一种基于 Petri 网的软件定义构造方法*

何志均 应 晶 吴朝晖 高 济

(浙江大学人工智能研究所 杭州 310027)

摘要 本文描述一种支持 MHSC 方法论的软件需求定义构造的 Petri 网方法. 基于提供的需求描述模型,可形成 Petri 网表达的软件定义结构,通过变换与求精方法,逐步生成实现层的功能模型. 利用支持不同粒度功能模拟的仿真机制对定义进行证实与验证. 论文提出一种新的定义框架,并为软件自动生成的研究从定义层的描述机制出发提供一种新思路.

关键词 Petri 网,软件定义,高层构造.

1 软件定义的方法论需求

软件工程的一个重要挑战是软件生产过程需要有一个根本性的改进. 其原先的重点主要放在生产过程的控制上,而不是放在改变生产过程从而达到生产的根本性改进上. 对生产过程的控制,一般说来不能给软件生产过程带来实质性的改进,实现根本性的改进需要采取构造性方法.

软件实践表明,软件系统的开发存在着其定义层(指软件开发前端的需求描述、分析及软件定义的形成)与实现层(指软件系统的算法、代码、数据结构的实现与产品形成阶段)之间的断层. 表现在:生命周期开发方法的重点在软件的设计实现层,给系统的功能更新与维护以及重设计工作带来较大的困难;用户的需求经需求分析阶段细化成若干缺乏相关依赖性和一致语义的信息,从实质上已脱离了与用户的反馈能力,定义不具有可操作性而由人工解释后直接进入面向程序的实现工作;软件定义的描述缺乏灵活性和可修改性,阻碍了软件系统的定义证实和功能验证. R. T. Yeh 在文献[4]中指出,软件开发中存在双重压迫:一是大型系统的问题定义方法,二是复杂系统的维护问题. 由此迫切需要引入一种新颖的方法论来指导软件系统开发,使软件生产率有较大的改进,打破开发瓶颈.

综合上述讨论的 2 个方面,软件的开发问题实际上是如何描述功能需求的问题. 引入一种形式化的定义方法,通过构造方法抽取软件系统的功能模型. 软件的整个开发演变为基于定义的增量开发过程,即作为以后开发阶段的唯一基础,通过在此高层定义上的连续加工,

* 本文研究得到国家 863 高科技项目基金资助. 作者何志均,1923 年生,教授,博士导师,主要研究领域为人工智能,智能软件开发环境,CAD. 应晶,1971 年生,博士,主要研究领域为人工智能,基于知识的软件工程,CASE. 吴朝晖,1966 年生,博士,主要研究领域为人工智能,软件工程. 高济,1946 年生,教授,主要研究领域为人工智能,智能软件开发环境,知识建模.

本文通讯联系人,何志均,杭州 100080,浙江大学人工智能研究所

本文 1996－01－08 收到修改稿

直至底层产品的生成.这种作法弥补了软件开发中存在的断层问题,并给产品的维护、再实现等提供了可能.因此,围绕软件系统开发过程出现的断层,无论是从软件的构造性方法与重设计工程、或是定义层与实现层之间的断层问题入手,均存在着软件定义的方法论需求.

2 基于高层定义构造的方法论

作者在文献[8]中提出一种支持软件定义高层构造的方法论 MHSC(methodology for high-level software specification construction).MHSC 的思想是从软件开发的需求定义层入手,研究软件定义的高层构造过程.提出一种能支持多语义维的定义语言,并构造实际领域软件系统的合一化功能模型.在形成的合一化模型基础上,作用基于知识的变换、求精方法,逐步过渡到实现层的软件设计,以支持软件的自动开发过程.MHSC 支持具有增生结构的软件构造过程.

MHSC 强调一种多语义维的定义,统一地描述数据流、控制流、状态转换、实体关系及模块组织等语义信息(在以往的方法论中是分离的,而且每一维均有单独的处理工具,难以耦合),形成软件系统的一个可演化的单一模型,作为整个软件加工过程的核心.软件开发过程可以简化为一个三元组:$SD=(I_i,O_i,T_i)$,其中 I_i:功能、资源、性能需求;O_i:提供了实现系统所需的定义;T_i:变换、求精、概念具体化及合成方法.

软件定义是一个概念模型,同时又是一个行为模型,通过描述性语言可以支持定义的可执行性与表达能力 2 个方面,而且这使得抽象层上的程序验证变得可能;同时可执行定义提供了一种直接过渡到系统设计的可能.[6]下文中所述的基于 Petri 网的定义描述机制较好地体现了 MHSC 的上述思想.

3 基于 Petri 网的定义构造

自德国人 G. A. Petri 于 1962 年在他的博士论文中提出 Petri 网后,许多研究者致力于这方面的研究.Petri 网作为研究系统构成模型、系统数学描述及了解系统动静态特性的工具,已在计算机系统的各方面得到成功的应用.对它的描述有图形表示和形式化描述 2 种方法,利用带标志的 Petri 网则全面地反映系统的动态和静态特性.

Petri 网可由一个四元组描述:$PN=(P,T,I,O)$

$I:T\to P^\infty$

$O:T\to P^\infty$ 且 $P\cap T=\Phi$

其中 I 是与 T 关联的输入函数;O 是与 T 关联的输出函数.P 代表 Petri 网中的地点(或位置,PLACE);T 代表 Petri 网中的转移(或转换,TRANSITION).

把 Petri 网作为一种设计方法的思想见于文献[3,4]中提出的一种带仿真器的基于图形的编辑器 SPECS,它支持结构化网络的构造和仿真.本文作者提出的一种扩展是支持:① Petri 网描述的合一化模型;② 不同粒度功能模拟的仿真机制;③ 可视支持的变换求精;④ 针对快速原型和功能需求的早期检查.Petri 网与图形定义方法中的 DFD,SADT 相比,后者主要描述系统静态特性,而前者描述动态行为.形式定义的 Petri 网具有较丰富的语义并且可以通过合适的工具进行模拟.Petri 网理论的结果主要用于分析和验证.作者提出一种

基于Petri网的知识库一致性检查系统[10],并在Petri网基础上提出一种支持产生式系统并行推理的动态网.[12]综上所述,考虑通过Petri网来支持MHSC方法论所体现的软件定义高层构造.具体体现在:

Petri网支持了多语义维的描述方法,包括其描述机制中反映的状态转移图形式(PLACE对应于状态),利用Petri网的点火激活方式完成状态之间的动态变迁;同时把事件驱动机制作为标志生成器,较好地体现了原型系统中的控制流语义;转移中包含的丰富语义较好地支持了不同粒度功能、模块的表达,以及多种网络触发机制(定时器、触发器、手工驱动、事件驱动);

Petri网是一种图形表示法,通过图示构造为用户提供了整个网络图的浏览、多视图功能,采用各类可视支持设施(可视编辑器、图文转化器、可视求精算子、可视变换算子、可视模拟机制),用户不仅对定义有直观的理解、更新和维护,还能积极参与开发过程,形成开发人员与用户需求之间充分的反馈能力;

Petri网是一种活性网,本身包括了特有的描述功能.利用点火激发机制,根据此描述的软件定义置入实际的运行环境中,辅以事件生成与资源管理,展现在用户面前的是一个系统功能仿真环境,不仅清晰地反映整个系统的动静态功能,包括实际运行环境,对系统的整体外部行为有精确的把握,还能在此基础上发现问题,为定义的证实及系统功能的验证、维护提供了基础.

考虑用Petri网的变换方法实现:扩展Petri网,利用网络来建造定义的可执行模型,并实现模型结构的自动变换,同时完成仿真与自动的代码生成.实际上在定义之上作用变换,实质是内部结构的变化,而外部行为则保留原来的特性(保意变换).实现步骤包括:建造由定义语言描述的需求;形成Petri网;在网上作用变换,使此网络结构能满足实现环境的需求;实现(任何时候Petri网作为原型执行,并支持仿真或解释).作者对定义设计的出发点是:忽略需求定义和设计定义的界限.经典描述方法中的DFD等方法不能描述大型系统中的黑盒子(主要是大粒度模块),所以必须涉及那些高层处理的底层细节才能反映系统的整体行为.我们的实现方法主要通过信息局部化、分离细节、表达可重用概念、定义动作的粒度、定义时序约束,并把语义的形成建立在概念模型之上.要求能支持领域实体的描述;各类变化(过程、处理、事件);约束、假设;时间特性;知识的组织;抽象机制(求精);一致性检查.

在定义模型的构造中,状态转移是定义构造的核心成分,定义为一个九元组:

$$Transition = (N, PreCond, PostCond, TC, GC, GEL, LEL, IF, A)$$

其中N:转移名称;$PreCond$:前条件(执行前须满足的条件);$PostCond$:后条件(执行后须满足的条件);TC:时序约束;GC:通用约束;GEL:感知的全局事件链(不同机构能访问的事件);LEL:感知的局部事件链(仅激活本机构中的状态改变);IF:状态转移过程的信息流;A:执行动作.

Petri网的核心结构可描述为:

```
-State: /* 即扩展的 PLACE */
char        * index;              /* 状态索引名 */
char        * type;               /* 状态类型(服务,机构类属) */
Transition  * trans[MaxTRAN];     /* 与状态相关联的转移 */
```

```
char        * remark;          /* 注释内容 */
int         cycle;             /* 激活周期 */
int         tag[];             /* 状态标志 */
—Transition: /* 扩展的 TRANSITION */
char        * index;           /* 转移索引名 */
char        * type;            /* 转移类型 */
State       * Prestate;        /* 触发状态 */
State       * Poststate;       /* 转移状态 */
Condition   * Precond;         /* 激活前需满足的前条件 */
Condition   * Postcond;        /* 激活后需满足的后条件 */
Constraint  * Timeconst;       /* 时序约束 */
Constraint  * Genconst;        /* 通用约束 */
InfoFlow    * infoflow;        /* 信息流描述 */
Event       * event;           /* 感知的全局/局部事件 */
Action      * action;          /* 激发动作 */
char        * remark;          /* 注释内容 */
int         cycle;             /* 激活周期 */
int         tag[];             /* 状态标志 */
```

4 定义变换、求精与仿真支持

变换方法是一个较为抽象的定义被重复地变换(细化),通过更加具体的形式,直至一个目标系统的生成. C. Rich[5]曾预测:具有某种形式的变换法肯定是所有未来自动程序设计系统的组成部分. 本文以 Petri 网定义模型为基础,充分发挥变换过程的特色来体现软件开发过程的自动化程度.

MHSC 方法论能够较好地支持演化式的变换方法. 变换并不是提供定义的实现,而是重新对定义进行求精. 软件整个开发过程可视为不同定义层次上的变换序列,每个变换均是在同一种模型上作一些改变,或用较底层的构造来替换高层构造,或是在同一层次上进行重组. 从实质上体现了使用变换来支持基于定义的原型开发的思想.

在基于变换的实现这一点上作者提出 3 个原则:① 定义描述模型必须是一种宽域语言,即能描述不同程度、不同层次的功能设计信息(Petri 网很好地支持了这一点);② 为了获取转换规则,需要获取程序设计知识. 基于此需要研究目标语言的语言特性;③ 支持手工的求精机制. 在变换的作用模型上,作者结合 AI 技术来支持变换目标的形式化、变换的作用策略、变换选择的合理性及手工(专家)变换几个方面.[9,11,13]利用问题求解技术来自动地产生各个变换序列,并支持变换的选择和作用的人工干预. 从广义上讲,整个变换作用过程可视为一个软件产品.

求精方法是对形成的定义模型的扩展. 变换是领域知识的综合体现,具有领域的适应性,而且包含了部分程序设计知识;求精主要是为用户的交互操作提供基本设施. 因此把基于知识的支持体现在变换过程中,而把基于图形的支持体现在求精过程中. 可以提供图形设

施支持定义的可视表达与各类可视操作，使得用户能在图示界面对定义完成初始构造，并通过交互手段对定义进行求精，辅助整个变换过程.

基于 Petri 描述的定义支持如下仿真机制：通过面向实际领域的仿真与模拟功能，真正引入资源的操作与分配，不仅体现系统所具有的各项功能，还能从整个实际运行环境中观察软件系统的整体行为；通过软件定义和用户界面的分离，把可执行软件原型置入另行构造的环境界面中；支持多粒度的功能证实；合一化模型体现了可视化、图文并存、强交互机制、多重视图，从而减轻用户的理解难度.

综上，整个 MHSC 方法论对软件开发过程的支持开发流程为：

① 分析某一类领域模型，并扩充或修改已有的重用库，获取及更新各类变换知识及其作用策略，对各类机构、状态、事件、信息结构、服务、模块等加以访问并进行重新组织；

② 利用 Petri 网的定义描述模型构造软件的合一化模型，从重用库中匹配或抽取各种子模型及构造成分；

③ 由变换机制完成合一化模型的演化式变换过程，从模型变换到底层代码变换. 同时对变换过程加以记录和跟踪，并提供变换过程的重现机制；

④ 支持可视集成界面的功能. 在合一化模型的可视布局中应用各类求精算子、可视操作加以手工(或半自动)变换，直接反映在基于 Petri 网的定义描述上；

⑤ 体现 Petri 网的可执行能力. 通过可视化仿真环境对定义模型进行解释执行，从总体上把握所构系统的功能；

⑥ 对重用库的扩展，为同一领域的软件构造提供重用基础.

5　讨论与结论

本文指出软件开发中存在的断层及其影响. 针对这种现状与需求，MHSC 方法论提出一种基于 Petri 网的定义描述方法，并提供 Petri 网模型的图形仿真、动态行为跟踪、更新手段、定义的交互编辑能力、通过窗口系统控制模型仿真的激发与否、点火机制、用户视图等功能，并体现了 KBSE 所支持的“定义层－证实－可执行定义(快速原型)－检验－具体实现(实现层)－需求的反馈修改－定义层－重用－重实现过程”这一流程. 因此，能够较为精确地把握定义层上对用户所需系统的多维描述，对整个开发过程起着不可估量的作用.

如何发挥定义的可执行能力是基于 Petri 网的描述方法的关键环节. 利用 Petri 网来构造定义，能形成一个综合描述多种语义维的定义，形成软件系统的一个可演化变换的单一模型作为软件加工过程的核心. 可以说基于 Petri 网的定义描述支持了一种软件定义的高层构造方法.

结合上述讨论，进一步应该考虑如何重用以 Petri 网描述的定义. 作者认为能重用软件的唯一现实想法是要求软件设计者一开始就考虑重用性，即把其作为初始设计的一个目标，这亦是软件重设计工程与逆向工程的主要立足点. 对此我们将开展进一步的研究.

参考文献

1　陆维明，林闯. Petri 网研究：机遇与挑战. 计算机科学，1994，21(4)：1～5.

2　袁崇义. Petri 网. 南京：东南大学出版社，1989.

3 Reisig W. Embedded system description using Petri nets. LNCS, Vol. 284.
4 Yeh R T. 新型的软件演进风范. 计算机科学, 1991, 18(6): 18～43.
5 Rich C, Waters R C. The programmer's apprentice: a research overview. IEEE Computer, 1988, 21(11): 10～18.
6 Zave P. Salient features of an executable specification language. IEEE, 1986, SE－12(2): 312～325.
7 Lowry M R. Software engineering in twenty－first. AI, 1992, 13(3): 71～87.
8 Ying Jing, He Zhijun *et al*. A methodology for high－level software specification construction. ACM Software Engineering Notes, April 1995, 28(2): 48～54.
9 Ying Jing, He Zhijun. New inquiry into integrating AI techniques with CASE. Proc. 6th International Symposium on Artificial Intelligence(ISAI'93), Monterry, Mexico, Sep. 1993.
10 应晶, 何志均. 支持产生式系统并行推理的动态推理网. 自动化学报, 1996, 22(2): 40～47.
11 应晶, 何志均. 支持软件需求分析及其开发的智能环境. 浙江大学学报, 1994, 28(1): 9～17.
12 应晶, 吴朝晖, 何志均. 基于 Petri 网的知识库一致性检查系统. 计算机研究与发展, 1992, 29(8): 30～35.
13 应晶等. 基于高层定义构造的 KBSE 方法. 第 3 届中国人工智能联合大会论文集(CJCAI'94), 1994.

A PETRI－NET BASED METHOD FOR SOFTWARE SPECIFICATION CONSTRUCTION

He Zhijun Ying Jing Wu Zhaohui Gao Ji

(Institute of Artificial Intelligence Zhejiang University Hangzhou 310027)

Abstract This paper addresses a Petri－net based approach to supporting specification construction and execution of software system. On the basis of requirement descriptive model, a kind of specification represented in Petri－net is formed, which can generate the functional model on the implementation level by applying transformation and refinement. The specification is validated and verified by different simulation mechanism. This paper has put forward a kind of specification framework and may provide a novel idea for software automatic generation from the specification level.

Key words Petri－net, software specification, high－level construction.