第7卷第3期 软件学报 Vol. 7, No. 3
1996年3月 JOURNAL OF SOFTWARE Mar. 1996

一种严格的软件开发方法框架*

陈火旺 齐治昌 王兵山 宁 洪 谭庆平

(长沙工学院计算机系 长沙 410073)

摘要 本文系统地提出一种严格的软件开发方法,它基于逐步精化和重用组合的程序设计思想,将基于图形的半形式化方法和基于逻辑和转换系统的形式化方法镶嵌为一体,使软件开发中的"创造"和"演算"得到合理的折衷.本文已初步实现了面向该方法的实现工具.

关键词 软件开发,形式化方法,设计演算.

1969年,E. W. Dijkstra首先提出了结构程序设计的概念,它强调了从程序结构和风格上来研究程序设计.它的推行,其意义远不止当前的实用价值,更深远的意义在于它向人们揭示了程序设计方法研究的必要性.70年代程序设计技术的发展是以方法的研究为特征的,除结构程序设计的方法日趋完善外,在数据抽象程序设计的推导和程序综合、变换和自动化等方面,都取得了重要的进展.80年代,人们把注意力集中到程序设计环境的改善方面,从理论高度探讨程序设计的一般规律,同时将方法研究成果实现为有实用意义的工具和环境.随着70年代和80年代关于程序设计理论和方法研究的不断深入,人们总结出一整套关于如何进行程序设计的原则和方法,这里不仅包括结构化程序设计,而且还涉及程序语义性质的表示、程序正确性、程序的形式化开发和程序的效率等多方面.这就使程序设计方法作为计算机科学的一门前沿学科应运而生了.[1,2]

我们认为,原则上说,一个好的软件开发方法从技术上应该:

(1)有效地融合软件开发中的"可自动化性"和"人的创造性".

"可自动化性"即系统中可以用数学化手段或者程序来描述或实现的性质,而"人的创造性"则是仍旧不能被代替的人的作用.每个系统都不可避免地存在这2类因素,只有对它们进行仔细划分,才能有效地建立人机合作机制.人机分工也应有一定规律可循.比如说人应当完成那些经常变化的难以用模型来表述以及需要大量的客观经验经直观推理而完成的工作,而机器则完成任务性质较稳定的计算,操作复杂而规律的、重复性的工作.人的因素是活跃的成分,应当特别加以分析,并采取积极的手段来使人的因素更适合于系统要求.一般认为,人是积极的主体,在思维上能综合声、图、文及一些隐含的信息,运用形象思维和理性思

* 本研究得到国家863高科技项目基金资助.作者陈火旺,1936年生,教授,研究领域为软件工程,人工智能.齐治昌,1942年生,教授,研究领域为软件工程.王兵山,1938年生,教授,研究领域为科学理论及计算机软件.宁洪,女,1961年生,副教授,研究领域为软件工程.谭庆平,1965年生,副教授,研究领域为科学理论及软件工程.

本文通讯联系人:陈火旺,长沙 410073,长沙工学院计算机系

本文1995-08-31收到

维,较快地达到问题的实质,找到解决方法;同时,人倦于重复、单调性工作,缺乏耐心,还常常受脑中固有概念的影响,带有难以克服的偏见.在系统中由于这些原因,常导致故障的产生.因此,我们希望在软件开发环境中,软件开发的"创造"和"演算"得到合理的折衷.

(2)尽量逼真地模拟客观世界及其事物,再现人类认识事物的思维方式和解决问题的工作方式.

对客观世界的抽象认识,包括事物以及事物之间的关系.事物包括其内部特征和外部特征,事物之间的关系指它们的相互作用.倘若从突出事物之间的关系出发,则可引入面向关系的设计方法和语言(如函数型语言、逻辑型语言);倘若从突出事物的观点出发,则可引入面向事物的设计方法和语言(如面向对象的语言).从软件开发过程上,能有效地控制复杂性,处理好功能和性能、控制流和数据流、分解和合成等之间的关系.

本文基于以上2个方法学原则,以我们近年来863软件生产自动化课题方面的工作为线索,给出一种严格的软件开发方法,它基于逐步精化和重用组合的程序设计思想,将基于图形的半形式化方法与基于逻辑和转换系统的形式化方法镶嵌为一体,使软件开发中的"创造"和"演算"得到合理的折衷.

1 基本思想和方法学原则

1.1 软件开发的思想

抽象和分解是在人类认识和解决问题时控制问题复杂性的2个基本思想.它们已经以不同的形式出现在软件开发中.

关于抽象,Wirth曾做过如下说明:"我们对付复杂问题的最重要的方法是抽象.因此,对一个复杂问题不应该立刻用计算机指令、数字和逻辑符号来表示,而应该用较自然的抽象语句来表示,从而得出抽象程序".抽象有2种含义,一是通过对复杂问题的分析,区别本质和非本质的,去掉非本质的成分,明确问题的实质,准确明了描述问题的结构,从而把非结构问题转化为结构问题;其二是在模块化过程中,把下层模块的功能概括到上层模块,形成上层模块的抽象机制.例如,对多视角需求描述的选取,我们遵循这样一个准则:它们必须严格、精确以适于后期计算机的实证、模拟和分析,同时也要求明了、直观,以便于开发人员生成、检查和修改需求.

分解是基于分而治之的策略以克服整体设计的复杂性而形成的.在程序设计语言中所提倡的模块化和信息隐蔽即是它的体现.从软件的开发技术上讲,分解是抽象趋向具体的途径,例如逐步精化思想;与之相对偶,合成是分解后软件实现的重要途径;由此,软件开发过程中形成了自顶向下和自底向上2种手段的并存.

软件开发的中心环节是软件设计,类似于其它科学和工程问题的求解过程.软件设计是设计者设计问题的求解过程,对于科学工程应用软件研制而言,是设计者对解决问题所需要的知识体系的认识过程,该知识体系应该包括理论模型、计算方法、软件开发方法与技术等,而设计方案是认识达到一定深度后形成的.软件设计是设计者进行创造性劳动的过程.事实上,上述抽象和分解可以视为软件开发中创造性的2个方面.

再从人工智能的角度看,软件开发是一种智能密集型活动.软件开发的第1步是认识软件的客观知识体系,需求分析则是认识工作的开始.但是,需求分析完毕并不意味着这种认

识的完成.相反地,有时认识刚开始,需要继续加深和完善.我们提倡,从认识构成系统关键功能的客观知识体系出发,逐步完善功能集合及其相应的知识结构.这是一种从简到繁、逐步增加的认识过程,并且在对构成关键功能的客观知识体系取得一定认识之后,可着手进行实现关键功能的前期设计,把认识和实现紧密结合起来,从而简化了认识和实现全部功能需求的复杂性.但是,最终的设计方案仍归结到全局的方案.模型技术的成功就在于它是一种通过构造原型,加速和加深对客观知识体系和实现方案的认识途径.我们希望软件开发方法是一个符合认识过程的指导软件开发的思想模式,体现的是一种开放性软件开发过程.

1.2　形式化方法

形式化方法[3]有2个方面的含义:一方面是从数学(数理逻辑、抽象代数等)导出的概念和技术;另一方面是应用这些技术的方法途径.目前,形式化方法和技术在前一方面有了较强的基础,然而,从方法学的角度看还很薄弱.注意,对一个形式化的规范语言或技术,可以有不同的应用方式.因此,从软件开发方法学的角度,形式化方法强调在软件开发周期中合理使用形式化概念和技术的途径.现有的现代软件工程途径的形式化程度大体上可以分成以下几种情况:(1)非形式化方法(包括半形式化方法).例如,常见的基于图形的软件开发方法:DFD结构化分析、SD结构化设计,复杂系统规范语言Statecharts.[4]这类情况是目前不少(商用)软件开发的主流之一.(2)基于形式化概念和记号的方法.例如,目前在软件可靠性工程和设计中流行的Cleanroom方法学.[5](3)严格的软件开发方法.例如,RAISE开发方法学[6],它包括了方法、规范语言RSL和机械化支持工具.此外还有Z[7],VDM[8]等.(4)完全形式化开发方法.它具有数学化的规范语言和支持定理证明与检查的环境,例如Nqthm[9]和PVS[10]等.

形式化方法在软件开发中的作用主要体现在2个方面:①描述:用规范语言编写文档、规范接口,这适用于软件开发周期各阶段.②分析:在抽象的基础上,确认或验证软件开发的主要设计决策和关键算法,形成软件体系结构.

1.3　出发点

由于关于软件开发周期的早期阶段的需求通常是非结构化的,容易产生错误.目前,在该阶段使用的方法以非形式化方法和半形式化方法为主流.最近的热点集中在面向对象方法和技术.[11]我们认为,面向对象和形式化方法沿2条线索发展,将两者结合起来是有益的,这是我们工作的出发点.当然,形式化方法与现代软件工程融合还需时间.

2　一种面向对象的软件方法框架

我们在下面给出一个严格的软件开发方法(图1),它较好地在形式化开发技术的基础上对软件开发中的创造和演算进行折衷,并集成了目前广泛使用的软件开发技术.我们所给的方法的一个特点是,在该方法指导下软件开发有着严格的形式化基础(可以在计算机辅助下做验证).软件开发分为以下3个步骤:面向对象的需求分析与建模;软件体系结构的生成;软件变换和软件重用.

2.1　面向对象的需求定义与建模

需求分析的目的是通过考察需求,分析含义,把系统的重要特征抽象出来,以严格的方

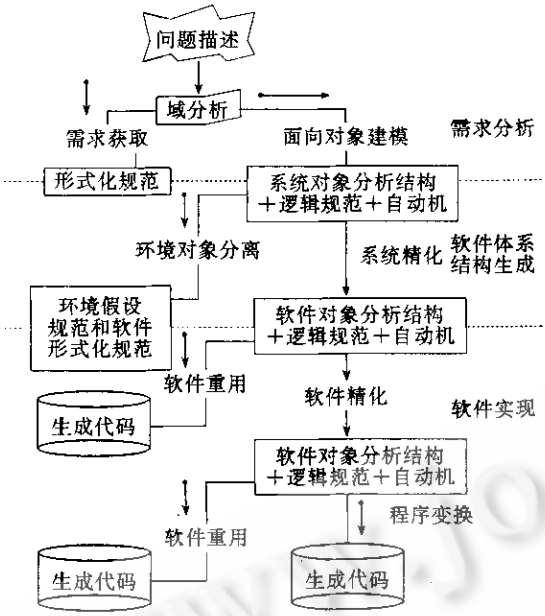

图1 软件开发方法示意图

式定义需求,并为现实世界建立模型.我们以面向对象思想为指导,提出了在域分析基础上进行需求定义和建模的方法,包括基于自然语言的域元素识别方法、面向对象可视化建模方法以及基于逻辑的全局需求规范方法.

(1)域元素识别方法

自然语言需求的描述有这样一些概念:名词、代词、直接宾语、间接宾语、形容词、副词.我们给出了域分析中的概念与自然语言中的概念的一般对应关系.

实体:如果与名词和代词的概念相比较,实体应该能完成行为(作为主语);能接受行为(作为宾语);能用来完成行为(间接宾语);能影响到对事实的陈述(作为不及物陈述的一部分).

动作:动作是指改变系统状态的一次操作.与动作有关的是主语、直接宾语,有时还涉及到间接宾语.一般动作如同及物动词.

关系:与关系相关的是非及物概念或陈述事实.

(2)面向对象可视化建模

作为域元素识别的后继,我们用面向对象的术语基于图的描述为问题域和问题建模.

〈1〉对于识别对象,可以有以下策略:识别具有问题域中意义的实体;使用聚集抽象,分出对象的层次;去掉别名:用一个名统一替代其他别名;区分对象和属性;去掉不属于最终产品的那些实体.〈2〉识别对象的行为,可以有以下策略:当对象间有请求关系时,则被请求的对象应具有接受该请求的接口;若环境中的一些事件需要某对象处理,则该对象应具有接受该事件的接口;从两方面入手:一个对象需要哪些服务,必须提供哪些服务;消息(服务或请求)可以具有条件前缀.〈3〉识别对象属性:指识别与特定应用相关的属性,即对象的性质.属性的取值决定了对象的状态.〈4〉对象状态转换建模:基于有穷自动机,描述对象状态和事件、消息传递的因果关系.〈5〉识别对象所属的类:根据分类思想,用类来抽象具有相似性质、行为的对象,并用继承、聚集等关系将不同的类组织起来.

我们提供一套图形可视化语言族来给出建模文档,它以多视点方式(基于字典和ERD的域结构、基于OMD的问题结构、基于FSM的对象需求)组织和描述问题域及待解问题.

(3)全局需求定义规范方法

对上一小节,我们作2点注解:①从建模的角度看,全局需求定义规范尤其是系统的性能规范在建模中被忽略了;②从形式系统的角度看,上节提示了关于待解问题理论的语言.在我们的软件方法学中,用时态逻辑规范来定义问题的全局需求,包括性能需求.注意到,对象本身具有一个被精化的问题,它亦能定义成一个子问题,因此,在需求定义阶段,鼓励在对象规范中引入时态逻辑规范.更进一步地,在需求分析阶段,我们以时态逻辑为基础建立整个系统模型的时态语义.另一方面,我们在早期阶段引入可操作的系统建模,其目的在于做

需求确认,包括在确认过程中加速对问题域的认识.

2.2 软件体系结构的严格生成

在问题建模和需求定义后,就得到了初始规范.根据逐步精化的思想,对应于一个规范,可能有多种实现.在每一个开发步中,沿用形式化演算的思想,我们希望由规范加设计决策导出一个实现.软件开发中的创造性被包含在设计决策中.软件体系结构由系统模型根据设计决策逐步生成,其严格性体现在2方面:体系结构本身的内部一致性和体系结构与需求规范的一致性.形式化验证中,为证明某规范是另一规范的实现或某规范满足特定性质,一个常用的途径是证明蕴涵式.然而,若规范不一致,则出现如"false → …"的平凡式.为此,我们研究了真值维护系统(Truth Maintanence System),并试图将其用于形式化规范的一致性维护的基础.对于规范协调性的判断,我们沿用数理逻辑中的2种方法,即基于理论解释的模型方法和基于保守扩张的良定方法.

与传统的软件开发相比,软件体系结构对应着软件初步设计,其设计导向是:①由设计决策给出体系结构及相应的软件开发的基本策略.例如,对于一个控制回路中数字化控制程序,它可以是基于"采样—计算控制—送控"的顺序结构,也可以是基于多进程"周期采样"、"周期送控"、"计算控制"的并行结构;②体系结构将在系统模型的基础上具有更好的可操作性和可实现性.

2.3 基于变换和重用的实现技术

在支持软件体系结构的实现中,我们的软件方法兼蓄了自顶向下和自底向上的实现,它们对应的技术即基于变换和重用的实现技术.这2种技术在软件开发的实现阶段是互补的.

(1)软件重用:自底向上的合成

随着软件开发和软件库的积累,合理地应用已有资源可以避免任何一个软件开发都是从头至尾开发新产品的过程.因此,对于软件开发中形成的软件规范,我们提供重用手段选用是否有满足该规范的软部件,如有,则重用该软部件构成系统中的元素;否则,采用精化方法对规范逐步开发(图2).

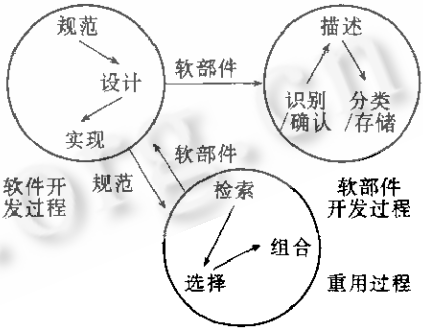

图2 软件重用过程

从辩证的角度看,系统的合成伴随着系统的分解,它是对软件体系结构分而治之的支持,也是将软件重用融入软件实现的手段.

(2)程序变换:自顶向下求精

毫无疑问,软件重用无法解决所有软件体系结构的实现问题.与此相适应的实现技术是自顶向下求精的程序变换技术.在变换的每一步,更多的设计决策加入到软件体系结构中,例如,消除非确定性和抽象算法的引入.特别地,对于性能指标分解的一般策略是,性能指标依赖功能指标的分解,我们将其归整为功能规范上的性能指标.在分解求精时,功能指标和性能指标分离,再在功能需求分解的基础上,形成新的关于子功能的性能指标.

3 方法的计算机辅助支持

程序工具和程序方法是一个问题的2个方面.方法是工具研制的先导,工具是方法的实

际体现，程序方法的研究成果最终实现为程序工具. 所以程序工具的研制也是特别重要的，而程序设计环境，就是在各个不同级别上的程序工具的有机结合. 对应于上节给出的方法学框架，我们初步在 PC 和工作站上建立了一个软件工具集，并准备在工作站上形成一个集成环境.

(1)需求分析[12,13]工具集　我们在 UNIX 和 WINDOWS 下建立了一组需求分析工具，包括从一段自然语言描述提取域元素进行面向对象分析并建立多视点的建模工具，含对象模型、对象状态模型和对象交互模型. 为了维护需求获取过程的一致性，我们给出了一个非单调的逻辑程序设计系统，它将 TMS 和 Prolog 有机地结合在一起从而从基础上支持需求规范获取这一认识过程的非单调性.

(2)软件设计[14,15]工具集　为支持软件设计中从系统模型到体系结构的生成，在需求工具的基础上提供了结构化的基于 Statecharts 的规范生成工具. 同时，为兼容传统的结构化设计，我们建立了基于数据流图生成结构化规范并以编译方式生成结构图的设计工具. 此外，在高阶定理证明器的基础上初步实现了一个证明开发环境. 在上述的工具集中，我们均支持层次化的求精和抽象，以支持软件开发中强调求精和抽象的层次化.

(3)软件重用[16]工具　在软件体系结构生成后，我们支持用软件重用的方法得到所需的软部件，一方面我们建立了基于域分析和规范的软部件的浏览、提取和生成工具；另一方面，还研究了一个被称为“幽灵”的监视器，它确保相当的软部件重用率，同时提醒用户软部件的规范化，以充实软部件库.

(4)验证[17,18]工具　严格的软件开发方法的一个优势在于能够对软件开发各阶段的支持进行演算，它包括传统意义上的验证. 我们开发了命题时态逻辑证明器作为抽象需求规范和设计决策的验证工具. 在此基础上，为应付软件设计的复杂性，利用高阶定理证明器开发了面向转换系统和时态逻辑的实时系统验证工具.

(5)程序设计环境　面向对象方法学是贯穿我们开发方法的一条主线，整个完成了一个面向对象的程序设计环境 GWOOSE，从而在整体上支持在面向对象框架下的软件开发.

4 结论和将来的工作

本文系统地提出了一种严格的软件开发方法，它是在现有软件工程方法上的一个发展. 在传统的开发方法的基础上，我们强调软件开发的演算性质，即尽管软件验证是十分昂贵的，但软件开发本身应当具有演算的数学基础；与完全的形式化开发相比较，我们侧重于集成软件工程师乐于接受的方式，如面向对象、自动机建模、多视点技术等，以便于软件开发中创造性的挖掘.

因此，我们希望“创造”和“演算”能在软件开发中得到一个平衡. 以此为基础，建立了一组较为丰富的支持工具，并尝试地用于一个地理信息系统的开发. 毫无疑问，还有许多进一步的工作，“创造”在软件开发过程中呈现一种跳跃的趋势. 事实上，软件生产率的提高依赖于这些跳跃的幅度降低、次数减少. 软件开发方法的工作是研究提供丰富的开发策略，使得软件开发过程平滑；目前的支持工具尚未集成为软件开发环境，有的还需进一步完善以处理复杂的软件开发；还有些问题需更深入的研究，例如软件规范的一致性检查和维护等.

本文目的之一是试图寻找一种介于软件形式化理论和目前实用开发方法的一个桥梁.

因为计算机应用领域的不同，软件开发的理论和方法亦存在差异.[19]我们正进行的是嵌入式系统中实时软件的开发工作[20,21]，它是文中软件开发方法面向应用领域的一项研究.

致谢　感谢 863 软件自动化课题组王戟、毛晓光、徐锡山、王献昌、贲可荣、陈晓桦、罗铁庚、王峰、杨树强、胡成军、毛新军、陆朝甫、王与力的工作.

参考文献

1 陈火旺，罗朝晖，马庆鸣. 程序设计方法学基础. 长沙：湖南科技出版社，1987.
2 冯玉琳，仲萃豪，陈友君. 程序设计方法学. 北京：北京科学技术出版社，1989.
3 Rushby J. Formal method and the certification of critical systems. Technical Report CSL－93－7, Computer Science Laboratory, SRI International, 1993.
4 Harel D. A visual formalism to complex systems. Science of Computer Programming, 1987.
5 Mills H D, Dyer M, Linger R. Cleanroom software engineering. IEEE Software, 1987, 4(5).
6 The RAISE Language Group. The RAISE Specification Language. Prentice Hall, 1992.
7 Spivey J M. Understanding Z: a specification language and its formal semantics. Cambridge University Press, 1988.
8 Jones C B. Software development: a rigorous approach. Prentice Hall, 1986.
9 Boyer R S, Moore J S. A computational logic handbook. Academic Press, 1988.
10 Owre S R, Rushby J, Shankar N. PVS: a prototype verification system. LNAI 607, 1992.
11 Rumbaugh J *et al*. Object－oriented modelling and design. Prentice Hall, 1991.
12 陈晓桦，陈火旺. 从自然语言描述的需求提取形式规范. 中国计算机学会第八届年会论文集，1992.
13 王献昌，陈火旺. 真值维护系统的语义研究. 中国科学(A)，1993，23(11).
14 谭庆平，陈火旺. 基于类型理论的递归元程序设计，软件学报，1994，5(8).
15 罗铁庚，齐治昌. PC/CASE：支持软件开发过程的 CASE 工具. 计算机科学，1994，(6).
16 毛新军，齐治昌. 软件重用研究与应用. 计算机科学，1994，(4).
17 Kerong Ben, Chen Huowang, Wang Binshan. PTL sequent calculus system. Science in China A, 1995, 38(1).
18 Hu Chengjun, Wang Ji, Chen Huowang. Towards a formal and mechanical verification of real－time systems. Proceedings of CICS'95, 1995.
19 Khalil Sima'an. Design principle for real－time process control systems. Report 94－42, Delft University of Technology, 1994.
20 Wang Ji, Chen Huowang. A formal technique to analyze real－time systems. IEEE COMPSAC'93, 1993.
21 Mao Xiaoguang, Chen Huowang. A refinement framework for hybrid systems. Proceedings of CICS'95, 1995.

A FRAMEWORK FOR RIGOROUS SOFTWARE DEVELOPMENT

Chen Huowang　Qi Zhichang　Wang Bingshan　Ning Hong　Tan Qingping

(*Department of Computer Science　Changsha Institute of Technology　Changsha*　410073)

Abstract　This paper proposes a framework for rigorous software development, which is based on stepwise design by refinement and composition by software reuse. In this framework, the authors integrate logic－and transition system－based formal method with visual formalisms, and get a reasonable compromise between creativity and calculus in software development. A set of computer aided tools has been built to support the approach in this paper.

Key words　Software development, formal method, design calculus.