)$[T_{cn}]=$"?"$\wedge$brothernum(t,a)$\geqslant 3\wedge$eldest(t,a)$\to a[T_{ln}]=$"WHN"$\vee a[T_{ln}]=$"OTW"。

r3:其它情况 $a[T_{ln}]=$"SEQ"。

·循环结点映射规则集 $R_{cl}(*)=\{r4\}$。

r4:$a[T_{ln}]=$"WDO"$\vee a[T_{ln}]=$"FDO"$\vee a[T_{ln}]=$"RPT"。

·选择结点映射规则集 $R_{cl}(?)=\{r5,r6\}$。

r5:childnum(t,a)$=2\wedge\exists$ n$\in$d(a)(n$[T_{cn}]\neq$"o")$\to a[T_{ln}]=$"IFE"。

r6:其它情况 $a[T_{ln}]=$"CAS"。

·可选结点映射规则集 $R_{cl}(o)=\{r7,r8,r9\}$

r7:parent(t,a)$[T_{cn}]=$"?"$\wedge$not eldest(t,a)$\wedge$(brothernum(t,a)$>2\vee$brothernum(t,a)$=2\wedge\forall$ n$\in$d(parent(t,a))(n$[T_{ln}]=$"o"))$\to a[T_{ln}]=$"WHN"。

r8:parent(t,a)$[T_{cn}]=$"?"$\wedge$eldest(t,a)$\wedge$(brothernum(t,a)$>2\vee$brothernum(t,a)$=2\wedge\forall$ n$\in$d(parent(t,a))(n$[T_{cn}]=$"o"))$\to a[T_{ln}]=$"WHN"$\vee a[T_{ln}]=$"OTW"。

r9:其它情况 $a[T_{ln}]=$"IFT"。

·跳转结点映射规则集 $R_{cl}(!)=\{r10\}$

r10:$a[T_{ln}]=$"ESC"$\vee a[T_{ln}]=$"NXT"$\vee a[T_{ln}]=$"RET"。

·块结点映射规则集 $R_{cl}(\|)=\{r11\}$

r11:$a[T_{ln}]=$"CAL"$\vee a[T_{ln}]=$"SEG"。

·终结点映射规则集 $R_{cl}(;)=\{r12\}$

r12:$a[T_{ln}]=$"OPE"$\vee a[T_{ln}]=$"OPB"$\vee a[T_{ln}]=$"DCL"$\vee a[T_{ln}]=$"DEF"。

·并发结点映射规则集 $R_{cl}(\&)=\{r13\}$

r13:$a[T_{ln}]=$"PAR"。

·未定义结点映射规则集 R_{cl}(空格)$=\{r14,r15,r16\}$

r14:parent(t,a)$[T_{cn}]=$"?"$\wedge$brothernum(t,a)$>2\wedge$not eldest(t,a)$\to a[T_{ln}]=$"WHN"。

r15:parent(t,a)$[T_{cn}]=$"?"$\wedge$brothernum(t,a)$>2\wedge$eldest(t,a)$\to a[T_{ln}]=$"WHN"$\vee a[T_{ln}]=$"OTW"。

r16:其它情况 $a[T_{ln}]\in T_{ln}-\{WHN,OTW\}$。

②逻辑结点类型到概念结点类型的抽象规则集

R_{lc}

·R_{lc}(SEQ)={r1'}

r1':a[T_{cn}]="+"∨a[T_{cn}]="空格"。

·R_{lc}(WDO),R_{lc}(FDO),R_{lc}(RPT)={r2'}

r2':a[T_{cn}]="*"∨a[T_{cn}]="空格"。

·R_{lc}(IFE),R_{lc}(CAS)={r3'}

r3':a[T_{cn}]="?"∨a[T_{cn}]="空格"。

·R_{lc}(WHN),R_{lc}(OTW)={r4',r5'}

r4':brothernum(t,a)=2→a[T_{cn}]="o"。

r5':brothernum(t,a)≥3→a[T_{cn}]="⊥"∨a[T_{cn}]=·R_{lc}(IFT)={r6'}

r6':a[T_{cn}]="o"∨a[T_{cn}]="空格"。

·R_{lc}(PAR)={r7'}

r7':a[T_{cn}]="&"∨a[T_{cn}]="空格"。

·R_{lc}(CAL),R_{lc}(SEG)={r8'}

r8':a[T_{cn}]="!"∨a[T_{cn}]="空格"。

·R_{lc}(OPE),R_{lc}(OPB),R_{lc}(DCL),R_{lc}(DEF)={r9'}

r9':a[T_{cn}]=";"∨a[T_{cn}]="空格"。

·R_{lc}(ESC),R_{lc}(NXT),R_{lc}(RET)={r10'}

r10':a[T_{cn}]=")"∨a[T_{cn}]="空格"。

·R_{lc}(IFT)={r11'}

r11':a[T_{cn}]="空格"。

④结点语义描述一致性规则集 R_n={r1",r2"}

r1":a[S_{ll}]≠null→a[S_{ll}]=a[S_{ll}]。

r2":a[S_{nn}]≠null→a[S_{ll}]=a[S_{nn}]。

(3)概念 ALSD　是集成 ALSD 在概念层上的表现形式,即 ALSD 的概念视图。它表现处理的概貌,不涉及具体细节,是用 t_c=(A_c,dc)表示的,由概念结点构成的一棵树。A_c={a|a∈a[T_{cn},S_{nn}]∧a≠null}是概念结点的集合,dc:A_c→2^{A_c}的概念结点分解函数。对 a∈A 且 a[T_{cn}]≠null,有 a[T_{cn},S_{nn}]∈A_c,若 a[T_{cn}]∈{;,|,.),空格},则 dc(a[T_{cn},S_{nn}])=0,否则对 a_i∈d(a),有 a_j[T_{cn},S_{nn}]∈dc(a[T_{cn},S_{nn}]),j=1,…,k。

(4)逻辑 ALSD　是集成 ALSD 的逻辑视图。它以一种自然且抽象的方式描述软件的工作原理,与具体实现语言无关,是一棵与概念 ALSD 结构一致、内容更为详细丰富、用 t_l=(A_l,dl)表示的树。A_l=A[T_{ln},S_{ll}]是逻辑结点的集合。dl=d[T_{ln},S_{ll}],是 A_l→2^{A_l}的逻辑结点分解函数。

(5)实现 ALSD　ALSD 的最低层抽象表示是用 t_i=(A_i,di)表示的实现 ALSD,它是集成 ALSD 的实现视图。因包含有足够的实现细节,程序源代码可以从该层的表示容易地生成出来。其中 A_i=A[T_{in},S_{ii}],是实现结点的集合。di=d[T_{in},S_{ii}],是 A_i→2^{A_i}的实现结点分解函数。

1.3 视图的图形表示法

集成 ALSD 是抽象逻辑结构图的内部表示,ALSD 的外部表现形式则是它的概念、逻辑和实现视图 t_c、t_l、与 t_i,我们用下面的办法来表现这三个树形图:给定 t_x=(A_x,dx)(x∈{c,d,i}),对每个 a∈A_x,用一个结点 n(a)表示 a,若 dx(a)=∅,则该结点分解结束,否则对 a_j∈dx(a)(j=1,2,…,k),按图1所示的方法用"L"形线依次把 n(a_1)…n(a_k)从上至下地连接到 n(a)的正下方,且保证各 n(a_j)处在同一垂直线上。最后形成一棵层次关系从左到右,弟兄关系从上到下的线形树图。

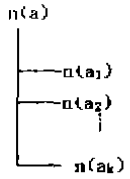

图1　分解图的基本画法

1.4 定义条件和规则的有效性

对抽象逻辑结构图形式化定义中给出的有关条件和规则,我们业已证明:对任一处理结点 a,a∈A,若 a[T_{cn}]满足结点分解条件,则满足概念结点类型到逻辑结点类型的映射规则集 R_{cl}的 a[T_{ln}]也一定满足结点分解条件;反之,若 a[T_{ln}]满足结点分解条件,则满足逻辑结点类型到概念结点类型的抽象规则集 R_{lc}的 a[T_{cn}]一定满足结点分解条件。因此,在定义中给出的结点分解条件和有关的结点构成规则是有效和无矛盾的。(详细证明略)

2　ALSD 的特点

ALSD 是在过去已提出并已普遍使用的多种程序图形化表示法基础上,结合当今对图形化技术发展的新要求而提出来的一种新的程序表示法。这种图具有如下的特点:

(1)使用二维树形结构图描述程序处理逻辑结构,用一个常用字符表示概念结点类型,用三个大写字母表示逻辑结点类型。用"L"形线作为结点间的连线,使图形的复杂性降到了最低限度;使用自然语言、受限自然语言来表达处理逻辑语义,使得表示法直观自然、结构清晰、层次分明,且具有高度可理解

性。

(2)多级抽象表现能力。它明确给出了程序在概念、逻辑和实现三个不同抽象层次的表示方法，并且将各层的表示统一在同一表示框架内。

(3)ALSD 支持结构化程序的所有基本控制构造和并发构造，表现法独立于具体的目标实现语言，适用于各种3GL 和4GL 的结构化程序表示，并可支持一种设计多种语言实现。

(4)ALSD 可以和多种设计方法，如面向功能的结构化设计方法、面向数据结构的设计方法、以及面向对象的方法密切配合，支持自顶向下、自底向上、和增量开发过程。

(5)ALSD 使用树遍法，使得表示法本身的扩充与维护非常容易，结点类型的增删和各类结点的表现内容都可根据需要进行增删以适应各种新的要求。

3 ALSD 的功能

ALSD 的表现力和特点决定了它具有多方面的功能和作用。

(1)ALSD 以一种简单、容易理解和图形的方式来表示程序的逻辑，成为设计人员清晰思维，充分发挥设计人员创造性，解决问题的辅助工具。

(2)ALSD 的简单性和高度的可理解性使普通用户能更多地介入开发与维护过程，增强了开发人员之间、开发人员与用户之间的相互理解与沟通，可成为信息开发部门的技术公约和标准。

(3)ALSD 的多级抽象表现能力使得它能用于软件生命周期各阶段，为各阶段的平滑过渡、文档及程序一致性自动维护在表现技术方面提供了良好的机制；使物理实现细节的抉择推迟到实现的最后阶段，利于发现早期设计错误，提高了软件可靠性，减少测试工时。

(4)ALSD 是面向 CASE 的图，它为 CASE 提供了新的程序表现形式，成为 CASE 的重要组成部分。在 CASE 的支持下，进一步完成内、外部文档及源代码的自动生成与一致性维护；进一步提高开发与维护的自动化程度，加快开发速度，减轻人员负担，降低费用。

(5)ALSD 是实现软件移植和再工程的新方法。实现层的 ALSD 可以生成源代码，用概念和逻辑 ALSD 描述的设计既独立于程序语言，又独立于计算机，因而可以在设计和实现两个级别实现软件移植。

(6)ALSD 为程序的理解与重用提供了新的途径。对现有源程序进行三级抽象，使程序容易理解。ALSD 的高度可理解性使它可以用来表示可重用部件，它的多级抽象性使同一重用部件具有可以在多个级别上进行重用的能力，提供了不同的灵活性。

4 ALSD 的应用

程序的表示 ALSD 将三个抽象层次上的程序表示内容集成在一个树形表示框架内，在保证统一、一致的前提下又可呈现出具有高度可理解性的概念、逻辑与实现三种外部表现视图，适用于整个软件生命周期内的程序表示。这种新的实用软件工具可在软件的开发、维护、软件的移植与重用及软件的再工程中发挥作用。

结构化规格说明 文本形式的规格说明通常是非结构化的，且表现能力也不足，即使使用结构化英语或结构化汉语写规格说明，所表现的处理逻辑结构也不够清晰直观，而使用概念 ALSD 表示的规格说明结构清晰、层次分明、错误更少、向设计阶段的过渡更为平滑自然。

数据结构的表示 概念 ALSD 可用于表示层次数据结构。数据与程序的表示形式统一，利于运用面向数据结构的设计方法导出程序的结构和完整程序源代码的生成。

行为过程 ALSD 能够表示软件开发过程、召开设计复审会、原型化过程、烹调等这样的行为过程。其好处在于能使基本的过程框架可复用，可根据每次使用的不同需要，把经常发生变化的细节加到过程框架上。由于具有明显的结构，使过程信息的组织及打印变得极其容易方便。

组织分类 ALSD 可以用来表示层次分类，如生物学的植物分类表示等。每个人都可以用 ALSD 把他们自己的思想和事实组织成层次图，每当有灵感时，就可以把想法或事项加到图上。这种保存了思想分类的图对从事写作等复杂思维的活动提供了有用的思想组织的工具。

程序设计的教学 ALSD 的一个重要特性是它们具有明显丰富的自然语言语义，它是在尽可能容易理解的原则下设计出来的，它提供了较为简单的方法来讲授结构化的规格说明与设计，可以成为程序设计和系统设计方面课程中最基本的教学内容。

结束语 抽象逻辑结构图是一种表现能力强，用途广泛，而又简单自然的实用工具，能够满足当今

(下转第90页)

①程序概念的抽象，概念抽象是以简单的高层概念代替复杂的低层概念。一个高层概念能代表低层概念群的一类。但是，程序概念的抽象不是纯省略性的抽象，而是以一定的知识的共享为背景的抽象。共享知识是实施程序概念抽象的主要难点。共享，意味着程序分析员与程序概念抽象器之间的共享，其难度可想而知。

②程序隐含信息的显性化。显性化的典型的例子是程序的控制流分析和数据流分析。这些信息是程序已有的，只是将它们显性化。显性化也是一种抽象，是纯省略的抽象。有些隐含信息隐藏颇深，使其显性非常困难，例如由多层指针变量而影响的变量的引用定值关系等。

③程序信息的可视化。可视化的作用是用实物演示弥补大脑思维中的表象演示的不足。在程序分析中可视化的主要困难是图迷航问题。

④程序信息的查询支持。查询可以看作对已显性化的信息再次使用纯省略的抽象。有效的查询依赖于信息的合理的存储和好的查询机制。

3. 结构程序代数

对于结构程序的自底向上的抽象的理论基础是结构程序代数，这个代数中的一条公理是代换公理。

定义 真程序是具有下述控制结构流图的程序：①有唯一的执行入口和唯一的执行出口；②对于每条语句，都有从执行入口到执行出口的执行路线通过。

代换公理：令 P 是 Q 的真子程序，【P】是 P 的语义，并令在 Q 中以 P' 代换 P 得到 Q'，则【P】=【P'】——→【Q】=【Q'】

定理 对于结构程序，存在一个结构程序代数。

结构程序代数是结构程序阅读、编写和验证的重要理论基础。

4. 程序注释

程序注释一般是非形式的，但形式化的注释有利于程序分析。注释分为两类：数据注释和真程序注释。真程序注释又分为状态注释和函数注释。程序证明中所使用的断言就是很好的状态注释。真程序对数据的作用相当于一个映射函数，该函数是很好的函数注释。

结束语 理想的程序分析支撑环境应当是与程序分析的专家模型相一致，从这一角度观察，现在的各类程序分析研究分别只反应了专家模型的不同的侧面。程序分析方法学提供了程序分析的方法和工具的理论依据。建立在专家模型和方法学基础上的程序分析环境将另行文介绍。（参考文献共10篇略）

（上接第86页）

对软件图形化表现技术的基本要求，我们相信它是目前程序图形化表示方面较为简单、能力较强的方法之一。它对于开发各阶段的平滑过渡、文档及程序代码的一致性维护、鼓励最终用户介入、新型 CASE 工具的研制等方面在程序表现技术方面提供了新的表示手段和基础。

目前，我们已经研制出基于抽象逻辑结构图的程序设计支援工具[6-9]，工具已能支持 foxpro 和 C++语言的程序开发。

参考文献

[1]James Martin, Carma Mcclure, Action Diagrams: Clearly Structured Specification, Programs, and Procedures, 2nd. ed. New Jersey: Prentice-Hall, 1989

[2]Leonard L. Tripp, A Survey of Graphical Notations for Program Design-An Update, ACM SIGSOFT Software Eng. Notes, 13(4), 1988

[3]S. P. Maj, Language Independent Design: An Introduction, Oxford: NCC Blackwell Limited, 1991

[4]Yaohan Chu, Software Blueprint and Examples, Lexington: D. C. Heath and Company, 1982

[5]James Martin, Principles of Object-Oriented Analysis and Design, New Jersey: Prentice-Hall, 1993

[6]M. A. Jackson, Principles of Program Design, London: Academic Press INC1975

[7]何克清，计算机软件工程学，武汉大学出版社，1983

[8]刘建宾，FOXPRO 程序设计支援工具 FPDST 的设计与实现，数据库研究与进展'95—全国第十三届数据库学术会议论文集，哈尔滨工业大学出版社，1995. 12

[9]刘建宾，C++函数开发工具 CFDST 的设计与实现，中国计算机科学技术新发展 - 中国计算机学会第九次全国学术会议论文集，西南师范大学出版社，1996. 5