

# 基于面向对象 Petri 网的软件体系结构描述语言

于振华, 蔡远利

(西安交通大学电子与信息工程学院, 710049, 西安)

**摘要:** 以面向对象 Petri 网为形式化理论基础, 提出了一种基于面向对象 Petri 网的体系结构描述语言(OPNADL). 与传统的体系结构描述语言相比, OPNADL 能描述系统的静态和动态语义, 可利用 Petri 网的数学分析方法对系统进行动态分析, 能形象、直观地刻画系统的整体和个体特性, 同时可以利用相应的 Petri 网支持工具对软件体系结构进行模拟、分析和验证. 通过 OPNADL 在公安地理信息系统和柔性制造教学系统开发中的实践证明, OPNADL 可以有效地辅助软件设计人员在体系结构层次上对系统进行分析和设计, 从而提高了系统的开发效率, 具有良好的应用前景.

**关键词:** 软件体系结构; 体系结构描述语言; 面向对象 Petri 网

**中图分类号:** TP311 **文献标识码:** A **文章编号:** 0253 - 987X(2004)12 - 1236 - 04

## Software Architecture Description Language Based on Object2Oriented Petri Nets

Yu Zhenhua, Cai Yuanli

(School of Electronics and Information Engineering, Xi an Jiaotong University, Xi an 710049, China)

**Abstract:** A novel architecture description language (OPNADL) that is based on object2oriented Petri nets was proposed. Comparing with the current architecture description languages, OPNADL can describe the static and dynamic semantics, analyze the dynamic behaviors of the software system by making use of the well2established analysis methods for Petri nets, and visually and intuitively depict the overall and individual characteristics of the system. Moreover, the software architecture based on OPNADL can be simulated, analyzed, verified and vali2dated by the supporting tools of Petri nets. Through applying OPNADL to police geographical information sys2tem and flexible manufacturing teaching system, it is demonstrated that OPNADL can help architecture design2ers to effectively analyze and design the complicated, distributed and concurrent software systems, and has bright application perspective.

**Key words:** software architecture; architecture description language; object2oriented Petri nets

软件体系结构作为一门新兴的计算机学科, 描述了软件系统中组件的组织结构、它们之间的关联关系以及支配系统设计和演变的原则和方针. 研究软件体系结构的核心问题是体系结构描述语言 (Architecture Description Language, ADL), ADL 描述的是软件系统的高层设计, 而不是系统的实现细节或源代码模块. 目前已有几种典型的 ADL<sup>[1-8]</sup> 用来描述软件体系结构, 但它们大部分是领域相关的, 局限于对特定软件体系结构风格的描述与分析, 在

动态体系结构、体系结构的分析和验证等方面存在着不足, 不能同时描述软件体系结构的静态和动态性质<sup>[9]</sup>. 本文以面向对象 Petri 网 (Object2oriented Petri Nets, OPN) 为形式化理论基础, 提出了一种新的体系结构描述语言——OPNADL, 可以对软件体系结构的静态和动态语义进行描述和分析, 突出反映系统的动态结构和组件的交互特性, 并可利用 Petri 网已有的数学分析方法和相应工具对软件体系结构进行模拟、分析和验证.

收稿日期: 2004 - 03 - 15. 作者简介: 于振华(1977~), 男, 博士生; 蔡远利(联系人), 男, 教授. 基金项目: 国家高技术研究发展计划资助项目(2003AA721070).

# 1 ADL 的形式化理论基础

ADL 以 Petri 网、CSP 和 Z 语言等形式化语言为理论基础,对软件体系结构进行描述、分析和验证. 现有的 ADL 一般都采用 CSP、Z 等形式化工具,但这些形式化工具难以同时描述和分析软件体系结构的静态与动态语义,如 CSP 较适合于动态行为的描述,而 Z 只适合描述静态性质. Petri 网作为一种具有严格数学语义的形式化图形建模工具,适合描述分布式、并行系统,能较好地描述系统的静态性质. 它具有的多种数学分析方法又能较好地刻画系统的动态语义,检验系统的死锁性和活性. 本文在一般 Petri 网的基础上建立了一种面向对象 Petri 网,并以此作为 ADL 的形式化理论基础,提出了一种新的 ADL——OPNADL.

## 1.1 面向对象 Petri 网

一般 Petri 网建立的模型非常复杂,模型对系统具有高度依赖性,容易导致模型状态爆炸. OPN 结合面向对象和 Petri 网的优点,提高了模型的模块性和柔性,弥补了一般 Petri 网建模的复杂性等缺点,可简洁地表示复杂系统中的各种资源. 在 OPN 模型中,系统是由相互通信的对象和它们之间的关联关系组成的.

**定义 1** 系统  $S$  是一个二元组,  $S = (O, R)$ , 其中  $O = \{O_1, O_2, \dots, O_n\}$  为系统物理对象的集合,  $O_n$  为系统中的物理对象  $n$  的 OPN 模型,  $R$  为  $O_n$  之间和  $O_n$  与外界交互的消息传递关系.

**定义 2**  $O_n$  是一个九元组,  $O_n = (P, T, P_i, P_o, F, A_i, A_o, E, C)$ . 其中:  $P = \{p_1, p_2, \dots, p_j\}$  为系统一个物理对象库所的有限集合;  $T = \{t_1, t_2, \dots, t_k\}$  为系统一个物理对象变迁的有限集合;  $P_i = \{p_{i,1}, p_{i,2}, \dots, p_{i,l}\}$  为一个物理对象的输入消息库所集合;  $P_o = \{p_{o,1}, p_{o,2}, \dots, p_{o,m}\}$  为一个物理对象的输出消息库所集合;  $F = A(P \times T) \cup (T \times P) \cup (P_i \times T) \cup (T \times P_i) \cup (P_o \times T) \cup (T \times P_o)$  为库所和变迁之间的输入输出弧的有限集合;  $A_i = \{a_{i,1}, a_{i,2}, \dots, a_{i,i}\}$  为外界到物理对象的输入弧,表示对象的输入接口<sup>[10]</sup>;  $A_o = \{a_{o,1}, a_{o,2}, \dots, a_{o,o}\}$  为物理对象到外界的输出弧,表示对象的输出接口<sup>[10]</sup>;  $E: F \rightarrow (I_s, T_d)$  为定义在弧上的表达式函数,其中  $I_s$  为弧的标识符,  $T_d$  表示复杂的数据结构或对象;  $C(P) = \{cp_1, cp_2, \dots, cp_j\}$  为与库所  $P$  相关联的颜色集;  $C(P_i)$  和  $C(P_o)$  为与输入输出库所相关联的颜色集.

**定义 3**  $R$  表示消息传递关系,是一个二元组,  $R = (P_1, C)$ . 其中:  $P_1$  为系统中的智能连接库所,用椭圆表示,系统从外界所获取的消息都存储在  $P_1$  中,系统中的各个对象通过  $P_1$  进行消息派遣;  $C(P_1)$  为与库所  $P_1$  相关联的颜色集.

**定义 4**  $P_1$  中的 Token 数据类型是一个二元组  $(I_s, T_d)$ . 其中:  $I_s$  为各个对象的输入、输出接口的标识符;  $T_d$  是复杂的数据类型或对象.

采用 OPN 对复杂系统进行建模,可以单独描述系统中的每一个对象. 当环境或某个对象的属性或行为发生变化时,只需更改某个或某几个对象,而不需修改整个系统的模型,这样就降低了建模的复杂度,提高了模型的适应度. 当 OPN 模型建立后,可以通过变迁的使能和发射显示标识的变化情况,利用 Petri 网分析工具对模型进行模拟仿真,对系统进行初步的验证.

在 OPN 中,变迁  $t \in T$  的前置集定义为  $i = \{p \in P | (p, t) \in F\}$ , 其后置集定义为  $t' = \{p \in P | (t, p) \in F\}$ ,  $t$  的前置集也称为  $t$  的输入库所,其后置集称为  $t$  的输出库所.

**定义 5** OPN 的使能规则:对于变迁  $t$ ,假定系统的初始标识为  $M_1$ ,若  $\forall p \in P, p \in i$ ,在标识  $M_1$  下,  $p$  中满足弧表达式要求的 Token 数大于弧  $(p, t)$  的权值并且满足弧表达式函数,则变迁  $t$  是使能的.

**定义 6** OPN 的发射规则:变迁发射,意味着事件的发生. 从变迁的各个输入库所中移走使能该变迁的 Token,而向它的各个输出库所加入 Token. 如果在标识  $M_1$  下变迁  $t$  发射,则生成标识  $M_2, M_2(p)$  的计算规则是

$$M_2(p) = \begin{cases} M_1(p) - W(p, t) & p \in i \setminus t' \\ M_1(p) + W(p, t) & p \in t' \setminus i \\ M_1(p) - W(p, t) + W(t, p) & p \in i \cap t' \\ M_1(p) & p \in i \cap t' \end{cases}$$

## 1.2 OPN 中面向对象的特性

OPN 是以面向对象为基础进行定义的,它能较好地体现面向对象的封装、继承、多态等特性.

**1.2.1 封装** OPN 模型将数据和基于数据的操作封装在一起,完全隐藏了对象的内部实现细节,对象和外界仅仅通过定义好的输入、输出接口进行交互. 封装使得 OPN 模型对内是一个结构完整的整体,对外则是一个功能明确、接口单一、在各种适合的环境下都能独立工作的有机单元.

1.2.2 继承 继承是新的对象使用现有对象功能的手段,也是实现软件重用的一种途径.在 OPN 中,通过聚合提供继承这种特性.如图 1 所示,对象 B 继承对象 A 的输入、输出接口(接口 2 和接口 1).当采用聚合模式时,对象 B 仍然创建一个对象 A,但对象 B 自己并不实现接口 1 和接口 2.当应用程序调用接口 1 和 2 时,对象 B 将对象 A 的接口指针直接返回给应用程序,为此在通常的实现中,对象 B 和对象 A 应相互保存对方的接口指针.

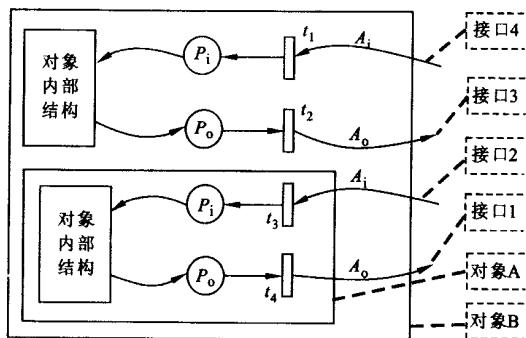


图 1 OPN 中对象的继承机制

1.2.3 多态 多态是根据所使用的对象展现多种不同行为的能力.在 OPN 中,实现相同接口的 2 个对象就被认为是多态的,即同一个接口可以由多个对象以不同的方法实现.

## 2 基于面向对象 Petri 网的体系结构描述语言

ADL 定义和确定了适合于软件体系结构表达与描述的有关元素,揭示了软件系统高抽象层次上的特性,主要包括对组件、连接件、角色、约束等 4 个元素的描述.OPNADL 采用 OPN 为形式化理论基础,对软件体系结构的 4 个抽象元素进行建模.

定义 7 OPNADL 是一个四元组,其形式化定义为  $L = (C_m, C_c, R_r, C_a)$ .其中:  $C_m$  为软件体系结构中的组件;  $C_c$  为软件体系结构中的连接件;  $R_r$  为软件体系结构中的角色;  $C_a$  为软件体系结构中的约束.

下面对上述定义中的各元进行详细论述.

组件是一个数据单元或一个计算单元,是具有扩展性和集成性的计算或状态存储的场所,它由组件接口和组件实现组成.组件是一个二元组,  $C_m = (D_s, O_n)$ ,其中  $D_s$  是 OPN 中输入、输出接口弧的标识符集合,  $O_n$  为系统中的组件  $n$  的 OPN 模型,定义了组件的接口和内部实现.

$O_n$  中的元  $A_i$  和  $A_o$  描述了组件的输入和输出

接口.接口描述组件需求的服务和提供给外界的服务,特别是描述组件能够接收和发送的消息,组件的实现部分由 OPN 的其他元素进行描述.各组件在逻辑上是互相独立的,它的正确性不受其他组件的影响,组件之间的交互由连接件进行描述,这样就可以提高组件的独立性,有利于重用和维护.

组件是可重用的软件单元,可以分为复合组件和原子组件.原子组件是不可再分的实体,一般对应于一个过程、进程或数据存储单元.复合组件由若干原子组件组成,或由其他复合组件和原子组件经过信息交互连接而成.

连接件是组件交互协议的实现,定义了组件之间交互的规则并且给出了一些实现的机制.连接件是一个一元组,  $C_c = (R)$ ,其外部界面被称作协议.从通信的角度来看,连接件控制和管理着组件之间的通信和协作活动.从系统的连接与粘合角度来看,连接件则充当着粘合软件系统的胶水作用.

连接件是组件间进行连接通信的设计框架,在本文所提出的连接件中,其实现机制采用软件总线(如 CORBA/DCOM/EJB)的方式,软件总线通过消息或报文进行消息传递,消息可以是同步消息或异步消息,组件之间通过软件总线进行通信.软件总线负责消息或报文的分派、传递和过滤,各个组件挂接在软件总线上,向总线登记感兴趣的消息类型.组件根据需要发出消息,由软件总线负责将该消息分派到系统中所有对此消息感兴趣的组件,组件接收到消息后,根据自身状态对消息进行响应,并通过总线返回处理结果.由于组件通过总线进行连接,并不要求各个组件具有相同的地址空间或局限在一台机器上,可以分布在局域网、广域网甚至 Internet 上的不同计算机中.这种实现机制可以较好地刻画分布式并发系统.

角色为与连接件相交互的、在连接件中处于相同地位的所有组件的抽象集合,  $R_r = \{C_{m,1}, \dots, C_{m,i}\}$ ,  $C_{m,i}$  为系统中组件的标志符.

角色有静态和动态 2 种类型.静态角色在软件体系结构建立时就已确定,但随着软件复杂性的提高,系统中的组件可能会随着环境变化动态地增加或删除,角色因而也就发生动态的变化.

约束  $C_a$  为五元组,  $C_a = (T_t, N_n, J_r, N_i, S_a)$ ,其中:  $T_t$  为组件,只能通过接口和连接件相连,而不能与其他组件直接相连;  $N_n$  为组件的接口数,必须大于 0;  $J_r$  为连接件必须有角色;  $N_i$  组件在执行时可以有多个实例;  $S_a$  为原子组件,不能包含其

他组件.

采用 OPNADL 描述软件体系结构如图 2 所示,系统由 4 个组件构成,每个组件都通过接口挂靠的连接件上,形成了一种树状的拓扑结构.采用软件总线,更容易实现组件的动态添加和删除,可以使用户在事先不知道组件接口信息的情况下通过查询接口库或采取其他手段动态地获得对象接口信息,然后使用动态调用方法构造客户请求并发送到对象实现.图 2 描述了软件体系结构的静态语义,非常形象地刻画了整个软件体系结构的组成及组件之间的交互关系.

OPNADL 对软件系统动态语义的描述如下所述.系统中的元  $R$  描述了连接件,负责整个软件系统的消息传递.当  $P_1$  库所中的 Token 数大于 1 时,根据 Token 的  $I_s$ ,把 Token 发射到相对应对象 OPN 中的输入库所  $P_i$  中.在  $P_i$  中,所有与某个 OPN 输入接口有关联的 Token 都形成一个消息队列.如果  $P_i$  中有 2 个 Token,分别对应某个 OPN 的 2 个接口,那么这 2 个接口和  $P_i$  之间的变迁可同时发射,并发执行.

OPNADL 形象地描述了软件体系结构的动态语义.通过变迁的发射,Token 从一个库所分配到另外一个库所,表明了资源或消息的传递,较好地说明了整个软件系统的流程.OPNADL 还可以用形式化的分析方法对软件系统的死锁和活性进行动态分析

和验证,进行早期的预防和检测,避免建模时的人为错误,同时可以利用相应的 Petri 网支持工具,对软件体系结构模型进行模拟和仿真.

在利用 OPNADL 对软件体系结构进行建模的过程中,应忽略对象之间的复杂关系和约束,只考虑对象之间最基本的消息传递,确定系统中的各个对象,抽象成软件组件模型,建立和实现系统的软件体系结构模型.在以后的软件开发中,利用已创建的可重用的软件组件,对其直接进行复用,组装成一个实际的软件系统.

### 3 结 论

为了检验 OPNADL 的实际效果和应用前景,我们在“某市公安地理信息系统”、“柔性制造教学系统”控制软件的开发和设计中均采用了 OPNADL 来描述系统的软件体系结构.这 2 个系统都是多层次、分布式软件系统,功能需求复杂.实际的使用情况和效果证明,OPNADL 确实能够有效地辅助软件系统的设计人员在软件体系结构层次上对软件系统进行有效的分析和设计,并且基于 OPNADL 的软件体系结构能很好地适应系统的动态变化.OPNADL 作为一种可视化的体系结构描述语言,能非常形象、直观地刻画系统的整体和个体的结构和行为,能促进客户、体系结构设计者和开发人员之间

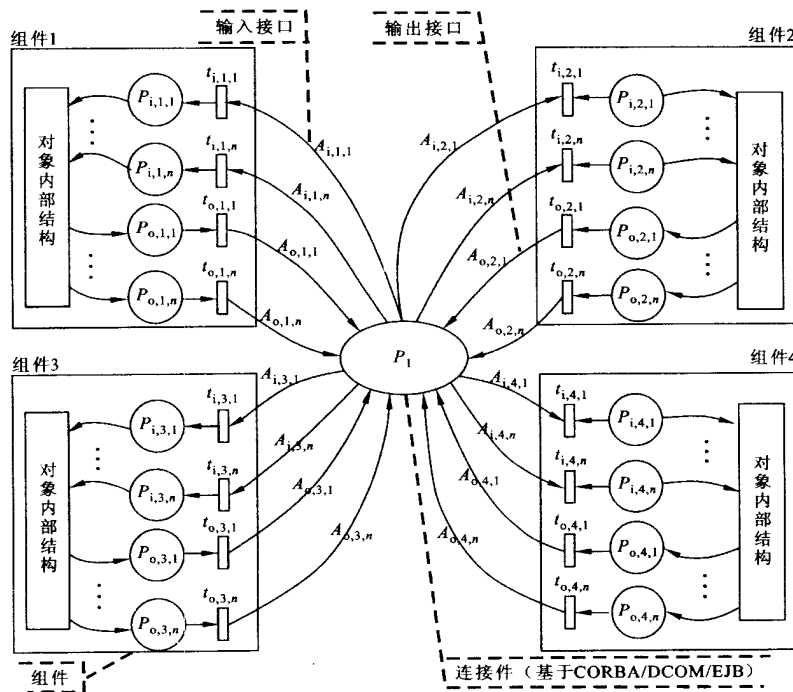


图 2 基于 OPNADL 的软件体系结构

(下转第 1275 页)

表 2 声表面波式小波变换阵列器件的特性

| $j$ | $s$      | $f_j/\text{MHz}$ | $f_j/\text{MHz}$ | 频带/ $\text{MHz}$     |
|-----|----------|------------------|------------------|----------------------|
| - 5 | $2^{-5}$ | 4. 240           | 12. 720          | [8. 480, 16. 960]    |
| - 6 | $2^{-6}$ | 8. 480           | 25. 440          | [16. 960, 33. 921]   |
| - 7 | $2^{-7}$ | 16. 960          | 50. 880          | [33. 921, 67. 842]   |
| - 8 | $2^{-8}$ | 33. 920          | 101. 760         | [67. 842, 135. 684]  |
| - 9 | $2^{-9}$ | 67. 840          | 203. 520         | [135. 684, 271. 368] |

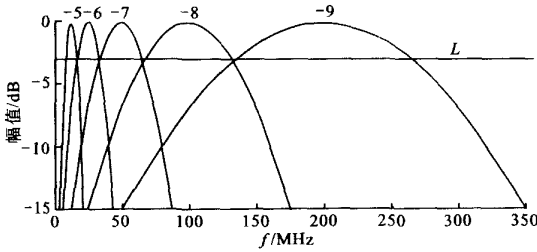


图 2 声表面波式小波变换阵列器件的频率响应

## 4 结 论

本文对声表面波式小波变换阵列器件的基波中心频率和基波频带半径之间关系进行了研究,最后

得出在基波器件中心频率等于 3 倍基波频带半径时,按照二进规则实现的阵列器件在 - 3 dB 处频带是完全连续的,这样用声表面波阵列实现了二进小波变换,不仅解决了频带重叠和不连续问题,而且对于解决小波变换和硬件实现中冗余性及粗糙问题也有重要的参考价值.

### 参考文献:

- [1] 朱长纯,卢文科. 声表面波式小波变换器件及其应用 [M]. 北京:国防工业出版社,2004. 26 - 36, 72 - 73.
- [2] 卢文科,朱长纯,刘君华,等. 声表面波式小波变换及重构器件的实现的实现的研究 [J]. 电子学报,2002, 30 (8) : 1 156 - 1 159.
- [3] 卢文科,朱长纯,刘君华等. 用声表面波器件实现小波变换的研究 [J]. 中国科学: E 辑,2003, 33(11) : 1 028 - 1 036.
- [4] 魏培永,朱长纯,刘君华. 用声表面波器件实现小波变换的方法 [J]. 西安交通大学学报,2001, 35 (4) : 394 - 397.
- [5] 冉启文,谭立英. 小波分析与分数傅里叶变换及应用 [M]. 北京:国防工业出版社,2002. 53 - 54.

(编辑 刘 杨)

(上接第 1239 页)

的交流和理解,非常适合描述复杂的大型软件系统.

目前,利用基于 OPNADL 的软件体系结构指导软件开发还处于起步阶段,许多模型和方法正在探索中,进一步的工作将主要研究基于 OPNADL 的软件自动生成.

### 参考文献:

- [1] Shaw M, DeLine R, Klein D V, et al. Abstractions for software architecture and tools to support them [J]. IEEE Trans on Software Engineering, 1995, 21 (4) : 314 - 335.
- [2] Allen R, Carlen D. A formal basis for architectural connection [J]. ACM Trans on Software Engineering and Methodology, 1997, 6(3) : 213 - 249.
- [3] Moriconi M, Qian X, Riemenschneider R A. Correct architecture refinement [J]. IEEE Trans on Software Engineering, 1995, 21(4) : 356 - 372.
- [4] Luckham D C, Kenney J J, Augustin L M, et al. Specification and analysis of system architecture using Rapide [J]. IEEE Trans on Software Engineering, 1995, 21

(4) : 336 - 355.

- [5] Taylor R N, Medvidovic N, Anderson K M, et al. A component2and2message2based architectural style for GUI software [J]. IEEE Trans on Software Engineering, 1996, 22(6) : 390 - 406.
- [6] Medvidovic N, Taylor R N. A classification and comparison framework for software architecture description languages [J]. IEEE Trans on Software Engineering, 2000, 26(1) : 70 - 93.
- [7] 骆华俊,唐稚松,郑建丹. 可视化体系结构描述语言 XYZ/ADL [J]. 软件学报,2000, 11 (8) : 1 024 - 1 029.
- [8] 张家晨,冯 铁,陈 伟,等. 基于主动连接件的软件体系结构及其描述方法 [J]. 软件学报,2000, 11 (8) : 1 047 - 1 052.
- [9] 周莹新,艾 波. 软件体系结构建模研究 [J]. 软件学报,1998, 9(11) : 866 - 872.
- [10] Saldhana J A, Shatz S M, Hu Z. Formalization of object behavior and interactions from UML models [J]. International Journal of Software Engineering and Knowledge Engineering, 2001, 11(6) : 643 - 673.

(编辑 刘 杨)