第7卷第8期 软 件 学 报 Vol. 7, No. 8
1996年8月 JOURNAL OF SOFTWARE Aug. 1996

软件过程建模语言研究*

柳军飞 唐稚松

(中国科学院软件研究所 北京 100080)

摘要 本文介绍了软件过程建模的基本概念,提出了对软件过程建模语言的基本要求并简要介绍了几个有代表性的过程建模语言,给出了一个基于时序逻辑的形式化过程建模语言XYZ/PME,该语言是时序逻辑语言XYZ/E的子语言,它支持以角色为中心的逐步求精的过程建模方法,可在统一的形式框架内表示不同抽象级的过程模型.

关键词 软件过程,软件过程建模,过程建模语言,时序逻辑,程序设计语言.

如何以合理的性能价格并在预定的时间内开发出高质量的软件是30余年来软件界一直在努力解决的问题.从60年代侧重于编码阶段的研究到70年代广泛采用软件工程学的观点,及至80年代对软件工具和环境的研制,注意力主要集中在软件制造过程的产品上,提出了多种方法和技术来提高软件产品的生产率和质量,希望质量的提高减少开发次数,降低维护费用,从而达到降低整个软件开发周期的费用.但随着软件项目规模的越来越庞大,系统也日益复杂,这个问题远未得到较好的解决.近几年来,注意力逐渐转移到制造软件产品的过程本身——软件过程(Software Process),这是基于如下的认识:软件和其它工业产品一样,其生产率和质量的高低最终取决于生产产品的过程的好坏,故通过对软件过程本身的研究,改进软件过程的质量,可以使软件项目的实施更有效,更可预见,获得高的生产率和高质量的软件.

软件过程(以下简称为过程)是建立、维护和进化软件产品的整个过程中所有技术活动和管理活动的集合.也可以认为,一个软件过程是一组受约束的协同活动,其中一些活动是自动执行的.[1]

目前,软件过程的研究日益受到重视,其首要问题是如何表示过程,即建立过程模型;其次是在模型建立之后,如何实施(Enact)过程;进一步的工作则是研制以过程为中心的CASE环境.过程建模语言是贯穿这些问题的核心,本文侧重对其进行了讨论,给出了对过程建模语言的基本要求,简要介绍了几个有代表性的过程建模语言,最后介绍了我们在这方面的一些工作.

---

* 作者柳军飞,1965年生,副研究员,主要研究领域为软件工具与环境,面向对象技术,软件过程技术.唐稚松,1925年生,研究员,博士导师,中国科学院院士,主要研究领域为计算机科学与软件工程.

本文通讯联系人:柳军飞,北京 100871,北京大学计算机系

本文1995-06-09收到修改稿

1 过程建模概念及其对建模语言的要求

软件过程模型是过程的抽象描述,对过程的描述可以是形式化、半形式化或非形式化的,如利用结构化语言或图形来描述过程则是半形式化的.一个过程模型通常表达了一定的过程抽象层次和看待过程的一种特定观点.过程建模的首要目的是为了更好地理解过程,精确地表示过程,从而使过程的参与者可以有效地交流、合作;其次则是试图自动地控制、指导和帮助参与过程中的人或工具,并可使一些开发活动可以自动地重复进行;最后,过程模型为过程的修改和评估提供了良好的基础.

一个软件过程包括许多方面,依据过程建模的目的择其重要的方面进行模型化才有意义且实际可行.Curtis[1]认为,过程的如下4个方面是重要的:组织、功能、行为和信息.Conradi等人[2]认为一个过程模型至少包括5个子模型:活动模型、用户模型、组织模型、产品模型和工具模型.

过程建模的主要方面是建模方法和建模语言,方法和语言是紧密相关的.目前主要的建模方法有:基于过程控制流的建模方法,基于角色的建模方法和基于模板的建模方法.基于过程控制流的建模方法将一个过程视为由若干步骤(过程步)组成的工作流程,先确定过程步之间的执行顺序关系,再以过程步为中心收集过程的其它数据来描述过程;基于角色的建模方法则认为一个软件项目是由一些角色(指参与过程中的人或工具,如程序设计员,测试工程师等)来协同完成的,故先描述参与过程的各个角色、角色的行为及角色之间的协同关系,然后再以角色为中心收集过程的其它信息来建立过程模型;而基于模板的建模方法是先确定过程中的各类实体(如事件、角色、产品等),将这些实体定义成模板,然后再针对每一个模板在各个不同的抽象层次上构造该模板的具体对象,最后再定义各类实体在同一抽象层次中的所有对象之间的各种关系.

过程建模语言是过程的表示工具,是过程技术的关键所在.由于软件产品和其生产过程不同于一般的工业产品及其生产过程,要达到过程建模的目的,具有严格的形式化基础的建模语言是必要的,图形语言和文本描述可作为辅助手段.L.Osterweil提出"软件过程也是软件"[3],认为软件过程建模是程序设计的一种形式,该观点得到广泛的赞同.本文基于此观点,考虑到过程建模的特点,认为过程建模语言应满足如下几点:

模块性,由于"软件过程也是软件",模块性要求是自然的;**抽象性**,支持不同抽象级的过程描述,抽象非相关的细节;**求精**,过程本身是非常复杂的软件系统,从高层的抽象表示到低层的详细描述是一系列求精的结果,这样可控制复杂性;**分布性**,一个软件过程往往是由多个角色在不同场所共同完成的;**并发性**,软件过程中的活动往往是多个角色的并发行为;**智能性**,软件过程中的活动序列是动态的,非确定的,过程实施中依据一定的条件必须作出判断和选择.

2 几个具有代表性的过程建模语言

目前在各种文献上介绍过的过程建模语言已达数十个,从表示风格上可以分为以下几种类型:逻辑规则,也称为基于规则的语言或逻辑语言;命令式程序设计语言;自动机,包括

有穷状态自动机和Petri网;人工智能,指采用人工智能技术的语言等.一些语言兼具不同的风格,本节介绍几种有代表性的语言.

2.1 ALF

在ALF[4]中,过程模型被描述为MASP(model for assisted software process)的分级结构,每个MASP描述了软件过程模型的一部分,且每个MASP又可被其它MASP进一步细化,这样可以建立不同抽象级的过程模型.

每个MASP描述为一个六元组(Om,Op,Ex,Or,Ru,Ch),其中Om为对象模型,它提供了一个基于ERA(实体、关系和属性)的概念数据模型;Ex是表达式集合,用一阶逻辑语言表示;操作符类型(Op)、规则(Ru)和特征(Ch)均使用一阶逻辑语言表示,操作符类型采用前置和后置条件描述过程活动的语义,规则定义了过程中的某些特定状态下可能的自动反应,特征说明了过程状态上的约束条件,如不满足,则引发一个异常;最后,Or是排序集合,采用路径表达式定义,它表示操作符的执行是并行、选择还是顺序的.

2.2 APPL/A

APPL/A[3]是一个过程程序设计语言,它是ADA的一个超集,在ADA的基础上增加了软件对象之间永久性关系的定义机制、关系操作的触发机制和表示关系状态的谓词机制;同时,APPL/A继承了ADA语言的基本特征,如类型系统、模块定义形式(包块)和任务通讯方式(会合机制).

在APPL/A中,软件过程描述以程序过程的形式给出,而对某些方面(如一致性条件)则可采用规则形式来表示.

图1给出了一个APPL/A软件过程程序的一部分:

```
Function resultOK(result,needed);
    declare
      result derived-result;
      needed req-output;
    if fcnOk(result. fcn-output,needed) OR
      (result. observ-timing > neede. time)
      then resultOk:=False;
      else resultOk:=True;
    endif
endresultOK;
```

图1 一个APPL/A例子

2.3 HFSP

HFSP(hierarchical and function software process description and enaction)[5]提供了一种基于属性文法理论的过程建模方法.其基本思想是:活动是把输入对象变换为输出对象的函数,一个过程被定义为上述数学函数的集合;活动可按语法规则进一步分解,而输入-输出关系采用属性规则来描述,一个活动的输入输出属性必须与它的子活动的相应属性一致.更精确地说,如果一个活动A的输入输出分别为x1,x2,...,xn和y1,y2,...,ym,那么,可以用一语法非终结符A表示,A具有继承属性x1,x2,...,xn和综合属性y1,y2,...,ym.

HFSP通过对函数输入属性的并行求值支持并发,同时,同一活动可有2条以上的规则来说明,以表示活动的非确定性.语言也可表示和对象库的接口,即在A(x1,x2,...,xn|y1,y2,...,ym)中,可从对象库中读取输入对象和将输出对象存到对象库中.

2.4 Marvel

在Marvel[6]中,过程建模基于规则进行,规则分为3类:项目规则集,它描述过程特定的问题;项目类型集,采用面向对象的类定义数据;项目工具集,它描述和外部工具的接口.

项目规则集由2类规则所组成:激活规则和推理规则.激活规则控制开发活动的起动,特别是包括工具的调用,激活规则通常有多个互斥的结果,说明活动在不同条件下可以结束.例如,一个表示编译器调用的规则有2个结果:编译正确,产生目标代码;编译失败,产生错误提示信息.推理规则只用于定义对象属性之间的关系,活动的激活部分为空,只有1个结果.

2.5 SLANG

SLANG[7]是建立在一个高级Petri网体系(称为ER网)上的过程建模语言,它通过网的标记描述分布的状态概念;转换表示给定状态下可能或不可能出现的事件;转换点火表示一个事件的出现,事件出现的条件是局部的,通过输入位置来描述,网络拓扑结构描述事件的优先关系,也描述了并行性和冲突情形.位置被认为是分布式永久对象库,它模型化数据、工具和资源等.对事件出现时活动执行的描述采用类似于逻辑语言的形式表示.

总的看来,类似于上述语言的过程建模语言均是基于各自对过程的认识而提出的,均能描述过程的一个侧面和多个侧面,且随着对过程认识的深入,语言也在不断改进和成熟.但目前尚无一个语言完全满足过程建模的要求且可实用.

3 XYZ/PME

XYZ/PME(XYZ/E for process modeling)是时序逻辑语言族XYZ/E[8,9]的一个子语言.XYZ/E适应多种程序设计风范,可以在统一的形式框架下表示不同层次的抽象描述和可高效执行的算法过程,适于在不同抽象级进行程序设计,为过程建模提供了严格的形式化基础.XYZ/PME侧重于过程建模,并具有面向对象的特征,较好地满足了过程建模的要求.

3.1 XYZ/PME过程建模概念框架

我们认为,一个软件过程是由若干角色来共同完成的,是这些角色按一定时序关系对产品进行加工的一个生产过程,故通过描述角色的动作、行为及角色之间的相互协作关系可以描述过程的主要方面.同时由于软件过程的实际参与人员和软件过程建模人员对软件过程的认识有一个逐步深入的过程,且从不同角度、不同时间和不同抽象层次去观察软件过程,所看到的细节不同,因此很难从一开始即给出一个过程的详尽描述,逐步求精的建模方式是解决该问题的一个较好方法.XYZ/PME支持逐步求精的以角色为中心的过程建模方法,该方法实质上模型化了Curtis的4个方面.

在XYZ/PME的过程建模概念体系中,软件过程是一组角色的活动集合,角色可以是参与过程的人,也可以是工具,如程序员、测试工程师和调试工程师等.角色是抽象的概念,同一工程师在不同时刻可以扮演不同角色,角色的划分同时是一种任务的划分,随着对角色的进一步分解,任务也被分解为更具体,角色之间的关系表示了任务执行过程中的一种动态关系.

每一角色的XYZ/PME描述为一个Agent(见下节),角色的描述由2部分组成:角色的活动描述和角色所处理的过程对象.过程对象指软件开发过程中的所有实体(如各种文档),

采用包块(Package)来定义,包块封装了数据和操作,包块的数据部分定义过程对象的属性以及该对象和其它过程对象的关系,操作部分定义对该过程对象的操作,操作以XYZ/PME的程序过程或谓词表示,在逐步求精的过程中,可以只先给出前置条件和后置条件,而留待具体算法实现时再给出完整的描述;过程对象可具有永久性.角色的活动描述为一个XYZ/PME进程,进程定义角色之间的关系和角色的行为,角色之间的关系利用通道通信来表示;角色的行为直接用一组XYZ/E语句来描述.一个XYZ/PME过程模型是一个XYZ/PME程序.

过程建模的具体步骤如下:

(a)标识软件开发过程中的对象,分析和定义对象之间的关系和对象的属性,定义施加于对象的操作,定义软件过程对象模型.为了控制复杂性,每一对象的操作定义可按逐步求精的方式,先给出抽象表示,随着对对象的分解,一个操作被分解为几个操作,逐步引入细节描述.

(b)标识第0级角色,分析该角色所处理的对象、行为及其与外界的联系,给出其形式化描述(Agent定义).一般而言,该级角色只有1个,即项目组.

(c)进一步分析和提炼实际过程,对$i-1(i=1,2,3...)$级角色进行求精,包括对$i-1$级角色进行分解,描述i级角色所处理的对象、行为、关系,对$i-1$级中的角色,如在i级中未被分解为多个角色,则在i级中仍需表示为i级角色,并依据i级的其它角色对其行为、关系进行求精.

(d)抽象级i的过程模型即由抽象级i的角色及角色之间的协同活动的XYZ/PME表示所组成,协同活动是一组XYZ/PME并发进程.

(e)重复(b),直到所有角色均已精确地表示.

3.2 语言机制简介

(1)基本语句和基本程序单元

XYZ/PME的基本语句和基本程序单元即为XYZ/E的所有基本语句和基本程序单元,XYZ/E的基本语句是逻辑型合式公式(wff)的一种特殊情形.XYZ/E中逻辑型合式公式中除了允许出现一阶逻辑中常见的逻辑连接词、量词(~(非),∧(与),$∨(或),→(蕴含),==(等值),$A(全称量词),$E(存在量词),$T(真),$F(假)),还允许出现下列时序算子:$○:下一时刻算子,□:必然算子,◇:终于算子,$U:直到算子,$W:除非算子,此外,还定义了一些过去时时序算子.

一个具有如下形式的等式:　　$○V=exp　　(1)

称为状态转换等式,(1)式表明变量V在下一时刻的值等于表达式exp在本时刻的值.如(1)式左边变量恒为一特殊的系统变量"LB"(即执行标号),右边exp被一个符号y代替,则(1)式变成:

$$\$○LB=y \tag{2}$$

其含义等同于一般goto语句,即"goto y".

下列形式的任一蕴含公式称为条件元(c.e.):

$$LB=y∧R=>@(Q∧LB=Z) \tag{3}$$

及

$$LB=y∧R=>\$○(V1,...,Vk)=(e1,...,ek)∧\$○LB=Z \tag{4}$$

其中"@"表示$○或◇,"R","Q"为一阶逻辑公式,分别称为c.e.的条件部分与动作部分;

一个合式公式如具有以下形式,则称之为单元:

□[A1;...;An]

WHERE B1∧...∧Bm (5)

此处";"是合取词"∧"的一个记法,每一 Ai(i=1,...,n)为一条件元,WHERE 列出的合取式为该单元的约束部分.(5)式构成 XYZ/E 程序的基本单元.

(2)程序框架

过程

过程说明具有如下的形式:

```
ProcDeclaration::= ProcName [ParameterDeclPart]==
                        [[LocalVarDeclPart ]
                         [ProcDeclarationPart ]
                         ProBody]
                         [WherePart ]                    (6)
```

符号[ ]表示该部分可为空,ProBody 是程序的核心单元,XYZ/PME 的控制结构形式为 XYZ/BE 表示的单元构成,表示为%ALG.

进程

进程说明类似于过程(符号为%PROS),但进程定义包括通道参量,它表示该进程将通过什么通道与其它进程相连接.形式如下:

```
%CHN CHName(obj:Nm, * ):ChType;
→
...
→'
ChName( * , obj: Nm):ChType                              (7)
```

obj 表示进程实例文本的名字或指针名字,(obj, *)与(* .obj)分别表示该通道是由其它进程通向本进程传送信息还是由本进程向其它进程传送信息2个不同方向.

通讯通过输入命令与输出命令进行,其形式分别如下:

$$LB=y \wedge R=> \$\bigcirc ChNm?x \wedge \$\bigcirc LB=z \quad (8)$$

$$LB=y \wedge R=> \$\bigcirc ChNm!w \wedge \$\bigcirc LB=z \quad (9)$$

利用(8)、(9)式同时可以很好地表示2个进程之间进行通讯的约束条件,即角色之间并行活动的约束条件.

包块

每一包块(Package)是由一组操作围绕数据结构封装而成的模块,在 XYZ/PME 中,包块表示软件过程中的一个过程对象.说明如下:

```
PackageDecl::=%PACK[PackageName:
                    [TypeDeclPart ]
                    [ParentDeclPart ]
                     VarDeclaPart;
                     OperationDeclPart]
```

[WherePart]　　(10)

此处 parentDeclaPart 表示当前被定义包块的父亲,"根"包块无父亲. VarDeclPart 定义包块的一些共享属性,如永久性、建立者等. XYZ/PME 中的包块具有可分解性,一个包块可分解为若干个子包块,子包块的数据结构是其父亲的子集,子包块的操作可以是其父亲的操作,也可是新定义的操作.

Agent

Agent 包括2部分,在 XYZ/PME 中用于描述一个角色. 定义如下:

AgentDeclPart::=%AGT[Agent,...,Agent]　　(11)

Agent::=[Packages,Process]　　(12)

每一个 Agent 中的进程部分定义了该 Agent 的行为,进程负责和其它 Agent 的通讯,通讯利用通道进行;包块部分封装了数据和操作,但对于同一 Agent 内的进程而言这些数据和操作等同于局部变量和过程;Agent 的属性描述通过包块表示,如对角色的文字串说明等. Agent结构将包块的静态语义和过程的动态语义比较完美地结合在一起.

XYZ/PME 程序的形式

一个 XYZ/PME 的程序结构为:

ProgramName::= %PMPROG ProgramName ==[[ProsDeclPart;]
[AgentDeclPart;]
Probody]
[WherePart]　　(13)

一个 XYZ/PME 的过程模型即为上述结构的 XYZ/PME 程序.

3.3 XYZ/PME 的过程建模特征

XYZ/E 既是一个逻辑系统,又是一个程序设计语言,其子语言 XYZ/PME 较好地满足了本文第2节中所提出的几点要求;从上述简介看来,模块性是显然的;其次,它很好地支持了抽象性、逐步求精,这一点是其它过程建模语言难以达到的. 例如,对 ISPW－6过程实例[10],一个软件修改过程由一个角色(项目组)进行,因此通过先描述项目组(ProjectTeam)可得出该过程的高层抽象描述(第0级):

```
%AGT[ProjectTeam;
    ...
    %PACK[aSoftWare;
      ...
      %PROC[/* operations */
        workonchange(%INP f1: FILE;IOP f2: FILE;
                     %IOP softflag: FLAG )==[
        ALG[LB0=START_workonchange(∧f1/=NULL)∧(f2==f1)
            ∧(softflag == notok) /* pre condition */
            ⇒◇(f2/=f1)∧(softflag==ok)/* post condition */
        ]∧$○LB0=RETURN]]]
        %PROS[
        SoftChange(CHN ch－from－CCB(CCB:Nm,*): STRING)==[
          %LOC[oldversion, newversion: FILE;softflag: FLAG; notification: STRING];
          %ALG[/* behavior description */
            LB0=START_ProjectTeam0⇒$○((oldversion=NULL)∧(softflag=notok))∧$○LB0=l0;
            LB0=l0⇒$○ch－from－CCB?notification∧$○LB0=l1;
            LB0=l1∧(notification==="request")⇒$○READ(oldversion)
```

```
                Λ workonchange(INP f1|oldversion,IOP f2|newversion,
                %IOP softflag|softflag) Λ $◯LB0=l0;
            LB0=l1 Λ (notification ==“cancel”)⇒ $◯LB0 = EXIT]
        ];
]/* end of Agent ProjectTeam
```

图2 角色 ProjectTeam 的 XYZ/PME 表示

角色的第0级描述表示角色 ProjectTeam 接受 CCB 的请求，修改一个软件. 之后，包块 aSoftware 的定义可以进一步求精，如对操作 workonchange 进一步加以定义，直至给出算法描述；角色 ProjectTeam 可分解为项目管理员、设计工程师、质量工程师，而加以进一步定义.

每一个 Agent 均可独立地在分布式环境的节点上运行，满足分布性要求.

XYZ/PME 对多个角色的并发行为的描述是十分自然的，通过如下并发语句表示：

$$LB=y \Lambda R \Rightarrow ||[pr1(par1);...;prk(park)] \tag{14}$$

此处，pri 为实例化的进程，pari 为参数，(14)式表示 k 个角色的并发行为.

作为一个基于时序逻辑的语言，为表示非确定性和动态性提供了自然的支持，如利用选择语句即可表示：

$$LB=y \Lambda R \Rightarrow !![Con1 \triangleright Act1,...,Conq \triangleright Actq] \tag{15}$$

其语义等于：

$$LB=y \Lambda R \Rightarrow \$\bigcirc[Con1 \Lambda Act1 \$v'(...\$v'(Conq \Lambda Actq] \tag{16}$$

$\$v'$为不可兼析取算子.

XYZ/PME 同时是一面向对象的语言，其 Agent 机制为模型化角色(即一种对象)的静态、动态行为和相互关系提供了合适的形式的语义和语法定义，整个过程模型即是依据这些单个角色的定义和角色之间的关系定义的(见图3)，这是面向对象的形式化建模必须解决而往往未解决好的关键问题.

上述 ISPW－6例子的进程模型是一个可执行的 XYZ/PME 程序，其形式为：

```
%PMPROG theProcess ==[/* 可执行的过程模型 */
  ……
 LB2=l0⇒ $◯projectmanager1==ProjectManager{1}
          Λ $◯designer1==Designer{1}
          Λ $◯programmer1==Programmer{1}
          Λ $◯reviewer1==Reviewer{1}
          Λ $◯tester1==Tester{1}) Λ $◯LB2=l1;
 LB2=l1⇒||[projectmanager(…);designer(…);programmer(…);reviewer(…);tester(…)]
  ]
```

图3 ISPW－6过程模型

4 结束语

XYZ/PME 是基于时序逻辑的、严格形式化的，可以精确地描述过程，这是对过程进行理性的分析、评估和改进的基础；它又是一个程序设计语言，满足本文所提出的过程建模语言的各种要求，所表示的不同抽象级的过程模型均是可执行的，通过执行较高抽象级的模型，可以较早发现并解决问题，以获得好的过程模型. 此外，可以在统一的逻辑框架内验证不同抽象级的进程模型的一致性，也可通过执行来检查.

目前 XYZ/PME 及有关图形工具均已在 SUN 工作站上实现，下一步的工作是结合对

过程实施(Enact)的研究,使 XYZ 环境明确支持软件过程.

参考文献

1 Curtis B, Kellner M I, Over J. Process modeling. Communications of ACM, September 1992,**35**(**9**).

2 Conradi R, Fernstrom C, Fuggetta A *et al*. Towards a reference framework for process concepts. In: Proceedings EWSPT'92, volume 635 of Lecture Notes in Computer Science, Springer Verlag, 1992.

3 Osterweiol L. Software processes are softwares too. In: Proceedings of the 9th International Conference on Software Engineering, Monterey, California, March 1987.

4 Oquendo F, Zucker J, Griffiths P. The MASP approach to software process description instantation and enaction. First European Workshop on Software Process Modeling, Milan, Italy, May 1991.

5 Katayama T. A hierarchical and functional software process description and its enaction. Proceedings 11th, ICSE, Pittsburgh PA, 1989.

6 Barghuti N, Kaiser G. Scaling up rule—based software development environments. Proc. of the 3rd. European Software Engineering Conference, ESEC'91, Milan, Italy, October 1991.

7 Bandinelli S, Fuggetta A, Ghezzi C. Software processes as real time systems: a case study using high—level Petri nets. Proc. of International Phoenix Conference on Computers and Communications, Arizona, April 1992.

8 Tang C S. A temporal logic language oriented toward software engineering——an introduction to XYZ system(I). Chinese Journal of Advanced Software Research, 1994,1(1):1～29.

9 Tang C S. Introduction to the XYZ system. International Workshop on Logic and Software Engineering, Beijing, China, August 1995.

10 Kellner M, Feiler P, Finkelstein A *et al*. ISPW—6 software process example. Proceedings of the First International Conference on Software Process, Washington: IEEE Computer Society Press, 1991.

A STUDY ON SOFTWARE MODELING LANGUAGES

Liu Junfei　Tang Zhisong

(*The Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　This paper discusses the fundamental concepts of software process modeling and the basic requirements on software process modeling languages, introduces briefly several typical software process modeling languages, gives a formal process modeling language XYZ/PME . XYZ/PME supports the process modeling approach using a stepwise refinement technique, can represent process models of different abstract levels in a unified formal framework.

Key words　Software process, software process modeling, software process modeling language, temporal logic, programming language.