第6卷第4期 软件学报 Vol. 6, No. 4
1995年4月 JOURNAL OF SOFTWARE Apr. 1995

一个面向对象的集成化软件原型开发环境*

马 江 何克清

(武汉大学软件工程国家实验室,武汉 430072)

摘要 软件原型方法弥补了传统生命周期方法维护难度和费用过大的缺陷,为软件开发提供了一种新的开发范例.实现原型法的关键在于能否提供一个合适的开发环境,帮助开发者实现原型定义、构造自动化,支持原型的快速生成.为此,我们在VAX Ⅱ/GPX图形工作站上设计并实现了一个面向对象的集成化软件原型开发环境SPE.本文介绍了该环境的设计、实现及特点.

关键词 软件原型方法,数据模型,操作模型,原型开发环境.

按照传统的生命周期"瀑布"模型(见图1)管理、开发软件,均是建立在这样两个假设基础之上:(a)用户能清楚地、完整地提供系统的需求;(b)开发者能完整地、严格地理解和定义这些需求.然而在实际开发中,大多数系统的需求,往往用户事先难以说清;开发者也由于这样或那样的主客观原因,难以跨越与用户交流的鸿沟.其结果是系统开发完毕后,需经常修改、维护的开销及难度过大.因此,从80年代开始,人们提出了许多新的软件开发范例(paradigm),以克服传统模型的不足,其中软件原型化范例就是其中最有影响的代表.它采用演示原型的方法来启发、揭示系统的需求,把用户尽早地卷入软件开发,通过使用、评价软件原型,及早得到反馈,更全面、更准确地了解用户需求,审定系统设计的可行性.尤其是对于那些具有冒险性、极难在实际环境中测试的应用领域,原型的意义更大(见图2).

实现原型法的关键在于能否提供一个合适的开发环境,帮助开发者实现原型定义、构造的自动化,支持原型的快速生成.为此我们在精练、借鉴目前国际上现有的原型开发方法、技术、工具基础上,设计、开发了一个面向对象的集成化软件原型开发环境SPE.通过一个友好的交互式、多窗口图形用户界面,操作使用一组丰富的工具集(编辑工具,浏览工具,查询工具,解释执行工具等),可方便地为用户提供表达能力强、易修改、直观形象的可执行的工作原型,达到支持面向对象的软件信息系统原型开发的目的.

1 SPE环境的内部模型

为了很好地支持面向对象的软件信息系统原型开发,我们为SPE环境提供了3个内部模型:面向对象的语义数据模型(OSDM)、基于statecharts的操作模型(SBOM)和用户界面

* 本文1991-03-13收到,1991-11-11定稿

作者**马江**,1964年生,讲师,主要研究领域为软件工程,原型化方法,面向对象方法等.**何克清**,1947年生,教授,主要研究领域为软件工程方法、工具、环境.

本文通讯联系人:何克清,武汉 430072,武汉大学软件工程国家实验室

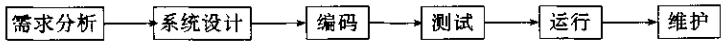


图1 传统生命周期model

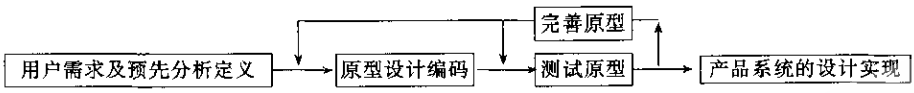
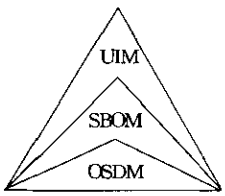


图2 原型开发model



OSDM:Object—oriented Sematic Data Model

SBOM:Statechares Based Operational Model

UIM:User Interface Model

图3 SPE内部模型关系图

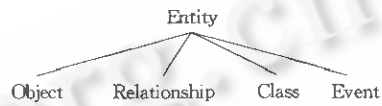


图4 4种不同的范畴

模型(UIM). 其关系如图 3 所示.

1.1 面向对象的数据模型 OSDM

为使软件环境中的信息库能有效地组织和管理各种软件信息,通常要求它的数据模型对软件环境提供以下几点支持:

① 能表示各种复杂的对象及对象间的复杂关系(Object & Relationship);

② 尽可能支持信息的可重用性(Reusability);

③ 能对复杂对象及其相互间的关系提供一致性管理或其它强有力约束(Constraint);

④ 能支持时间概念,对各种信息提供版本控制(Versioning)并支持"长事务处理"(Long transaction);

⑤ 能对过程信息的描述、分析及处理提供支持(Process Info.).

遗憾的是传统的数据模型,从文件系统、层次、网络到关系模型,都不能很好地满足这些要求,例如传统数据模型只支持简单的"记录"以及记录间的关系,而不能表示复杂的对象模型及其相互间的关系;只提供简单的内在约束(inherent constaints),并没提供描述约束的手段;还缺少时间概念,不能支持"版本"概念及"长事务处理";尤其是缺乏对动态过程信息的支持. 面向对象的思想强调良好的模块化、数据抽象、可重用性和信息隐蔽[2]. 这些优点非常适合于对软件信息的组织和管理. 目前,把面向对象的思想用于信息库日益成为一个研究的焦点. 如 Vbase[3]、SIB[4]等. 为此我们提出了一个面向对象的语义数据模型 OSDM 来弥补传统数据模型的不足.

1.1.1 OSDM 概述

为了能清晰描述各种软件信息和现实世界的问题,OSDM 提供了 4 种不同范畴的类型管理. 从概念上讲,现实世界任一实体须属于某一类型,而每个类型又须属于这 4 种范畴之一,即 OSDM 是严格类型化的. 这 4 种范畴是:对象(Object);关系(Relationship);集合(Class);事件(Event). 如图 4 所示.

Entity:OSDM 中的该概念,类似于 Smalltalk 中的 Object,是一种抽象概念;

Object:OSDM 中的 Object 主要指那些"实实在在"的实体. 如软件开发中的各种产品信息(如规范、文档、代码等);

Relationship：用于表示一组对象间的关系，通常一个关系是双向的(即关系与其逆关系)；

Class：由一个谓词所决定的一组对象组成.这些对象称为集合的元素；

Event：指在一段时间内对一些对象产生影响的过程，通常由一组操作(如读、写等)通过控制结构(顺序、分支、循环等)连结而成；

Type：这是一种分类抽象.是将一组具有相同属性和相同操作的对象抽象而成.这些对象称为该类型的"实例"(Instance).

OSDM通过这些不同范畴的结合来对现实世界的事物进行描述.

1.1.2　OSDM的主要特征

1.提供了4种数据抽象

(a) AIO(An—instance—of)分类抽象：OSDM严格类型化的手段广泛地支持了这种抽象.因为在OSDM中，软件信息根据其结构抽象为不同的类型，而软件开发中产生的大量信息作为这些类型的实例，这种不仅使各种软件信息有机组织在一起，而且使这些实例共享其类型的结构和操作.

(b) AKO(A—kind—of)概括抽象：OSDM用Super/Subtype的概念来组织同一范畴的各种类型，使这些类型通过AKO关系形成类型层次，故某类型的实例能通过AKO继承其Super类型的属性及操作.

(c) APO(A—part—of)层次抽象：OSDM不仅提供对象和属性之间的抽象关系，而且还针对软件开发的特点，特别对类型增加了一个"APO"性质来支持APO抽象，例如，汽车除了其型号、颜色等属性外，还可以指出汽车是由车身、引擎、车轮组成.

(d) AMO(A—Member—of)联合抽象：OSDM中，特将"集合"作为一类型范畴，将集合也类型化，这样对联合抽象提供有力的支持.

2.继承性

数据抽象的核心在于它们的继承性，以上几种数据抽象，都有对应的继承，体现出OSDM所支持的相应可重用性.

3.语义约束

OSDM提供了丰富的约束，不仅能对属性提供相互约束，还能描述多个属性间的约束，并能提供关系约束.

4.传递关系及传递闭包

OSDM对传递关系提供了特殊的支持，即生成传递闭包(transitive closure)，这对查询、浏览提供了有力的支持.

5.过程信息

OSDM采用statecharts对事件进行描述，对过程信息表达能力大大提高.

6.版本控制及长事务处理

OSDM采用"Delta"技术和过滤程序技术不仅能提供对象的版本控制，还能对类型提供版本支持，因而也对"长事务处理"提供了支持.

1.1.3　OSDM同相关数据模型的比较

同其它面向对象的数据模型(OODM)一样，OSDM继承了面向对象程序设计风格的优

点,即良好的可重用性、数据抽象、信息隐藏、继承性等,它同这些模型的比较可由表 1 所示.

表 1　OSDM 与相关数据模型的比较

DM 提供的支持	Vbase	SIB	OSDM
Object－Oriented	√	√	√
"Relation"	简单关系	√	√
data－abstraction (aio,apo,ako,amo)	没有明确支持 AMO 联合抽象	√	√
Constraint	属性约束	√	多种约束
Reusability	√	√	√
Versioning	√	√	√
Long－trans.	√	√	√
Process Info.	简单的动态过程	状态图	扩充 Statechart (Event)类型

1.2　操作模型 SBOM

对复杂的系统做快速原型,最大的困难之一在于能找到一个既能描述实际应用中软件系统的行为特性,又能直接操作这些描述,以达到快速原型目的的强有力的操作模型. 目前公认为较成熟的操作模型有 Gist 和 PAISLey,前者只能用来描述系统的功能持性,不能用来描述交互性的用户界面;后者主要用来描述嵌入式系统与其所处环境的关系. 因而,这两个模型不便于描述原型里用户所关心的那些较具体的操作行为. 我们的原型环境面向的领域主要是软实时系统(或称反应式系统),其主要特征是交互性强,强调的是其动态行为方面(dynamic behavioural aspects),而以色列 D. Harel 提出的 statecharts 的特长恰在于描述反应式系统的行为,因此我们选用了它作为 SPE 的操作模型的基础.

1.2.1　statecharts 概述

statecharts 是一种适于描述复杂行为过程,基于状态的图示方法. 它是在结合 Mealy 机和 Moore 机的基础上,加进了"层次(hierarchy)和"正交"(orthogonality)的概念;引入了"广播通讯"(Broadcast Communication)的机制,其主要特征可表示为"Statecharts =状态图＋层次＋正交＋广播通讯"[5].

1.2.2　SBOM 对 statecharts 的发展

为了适合于一般软件原型的开发,也为了更好地与数据模型集成,我们对 statecharts 作了如下扩充、修改:

1. 增加了类型的概念

为了使各种事件、过程的描述也可重用,我们给 Statecharts 中的状态赋以类型(type)的概念,把同样的或基本相同的事件抽象定义为一个类型,这样,若定义好了"田径比赛"这个事件,不管是奥运会、亚运会,还是全运会,均可重用已定义好的事件.

2. 与数据模型的结合

我们利用 Statecharts 观点在数据模型中增加了新的范畴——"事件",且"事件"除了类型外,同样有类型层次,继承等机制,同样可对它进行查询、浏览,这样使得数据模型除能对

对象、集合、关系等静态信息描述外，还能对动态信息——“事件”进行描述.

3. 状态活动的分类

为了更好地支持状态类型(state type)及层次的概念，SBOM 把状态分为 4 种：

并发状态(concurrency)：用来描述系统的并发行为；

串行状态(sequential)：用来描述系统中串行状态间的控制流；

函数状态(funtional)：用来描述以软件方式实现的状态；

手工状态(manual)：用来描述以手工模拟的状态；

函数状态和手工状态是不可再分的状态，称为原子状态. 并发状态由“与”(AND)分解来描述并发行为，串行状态由“异或”(XOR)分解来描述状态间的控制流.

4. 对功能性(functionality)的描述

主要扩充了以下两个方面：允许功能状态中的活动有明确的输入、输出以及对此功能的概要描述；允许转换中的“刺激事件”和“条件”中的表达式可以自由地使用信息库的信息，这样 SBOM 能更自然地描述诸如数据处理领域等其它类型软件系统的原型.

5. 对刺激事件的分类，主要分为瞬间刺激与持续刺激事件

6. 简单的语义检查，包括检查不确定性、不一致性及死锁等语义问题

7. 全数据库支持

SBOM 具有来自 OSDM 提供的所有支持，包括事件存贮，类型检查、继承性、浏览与查询能力及版本控制.

总之，SBOM 能够描述应用系统中内部功能与外界的动态关系，也能够描述应用系统中某一功能的操作行为. 它对软件原型开发的支持在于通过 statecharts 解释器执行 statech－arts 来展示应用系统功能的操作行为，使用户及设计者能依据原型的运行来制定明确的用户需求规范.

1.3 用户界面模型 UIM

在数据模型 OSDM 及操作模型 SBOM 的基础上，我们建立了 SPE 的用户界面模型. 为应用系统快速构造人机交互界面原型提供支持. 它主要由基本图形界面的对象集和操作集组成.

1.3.1 基本图形界面对象集

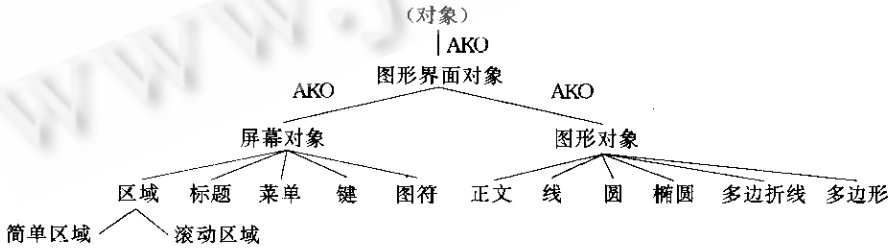


图5 基本图形界面的对象集的类型AKO层次图

其中屏幕对象(Screen－objects)用于图形用户界面的画面构造，如信息显示区域的安排. 图形对象用于构造显示给用户的交互信息，如正文提示或统计图表.

1.3.2 基本图形界面操作集

这些操作包括:Render:显示图形用户界面的画面;Move_on_screen:移动图形画面至一新位置;Remove_from_screen:抹去图形画面;Resize:按比例放大、缩小;Iconify:将图形画面缩成一幅小像;Deiconify:将像还原成显示画面;……

2　SPE 环境的系统结构

1. 正文编辑器

SPE 的正文编辑器包括 4 类子编辑器:对象编辑器(object Editor),关系编辑器(Relationship Editor),集合编辑器(Class Editor)和事件编辑器(Event Editor). 每个子编辑器又分为:(a)类型编辑(type);(b)实例编辑(instance). 因此,共有 8 个子编辑器,每个子编辑器都具有较完善的编辑功能.

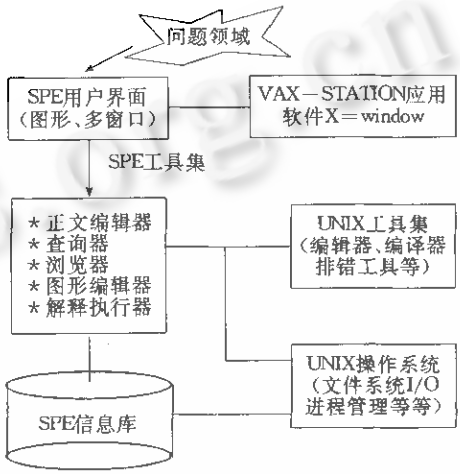


图6 SPE环境的系统结构图

2. statecharts 图形编辑器

为了更形象、直观地体现 statecharts 这种图形描述方法的优越性,并更有效地支持 SPE 环境中原型的快速生成,我们研制开发了 statecharts 图形编辑器. 它选择了面向用户、鼠标驱动、弹出式菜单和固定菜单的编辑方式,采用了在全屏幕方式下的自动语法检查的机制. 提供了较完善的图形编辑功能.

3. 查询器

在 SPE 环境中,能支持简单的事实性查询(factual query)和复杂的演绎查询(deduction query),演绎查询又称逻辑查询,是根据信息库中已有的信息推导出信息库中未直接存放的信息.

4. 浏览器

SPE 环境还提供了直观的图形浏览手段. 它不仅允许用户利用类型名、实例名、属性、AKO、APO 等已知信息来查看所需的信息,还允许他通过 AKO、APO 去查看他所感兴趣的类型层次图,浏览类型表中所有的类型或一个类型的所有实例或某个类型的完整结构. 所有的图形浏览操作均由鼠标驱动,可以根据需要上、下、左、右,灵活方便地浏览整个层次图,还可以利用关系的传递闭包来查看他想要的信息.

5. 解释执行器

编辑好的 statecharts(指 statecharts 描述的应用系统的规格说明书)还可能存在两类问题:((a)语义上的错误(如不一致性);(b)用户/软件工程师所做的规格说明不符合用户的需求. 为此,SPE 环境提供了解释执行器,来执行 statecharts 模拟原型的动态行为以发现语义上的错误.

解释执行器的工作过程为:解释执行器首先决定是否存在可激活的转换. 若无,则取消旧的瞬间刺激事件,而等待新的刺激事件;否则,执行所有激活的转换且产生新的状态. 若有些新状态是原子状态,则执行它们的活动. 在此期间,输出转换产生的刺激事件,送到广播站

中(以广播通讯形式)输送出去,这样才完成一个模拟步.解释执行器持续上述的模拟步直到终止整个事件或被控制部件中断.

在此模拟过程中,使用者相当于真实系统所在的外部环境.使用者发出的信息就相当于statecharts中"外部刺激事件".如果由软件、硬件实现的状态活动越多,由人模拟的状态活动越少,这个原型就越接近于真实的系统.

6. SPE环境中的用户界面

SPE环境用户界面主要采用了图形多窗口技术,层次模型的Zoom技术,菜单驱动技术,图形会话技术等等.为SPE环境各个工具提供一致的、良好的用户界面,并且为用户方便灵活地利用环境生成所需的工作原型提供了强有力的支持.

7. 信息库

在数据模型OSDM基础上,我们建立了SPE环境的信息库,用来对原型静态和动态信息进行组织和管理.

3 SPE环境的使用方法

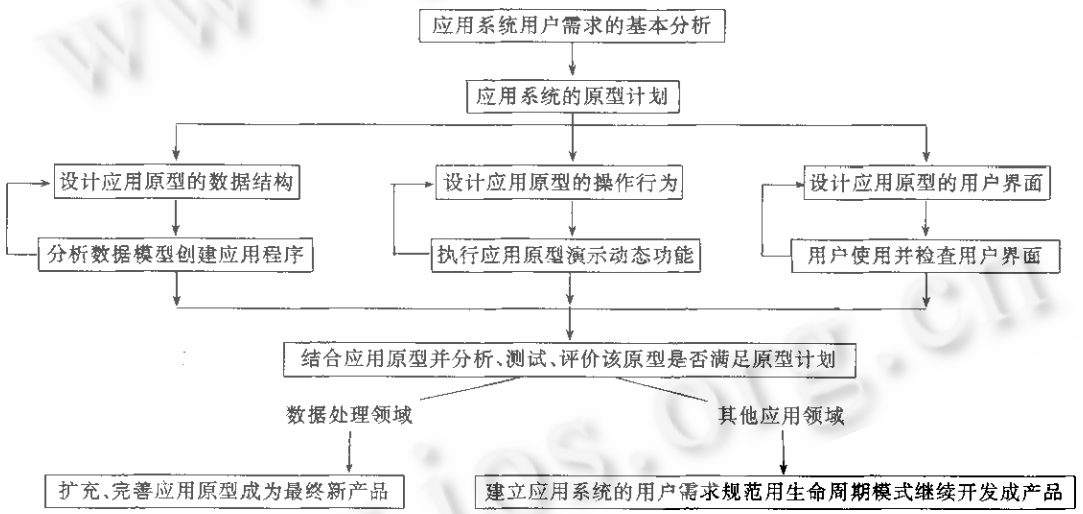


图7 SPE环境的使用方法

下面我们以一个蓝球运动会管理系统BTMS(Basketball Tournament Monitoring System)为例,说明利用SPE环境进行应用系统原型设计、开发的主要技术/步骤:

1. 设计应用原型的数据模型

开发者首先需要利用SPE环境的正文编辑器,定义BTMS原型的所有对象(Object)类型,关系(Relationship)类型和集合(class)类型.如图8所示.

2. 设计应用原型的操作模型

开发者利用图形编辑器定义BTMS原型的操作形为(事件类型).如图8所示.假定有8个队参加BTMS,我们可以将篮球运动会(TOURNAMENT)作为一个大的事件类型,篮球比赛过程(B—GAME)也作为一个事件类型(B—GAME的实例表示一场特定的比赛).所有事件类型可如图9所示.

Object types	person, bt—employee, administrator, refeeeree, athlete, team_staff, player, game, game_score, game_statistics, bt_statistics, coutr, ceremony.
Relationship types	(game)——uses——→(court) (player)——plays_in——→(game) (player)——has_stat——→(stat.) (team)——has_stat——→(stat.) (team)——has_score——→(g_score) (referee)——conducts——→(game)
Class types	administrators, referees, all_athletes, team, referees_per_game, courts.
Event types	B—game—event, bt—event, court—status, ceremony—event.

图 8 BTMS 原型数据模型中的所有类型

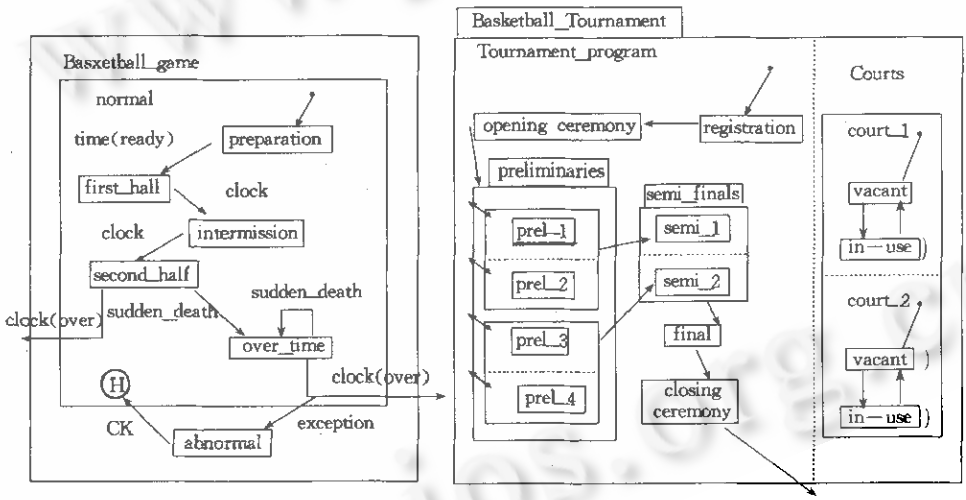


图9 运动会(TOURNAMENT)和篮球比赛(B_GAME)的事件类型

3. 设计数据模型的应用程序

开发者设计特殊的应用程序. 如设计产生赛程安排的应用程序.

4. 说明原子状态的活动

对原子状态的活动,采用执行函数或手工模拟来描述. 例如,比赛注册的活动,我们可采用人为的输入球队的名称,并存入相应数据库来手工模拟完成.

5. 产生数据模型和操作模型的实例

利用正文编辑器和图形编辑器产生原型真正的运动员、队及赛程. 同时产生 4 场初赛,2 场半决赛以及一场决赛,并将之存入信息库中.

6. 分析测试应用数据模型

利用查询器、浏览器对整个数据模型以及所有的实例进行检测,发现可能的错误. 例如,一个场地同时分给两场比赛等. 若有错,则重复 1、3、5、6 进行修改完善.

7. 模拟演示评估操作模型所体现的动态行为

利用解释执行器模拟演示. 若出现异常行为,则表明操作模型在设计上有错误,例如,解释执行器无法退出一个状态. 若出现错误,则重复步骤2、4、5、7修改完善. 周密地选择、设计测试情形(test case)对评估操作模型是非常有益的.

8. 设计定义应用原型的用户界面

利用SPE环境中用户界面模型UIM提供的支持,构造BTMS的用户界面.

9. 使用并检查用户界面

利用解释执行器直接操作用户界面. 发现错误及时修改完善.

10. 综合评估原型是否满足原型计划

对数据模型的反复修改设计会影响操作模型,反之亦然. 因此,需将数据模型,操作模型和用户界面模型集成在一起,同原型计划比较综合评估. 我们尤其强调在评估过程中用户的参与. 只有当用户和开发者均认为原型目标满足后,一个原型才被认可.

利用上述方法设计出的原型,既可以用来进行需求分析或设计方案论证,也可以逐步求精进化为最终产品.

4 环境的特点

SPE环境具有如下的特点:

1. 基于面向对象的语义数据模型(OSDM)的软件信息库,具有很强的描述能力,能支持高度数据抽象、信息隐蔽、可重用性及版本控制,支持面向对象信息系统原型设计过程中各种原型信息(程序、版本、文档等)的构造、存贮及管理;

2. 采用基于statecharts的操作模型(SBOM),对应用系统控制过程和动态行为,提供了极其自然、形象的描述;

3. 数据模型OSDM、操作模型SBOM及用户界面模型UIM的有机结合,为原型定义、开发测试提供了强有力的支持;

4. 图形、多窗口用户界面,图形编辑工具、解释工具、查询和浏览工具为用户使用SPE环境开发、测试所需原型提供了灵活、方便的支持.

5 结束语

本文介绍了一个面向对象的集成化软件原型开发环境SPE的设计与实现. 我们已经利用SPE开发了一系列的原型,其中一个较典型的大型实例是原型化一个运动会管理系统. 该系统能组织管理资源分配,赛程调度及控制整个运动会进行. 虽然这是一个相当复杂的系统,它有各种各样的数据对象及其之间的关系等,如运动员、代表队、项目、赛程、成绩及其之间的关系等. 整个运动会系统大约有100个类型(object约40个,Relationship约10个,class约40个,Event约10个)以及近千个实例. 但SPE环境成功地支持这个大型系统原型的开发,显示出其强有力的描述能力和动态模拟、演示原型的能力.

尽管SPE环境能在较大的实用领域中支持原型的快速开发,但是由于环境开发时间的限制,尚有一些有待进一步完善之处. 例如,由于缺乏数据定义语言DDL和数据库操作语言

DML. 目前，SPE 中的所有工具都只能以交互式方式提供给用户. 今后的发展主要包括以下几个方面：(1)设计一个面向对象的数据定义语言(ODL)；(2)设计一个方便的与 C 语言有良好接口的数据库操作语言(OML)；(3)用真正的商用 DBMS 代替目前的 UNIX 文件系统；(4)将 OSDM 与 AI 中知识库(KB)相结合. 相信随着进一步完善，SPE 会有更广阔的应用前景.

参考文献

1 Harel D. Statecharts: a visual approach to complex system. Technical Report CS84—05 The weitzman Inst. of Science, 1984.

2 Halbert D C, Obrien P D. Using types and inheritance in object—oriented programming. IEEE Software, Sep 1987.

3 Ontologic Inc. Vbase technical overview. Ontologic Inc. 244, 1986.

4 Kuo J H. Information structure for software environments. LNCS, 1986.

5 Harel D. Statecharts: a visual eormalism. CACM, June 1987.

6 Penedo H P. Prototyping a project master database for software engineering environments. ACM SIGPLAN Notics, 1987, **22**(1).

7 Rudmic A. Choosing an environment data model. LNCS, 1986.

8 Balzer R, Goldman N M, Wile D S. Operational specification as the bassis for rapid prototyping. ACM SIGSOFT Software Engineering Notes, 1982, 7(5): 3—16.

9 Taylor T, Stanish T A. Intial thoughts on rapid prototyping techniques. ACM SIGSOFT Software Engineering Notes, 1982, 7(5).

10 何克清. 计算机软件工程学. 武汉：武汉大学出版社，1983.

11 王俊彦，郭鸿志. 论数据模型在软件开发环境中的作用. 武汉大学学报自然科学版，1988，(增刊).

12 陈冀军，郭鸿志. Statecharts 作为软件原型的一个操作模型，武汉大学学报自然科学版，1988，(增刊).

13 杨芙清等. 软件工程支撑环境中的集成化问题研究. 计算机科学，1987，(4).

AN OBJECT—ORIENTED INTEGRATED SOFTWARE PROTOTYPING ENVIRONMENT

Ma Jiang　He Keqing

(*State Key Institute of Software Engineering, Wuhan University, Wuhan* 430072)

Abstract　The software prototyping approach was proposed as a new paradigm to remedy some drawbacks in traditional life—cycle model. The challenge is to develop a suitable integrated prototyping environment which enables the developer to specify and construct a prototype easily. Therefore, the authors designed and implemented an object—oriented integrated software prototyping environment (SPE) on VAX II/GPX workstation. This paper presented the design, the implementation and the main characteristics of SPE. The users and designers can draw a precise and correct requirement specification and verify the possibility of the application system design after running a prototype.

Key words　Software prototyping method, data model, operational model, prototyping environment.