基于时序逻辑软件构架形式化方法研究*)

任洪敏 朱 承 钱乐秋

(复旦大学计算机科学系软件工程实验室 上海 200433)

Research on Formal Specification of Software Architecture Based on Temporal Logic

REN Hong-Min ZHU Cheng QIAN Le-Qiu

(Department of Computer Science and Technology, Fudan University, Shanghai 200433)

Abstract Development based on software architecture is one of the most effective solutions to improve software quality and productivity, and minimize the difficulties of developing large and complex systems. Formal models and specifications of software architecture are one of the key research areas of software architecture. In this paper, the extension of XYZ/E Language based on set theory, called XYZ/E+ Language, is proposed, and a method of formalizing software architecture whit it is formulated. The formal method is distinguished by its ability to represent not only static properties but also dynamic behaviors, and its high-level abstraction and powerful expressiveness. Finally, through an example of formalizing software architecture of a specific system with XYZ/E+ Language, its powerful modeling ability and good practicality are demonstrated.

Keywords Software architecture, Software architecture modeling, Formalization, Temporal logic, XYZ/E

随着软件系统的规模和复杂性日益增加,人们已经超越传统的"算法+数据结构=程序"的软件设计模式,软件系统的整体结构,即软件构架(Software Architecture)成为软件设计的重心和难点。运用形式化方法描述大规模、复杂软件系统的构架,能够阐明软件系统的真实内涵,有助于系统设计人员和相关人员更准确地了解系统,达成共识,避免误解。但更重要的是,形式化描述软件构架奠定了利用数学方法对软件构架的属性进行严密分析、推导和验证的基础,如活性(liveness)和公平性(fairness)推导[1]。

时序逻辑语言 XYZ/E 基于一阶线性时序逻辑,既是一个时序逻辑系统,又是一种程序语言,能在统一框架之下,既描述软件系统静态语义,又描述软件系统动态语义,支持逐步求精和快速原型等软件工程方法,能在不同抽象层次上对系统进行形式化描述,适合描述软件构架[2]。

在我们运用 XYZ/E 语言形式化建模软件构架的时候,发现易于陷入繁琐细节,而如果运用其所提供的抽象机制,但又易丢失系统应该表达的信息。其原因如,XYZ/E 语言虽然对数据类型进行了一定的扩充,提供了字符串、栈和队列等数据类型,但仍然基本遵循 PASCAL 程序设计语言提供的数据类型[3]。PASCAL 程序语言提供的数据类型面向软件编码和实现,而软件构架的建模和描述要求高度抽象,把握系统的本质特征,忽略细节,能够简明表达数据和其间的复杂关系。

形式化规格说明语言 Z 语言[4]提供了丰富的类型系统和建模结构,具有很强的表达能力和抽象建模能力,并为广大软件工程师熟悉,国内外众多学者对其进行了长期研究,有丰富研究成果可以直接利用。故我们决定运用 Z 语言提供的基于集合论的建模结构扩充 XYZ/E 语言,把集合论中相关表示和运算有机融入 XYZ/E 语言整体框架之中,增强 XYZ/E 语言建模能力和抽象能力,方便软件构架建模。

论文在介绍相关研究工作基础上,首先系统介绍了 XYZ/E 语言基于集合论的扩充和运用其对软件构架进行形式化建模的方法,然后运用该方法对一个具体软件系统构架进行形式化建模,表明了该方法具有强的表达能力和良好实用性,最后是全文总结。

1 相关工作

软件构架是决定大型、复杂软件系统质量的重要因素,运用形式化方法描述软件构架能够避免采用自然语言、框图等非形式化方法描述而带来的误解,进而奠定分析、验证和推导基础。因此,自从软件构架学科领域提出以来,国内外众多学者就展开了运用形式化方法描述软件构架的研究。

Wright[5]采用 CSP(Communicating Sequential Process)形式化描述软件构架,CSP 适合描述、分析软件构架的动态行为特性,如死锁性。Darwin[6]用 π 演算系统对系统的行为进行建模,该模型擅长于描述动态软件构架,即构件和连接器的组织结构可以在系统运行时动态变化,并利用 π 演算的强类型系统进行静态检查。文[7]采用 Z 语言描述软件构架风格,该方法运用一组从语法描述域到软件构架风格语义域的语义函数描述软件构架。Z 语言类似指称语义,适合描述软件构架的静态性质,但却难以描述和分析软件构架的动态特性。

XYZ/E 语言是一种面向软件工程的时序逻辑语言,能在统一框架之下描述和分析软件构架静态语义和动态语义。基于其诸多特点,文[8~10]提出和阐述了运用 XYZ/E 语言建模软件构架、构架风格及相关的图形描述和表示。

本文吸收他们研究工作的思想和特点,对 XYZ/E 语言作基于集合论的扩充,增强其形式化描述和建模软件构架能力,并系统分析,得出运用其建模软件构架的方法。

2 XYZ/E 语言基于集合论的扩充

XYZ/E 语言的理论基础是一阶线性时序逻辑。一阶线性时序逻辑是在一阶逻辑基础上扩充相应时序算子得到,常见时序算子包括 $O(下一时刻)、□(所有时刻)、◇(某一时刻)等,故 XYZ/E 语言能够建模和研究系统动态行为。形式规格说明语言 Z 语言的理论基础是一阶逻辑和集合论,故它强调

*)本文研究得到上海市教委重点学科建设项目资助。任洪敏 博士生,主要研究领域为软件复用。朱 承 博士生,主要研究领域为软件复用。钱乐秋 教授,博士生导师,主要研究领域为软件自动化、软件复用、软件自动测试。

描述性思维而避免过程性思维，具有高度抽象能力和强的描述能力。

XYZ/E 语言和 Z 语言都具有一阶逻辑理论基础，对 XYZ/E 语言作基于集合论的扩充，增强抽象能力和描述能力，既有助于实现 XYZ/E 语言的目标：在统一框架之下，在不同抽象级上描述系统的静态性质和动态行为，又便于运用其进行软件构架形式化建模。

论文从三个方面探讨 XYZ/E 语言基于集合论的扩充。扩充后的语言称为 XYZ/E+语言。XYZ/E 语言和 Z 语言的相关语法和语义分别参见文[3,4]。

2.1 扩充 XYZ/E 语言数据结构

XYZ/E 语言数据类型基于 PASCAL，而 Z 语言提供了丰富的类型系统和其上的运算。Z 语言的幂集类型、笛卡尔积类型、序列、包、关系、函数和模式皆可引入作为 XYZ/E 语言的数据类型。同时允许用户运用方括号定义基本数据类型(Basic Type)，而不关心其内部结构。现举两例。

例 1 对文具用品管理系统构架建模，需一数据结构方便表达每件文具用品的库存数量，则可简明定义如下：

```
%TYPE [
        [ITEM]; // 定义 ITEM 为基本数据类型，表示文具用品
                   类型.
        STOCK=ITEM ↛N; // 即用文具用品到自然数的部分
                          函数描述其库存量.
    ];
%VAR [MyStock: STOCK] //定义库存变量
```

例 2 信号处理系统中信号可方便地定义为：

%TYPE [SIGNAL=Time→Volts] //

用时间到电压的函数表示信号

Z 语言中的模式(Schema)由变量声明和限制这些变量的谓词两个部分组成。XYZ/E+语言运用模式的水平表示形式，即：$S=[D_1;D_2;\cdots;D_m|P_1;P_2;\cdots;P_n]$。$D_1;D_2;\cdots;D_m$ 为变量声明，$P_1;P_2;\cdots;P_n$ 为约束谓词定义。模式语义相当于受到限定的记录数据类型，故模式可作数据类型。通用(Generic)模式定义提供了定义数据模板的功能，它与包块、代理机构结合，能够对类似 C++语言中模板类(Template Class)的软件模块进行建模。

2.2 扩充 XYZ/E 语言条件元

条件元(conditional element，简称 c.e.)是 XYZ/E+语言基本命令形式，表示状态转换和控制流，其形式为：

$$LB=y\wedge R\Rightarrow @(Q\wedge LB=Z) \quad (1)$$

及 $$LB=y\wedge R\Rightarrow \$O((v_1,\cdots,v_n)=(e_1,\cdots,e_n)\wedge LB=Z) \quad (2)$$

其中@表示 $O(下一时刻)或◇(某一时刻)，"R"、"Q"为一阶逻辑公式，分别称为条件元的条件部分与动作部分，⇒表逻辑蕴涵。(2)式是(1)式特殊情形，即用并行赋值等式作为动作部分。

条件元的扩充包括条件部分扩充和动作部分扩充。条件部分扩充指条件部分能够包括描述集合与集合之间关系或元素与集合之间关系的集合谓词。常用集合谓词包括："属于∈"、"包含⊆"、"真包含⊂"等。

动作部分扩充指动作部分能够运用集合相关运算作为谓词表示动作和状态转移，具体包括集合运算如并、交、差等，及相关数据类型运算，如关系的复合、限定、限定减、映象，函数的迭加、序列的连结等。特别允许集合赋值运算，即把一个赋值运算符右边的集合或集合表达式赋值给左边的集合变量。

例3 举例说明条件元扩充。对图书馆管理系统建模，图书借阅包括读者借阅登记、最后借阅该书读者登记，如果读者非法、或该书不存在，则打印错误信息。运用包块建模该图书管理系统。

```
%PACK[ LibrarySysterm:
  %TYPE[
          [PERSON]; //基本类型，表示人员类型
          [COPY] //基本类型，表示图书复本类型
        ];
    %VAR[
          readers: P PERSON; //读者定义
          availablebook :P COPY; //可借阅图书
          checkoutbook: COPY ↛ PERSON; //图书借阅
              信息
          lastcheckout: COPY ↛ PERSON //最后借阅图
              书的读者信息
          ];
  /*初始化部分和其它相关部分省略*/
  /*图书借阅过程*/
  %PROC[Checkoutbook( ) ==
    [%LOC [ reader : PERSON; book: COPY ];
    %ALG [
          LB=START-Checkoutbook⇒ $OLB=l1;
          LB=l1⇒ $OSREAD(reader, book) ∧ $OLB=l2;
          //读入读者和欲借读书.
          /*如果读者合法和该书可借，则修改借阅信息和最
          后借阅信息*/
          LB=l2 ∧ reader ∈ readers ∧ book ∈ availablebook⇒
              $O(
              ≫(availablebook= availablebook\{book},
                checkoutbook = checkoutbook ⋃ {reader ↦
                    book},
                lastcheckout = lastcheckout ⊕ {reader ↦
                    book})
               ∧ $OLB=STOP);
          LB=l2 ∧ reader ∉ readers⇒ $OSWRITE("Invalid
              reader") ∧ $OLB=STOP;
          LB=l2 ∧ book ∉ availablebook⇒ $OSWRITE("The
              book checked out") ∧ $OLB=STOP
          ]
  ]]
```

2.3 单元约束部分扩充

XYZ/E 语言中，单元约束部分运用逻辑语言、可计算函数或代数公理直接表示问题的精确语义、相关约束或定义特殊谓词用于单元描述部分，该部分可运用 XYZ/E 系统中的工具 XYZ/PROT 用类似 Prolog 的执行机制求值。单元约束部分扩充指在约束部分运用集合谓词表达约束条件，利用函数、关系、序列、包及其相关运算定义函数或过程，或者直接采用形如 f=A→B、f=A↛B(A、B 为任意集合)的方式抽象定义函数。抽象定义的函数不能求值，在软件求精或构件实例化时给出其精确定义。

抽象定义的函数适合软件构架建模，因为构架建模注重系统的整体结构，而非构件具体实现算法。其抽象程序高，但同时又保留相当丰富的信息，它给出了状态变换的起始空间、结束空间，利用函数种类给出状态变换性质。这些信息皆能用于和约束系统求精及构件实例化。

3 XYZ/E+形式化建模软件构架方法

XYZ/E+语言是 XYZ/E 语言的扩充，基于我们运用 XYZ/E+语言形式化建模软件构架的研究和实践，系统得出了运用其形式化建模软件构架三个要素(构件、连接器和系统配置)的方法。

3.1 XYZ/E+建模软件构件

运用 XYZ/E 语言提供的程序构造单元，包括过程(Procedure)、进程(Process)、包块(Package)、代理机构(agent)建模软件构件，该四种构造单元分别建模不同属性的软件构件。过程对被动执行的具有单个完整功能的软件构件进行建模。进程对能够主动执行和并发执行的软件构件进行建模。包块对围绕一数据结构封装一组运算而成的软件构件进行建模。代理机构由一数据包块和一与之相匹配的进程部分组成，实现静态语义和动态语义的联结和组合，对能够主动执行和通

信的包块构件进行建模。

在四类构造单元的 WherePart 部分，运用逻辑合式公式和集合谓词表达逻辑限制和制约条件，即构件的静态属性。在它们的主体部分运用 XYZ/E＋语言提供的条件元和三种控制结构描述构件的动态行为。结构化的 XYZ/SE 控制结构、产生式 XYZ/PE 控制结构因其简明和抽象程度高更加适合描述构件的动态行为。需要特别指明的是，XYZ/E 中表示不确定性的命令语句，即选择语句：

$LB=y_i \wedge R \Rightarrow \$O[Cond1 \triangleright ExeAct_1, \cdots, Condk \triangleright ExeAct_k]$

具有很强的描述和表达能力，在构件建模中经常运用。

3.2 XYZ/E＋形式化建模连接器

XYZ/E＋语言提供过程调用、数据共享、同步和异步的通道通信方式，直接支持常见构件之间的连接。其他构件之间的连接方式可以运用这些基本的连接机制和运用 XYZ/E＋语言描述其胶水代码(glue code)进行建模。如在文[9]中提供的数据流连接方式：它由存放数据的缓冲区(buffer)和向缓冲区发送、读取数据的操作 in/out 组成，其定义如下：

$In==[LB=y \wedge (BufLen+Indata \leqslant MaxLen) \Rightarrow \diamond(Buf=Buf+Indata \wedge \$OLB=z)]$

$Out==[LB=u \wedge (BufLen-Outdata \geqslant 0) \Rightarrow \diamond(Buf=Outdata+\$Obuf \wedge \$OLB=v)]$

$Data\text{-}flow==\|[in;out]$

3.3 XYZ/E＋形式化建模构架配置

构架配置是指构件通过连接器连接形成的一个软件系统的构架及约束。XYZ/E＋语言运用三类程序结构即：基于对象的程序、面向对象的程序和分布式程序建模软件构架配置。基于对象的程序由主块(Main Block)和包块组成。主块定义进程和过程模块。面向对象的程序其程序结构不同于基于对象的程序，不存在主块，亦无独立包块，主要成分是进程与代理机构。分布式程序最主要的特点是：一组程序在网络上分布运行，相互通信，合作求解某一问题，但不存在由某一用户书写的、在某一台计算机上运行、包含相应并行语句的“母程序”。故在 XYZ/E＋中系统构架配置定义如下：

ArchConfig==OBProgram | OOProgram | DProgram

在三类程序结构的 WherePart 部分运用合式逻辑公式和集合谓词表达整个构架的约束条件，在其主体定义部分，即 ProBody 部分进行构件和相关连接器实例化，并运用连接器连接构件。

4 XYZ/E＋形式化建模软件构架实例

软件构架形式化建模是软件构架研究的重要课题，是软件构架分析、验证和求精的基础。论文运用 XYZ/E＋语言形式化建模一个具体软件系统的构架，结果表明 XYZ/E＋语言具有强的建模能力和良好实用性，具体如下。

已知一示波器软件系统包括四个功能模块[1]：信息耦合、信号获取、信号变换、信号显示。每个功能模块皆有操作控制接口，具体构架见图1。

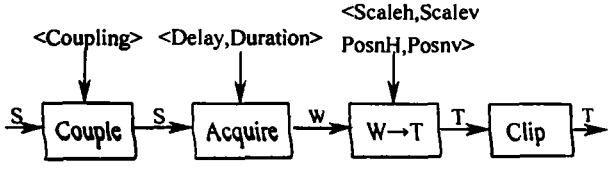

图1 示波器系统软件构架

该软件系统构架 XYZ/E＋建模如下。

```
/* 应用基于对象程序建模整个系统构架 */
%OBPROG Oscilloscope==□[
%TYPE [
   [TIME]; //给定基本时间类型
   [VOLT]; //给定基本电压类型
   SIGNAL=TIME→VOLT; // 利用部分函数定义信号类型
   COUPLING=DA|AC|GND; // 信号耦合控制类型
   TRACE=Z→Z]; // 定义显示图形类型为横、纵坐标的映射
%VAR [
   c: COUPLING;
   delay,dur : float;
   scaleH : float;
   scaleV, posnV : VOLT;
   posnH: INT ];
/* 应用进程模块建模各个构件 */
%PROS [
   /* Couple 读入信号，并根据耦合类型，输出相应信号 */
   Couple (%CHN in(signalproducer : NM, * ),out( * ,Acquire:
          NM))==[
          %STM [
            %LOC [x : SIGNAL];
            LB1=START-Couple$OLB1=|0;
            LB1=|0∧in ?$⇒O ( LB1=|1 | LB1=STOP );
            LB1=|1⇒$O( in ? X∧LB1=|2);
            LB1=|2∧c=DC⇒ $O(out ! x∧LB1=|0);
            LB1=|2∧c=GND⇒ $O(out ! 0∧LB1=|0);
            LB1=|2∧c=AC⇒ $O(out ! (λt:TIME. x(t)-dc(s)
             ∧LB1=|0))]];
   /* Acquire 反复根据给定时间，延迟 delay 之后，输出时段长为
   dur 的一段信号 */
   Acquire ((%CHN in(signalproducer : NM, * ),out( * ,WT:
          NM))==[
     %STL [
        %LOC [x : SIGNAL; trig: TIME];
        LB2=START-Acquire⇒ $O(Read(trig)∧LB2=|0);
        LB2=|0⇒ $O( in ? x∧LB2=|2);
        /* 应用关系定义域限定运算得出相应输出信号 */
        LB2=|2⇒ $ ⇒O( out ! { t:TIME|trig+delay≤t∧≤trig
             +delay+dur}◁ x∧LB2=|3);
        LB2=|3∧in ?⇒ $O(LB2= START-Acquire | LB2=
        STOP )]]
];
/* 其余模块的建模省略。程序主体部分进行构件实例化和连接 */
%ALG [
   LB=START-Oscill⇒ $OLB=|0;
   LB=|0⇒ $O aCouple==Couple{1}∧aAcquire==Acquire
   {1};
   //此为构架规约，并行语句应允许部分通道未赋与实参，表示其
   为与外界的接口 */
   LB =|1⇒|| [aCouple(%CHN in( * ,aAcquire) | out( * ,
              Acquire);aAcquire(%CHN out(aCouple, * ) |
              in(signalproducer, * ))];
   LB=|2⇒$OLB=STOP ]].
```

结束语 软件构架形式化研究和建模是软件构架研究的重要领域。XYZ/E 语言其理论基础是一阶线性时序逻辑，能以统一形式表示静态语义和动态语义，支持逐步求精软件工程方法。论文对 XYZ/E 语言进行基于集合论的扩充，集合论中相关的表示方法和运算有机地融入 XYZ/E 语言整体框架之中，增强其抽象能力和建模能力，既有助于实现 XYZ/E 系统目标，又能更好描述和建模软件构架。

在此基础上，论文系统地讨论了运用其建模软件构架包括构件、连接器、系统配置的方法。该方法既能够描述软件构架的静态属性，又能建模软件构架的动态行为，具有高度抽象能力、强的表达能力和良好实用性。

在进一步的研究工作中，我们将致力于 XYZ/E 语言基于集合论扩充之后，其带来的语义问题、程序验证问题和软件求精问题的研究，并运用 XYZ/E＋语言建模更多软件系统实例。

参 考 文 献

1 Shaw M, Garlan D. Software architecture: perspectives on an emerging discipline. Prentice Hall, Inc., Simon & Schuster Beijing Office, Tsinghua University Press, 1996

2 唐稚松. XYZ 系统的目的、意义、作用与应用. 软件学报，1999，10(4)：337～341

(下转第31页)

在移动环境中，实现事务绝对时间正确性的代价很大，例如，一个 MRTT 访问了一个实时数据对象的最新版本，然后向一个移动客户机发送请求要访问另一个数据项的最新版本，由于网络断接，后者可能要经过一段延迟才能得到，在这期间，可能又生成了前面访问数据对象的新版本，为保持绝对时间一致性，事务必须重启。考虑到这些情况，我们对绝对时间一致性做了一定程度的弱化，只要求事务提交时所有访问数据的版本"足够新"。

定义3 设事务 T 启动时间为 st，提交时间为 ct，用户所允许的"过期度"(staleness)为 Δ，读取数据集为 R 的事务如果

$$\exists\ t,(st-\Delta)\leqslant t\leqslant ct,\forall\ O\in R,LTB(O)\leqslant t\leqslant UTB(O).$$

则称该事务是相对时间正确的。

对相对时间正确的事务，并不要求事务提交时事务所读取的所有数据项都是有效的，而只要求所有数据项在$(st-\Delta)$之后的某个时间点上是有效的。

从上面定义可以看出，相对时间正确的事务不允许读取在$(st-\Delta)$之前某个时间点无效的数据。

4 移动实时事务处理

4.1 移动实时事务

定义4 一个移动实时事务的形式化定义为一个4元组：

$$T=(OP,D,<_T,\Delta)$$

其中 OP 是事务的操作集；D 是事务的截止期；$<_T$ 是事务中操作的一个偏序关系；Δ 是事务能够容忍的最大数据陈旧度(staleness)，数据的陈旧度定义为当前时间与该数据的有效期上界的差值。如果该差值小于 Δ，那么事务就可以使用该数据。

一个 MRTT 的执行可能需要访问固定主机和移动客户机上的数据，下面我们将介绍基于"影子"事务的事务执行模型，使得在移动环境中事务截止期和时间正确性要求能够更好地得到满足。

4.2 基于影子事务的事务执行模型

考虑到在移动网络环境中，移动客户机上数据访问延时是事务无法在截止期前提交的一个重要原因。影子事务模型的主要目的是减少事务数据访问延迟。

一个移动实时事务 T 在由移动客户机生成之后，提交到该移动户客机当前所在子单元对应的基站(一个固定主机)上，当事务被基站接受后，在基站上生成该事务对应的影子事务(Image Transaction，IT)，生成的影子事务由一些子影子事务(Sub Image Transaction，SIT)构成。每一 SIT 负责从各不同的移动客户机或固定主机上预取 T 执行所需的所有数据。

定义5 一个移动实时事务 T 的影子事务 IT(T)定义成一个3元组：

$$IT(T)=(P',D,\Delta)$$

其中 $P'=\{p_i'|p_i'$ 是事务 T 中的数据访问过程 p_i 的数据预取过程$\}$；D 是映像事务的截止期；Δ 是影子事务所能容忍的数据最大陈旧度。

影子事务 IT(T)的截止期和陈旧度分别是 T 的截止期限和陈旧度，Δ 用来决定影子事务将哪些版本从固定主机或移动客户机预取过来。我们用截止期最早最优先(earliest deadline first，EDF)方法确定 MRTT 的优先级，为照顾数据预取过程，所有影子事务被赋予比 MRTT 更高的优先级，而不同影子事务之间的调度方式是该映像事务的截止期限越小则优先级越高。需要注意的是影子事务的执行是并行的，所以预取过程并不具备原事务中操作的顺序关系。

对于 MRTT 中的每一个操作 p，p 访问的读数据集用 DS(p)表示，为减少 p 操作的数据访问延迟，$\forall\ p\in$MRTT，在 p 执行以前 DS(p)中的所有数据都必须已预取到该移动客户机对应基站上，一个影子事务对应一个包含操作 p_i' 的集合，其中对每个 $p_i\in T$，p_i' 具有下列的属性：

(1)$DS(p_i')\supseteq DS(p_i)$

(2)p_i' 是一个只读操作

在影子事务从相应的移动客户机或固定主机上预取到所有的数据之后，事务 T 开始执行，此时 T 所需的所有数据都已经能快速访问到了。

结论 移动环境下的实时事务处理近些年来受到了越来越多的关注，由于无线网络固有的特点，在 MRTDBS 中一个事务很难既满足绝对时间正确性要求又在截止期前正确提交。在本文中，我们考虑到无线网络的特性如经常断接及低带宽等因素，重新定义了相对时间正确性标准，并采用多版本实时数据模型和基于影子事务的执行模型来实现相对时间正确性。

在我们的事务执行模型中，事务从移动客户机生成后，被传输到它对应的基站上，然后被预分析并生成一个影子事务，影子事务又包含了多个子影子事务，每个子事务包含一个或多个读操作集，将事务 T 所需的数据预取到基站上。通过数据预取，数据访问的延迟被大大地降低，从而更好地满足了事务的截止期要求。

参考文献

1 Kayan E, Ulusoy O. An Evaluation of Real-Time Transactiong Management Issues in Mobile Database Systems. The Computer Journal, 1999, 42(6): 501～510

2 Ulusoy O. Real-Time Data Management for Mobile Computing. In: Proc. of Intl. Workshop on Issues and Applications of Database Technology (IADT'98), Berlin, Germany, 1998. 233～240

3 Abbott R J, Garcia-Molina H. Scheduling Real-Time Transactions: A Performance Evaluation. ACM Transactions on Database Systems, 1992, 17(3): 513～560

4 Lee V C S, Lam K, Kao B. Priority Scheduling of Transactions in Distributed Real-time Databases. Journal of Real-time Systems, 1998, 15(1): 31～36

(上接第3页)

3 唐稚松，等. 时序逻辑程序设计与软件工程(上册). 北京：科学出版社，1999

4 Sprivery J. The Notation: A Reference Manual. Englewood Cliffs: Prentice Hall, 1989

5 Allen R, Garlan D. Formalizing architectural connection. In: Proc. of the 16th Intl. Conf. on Software Engineering. May 1994. 71～80

6 Magee J, Kramer J. Dynamic structure in software architectures. ACM SIGSOFT Software Engineering Notes, 1996, 21(6): 3～14

7 Abowd G, Allen R, Garlan D. Using style to understand descriptions of software architecture. ACM SIGSOFT Software Engineering Notes, 1993, 18(5): 9～20

8 周莹新，艾波. 软件体系结构建模研究. 软件学报，1998，9(11)：866～872

9 焦文品，史忠植. 用 XYZ/E 形式化体系结构风格. 软件学报，2000，11(3)：410～415

10 骆华俊，唐稚松，郑建丹. 可视化体系结构描述语言 XYZ/ADL. 软件学报，2000，11(8)：1024～1029