基于递阶过程模式复用的软件项目过程控制

曹 健,张申生,李明禄
(上海交通大学计算机科学与工程系,上海市华山路 1954 号,上海 200030)

摘 要: 通过对过程模式的复用能够保证软件开发过程的规范性. 由于各个项目都是独特的,因此不存在一个统一的过程模式适合于所有的项目. 本文提出了递阶过程模式动态复用的思想. 为了便于重用,过程模式的描述信息建立在本体基础上,并采用了基于事件的软件过程建模方法. 文中讨论了过程复合方法和基于软件结构实现递阶过程模式动态复用的技术,并介绍了实现该技术的系统结构.

关键词: 软件项目过程控制; 过程模式; 递阶过程模式
中图分类号: TP319 **文献标识码**: A **文章编号**: 0372-2112 (2003) 12A-2059-04

Software Project Process Control Based on Hierarchical Process Patterns Reuse

CAO Jian, ZHANG Shen-sheng, LI Ming-lu
(*Department of Computer Science and Technology, Shanghai Jiaotong University, Shanghai* 200030, *China*)

Abstract: Process patterns reuse is an effective way to improve the software process. Since each software project is unique, it is not possible to define a detailed and general process model for all software projects. In the paper, a hierarchy process pattern dynamical reuse method was proposed. Ontology based pattern description and event based process model were applied to facilitate reuse. It also discussed process composition method and an automatic process pattern reuse method according to the software product structure. The system structure that can support this concept was also introduced.

Key words: software process control; process pattern; hierarchical process pattern

1 引言

由于软件开发固有的一些特性,它的工业化程度与其他行业相比还有不小的差距[1]. 软件开发过程的质量与软件本身的质量有直接的联系. 采用信息系统来控制软件项目过程将是一条有效的途径. 许多研究者已经认识到了提供此类信息系统的意义,并把这类系统称为以过程为中心的软件工程环境 PSEE(Process centered Software Engineering Environment)[2]. PSEE 通过定义软件开发过程模型并通过对过程模型的驱动和执行将开发者、软件数据对象、软件工具等有机集成起来. 目前,许多研究机构推出了 PSEE 的原型,如 EPOS, SPADE-1, 也有少量的商品化系统如 Process Weaver 和 LEU[3,4].

在 PSEE 中,在过程执行前必须先定义过程模型. 但是由于软件开发是一个复杂的、具有不确定性的过程,过程模型在执行时将发生动态演化. 如何在对过程进行严格控制的同时又保证过程执行的灵活性具有相当大的难度,目前的 PSEE 环境对此问题的解决并不满意. 本文提出了一种基于递阶过程模式动态复用技术,为软件开发项目过程控制提供了一种有效途径,提高了 PSEE 的适应能力.

2 软件开发过程与递阶过程模式

在软件开发过程中得到了成功验证的方法或者操作的序列被称为过程模式[5].

模式是针对某一场景和系统中的一般性问题的可重复的解决方案. 模式的概念在软件工程方面首先被用于设计问题,并被称为设计模式[6],此后,研究者还将模式的概念推广到分析问题[7]和软件体系结构[8]等方面. 近年来,研究者提出了过程模式的概念并进行了深入研究,如 Ambler 对过程模式进行了分类[5,9], Störrle 提出了采用 UML 描述过程模式的方法[10], Gnatz 等人提出了具有模块化结构的过程模式框架[11].

尽管过程模式可以重用在不同的软件项目中,但是各个软件项目显然都是独特的. 这种独特性意味着各个项目的实施过程将各不相同. 过程模式的重用和软件项目的动态性、不确定性从表面上看似乎构成了一对矛盾:过程模式的重用意味着需要采用统一的过程模型,而软件项目的动态性和不确定性又意味着各个软件项目的过程将各不相同. 在本文中,我们将通过引入递阶过程模式来解决该矛盾. 通过过程层次可以反映对过程的不同程度的抽象,我们可以用上层的过程模

收稿日期:2003-08-30;修回日期:2003-12-20
基金项目:国家 863 项目"基于 Agent 与工作流技术的面向实施的企业业务过程建模与评价环境"(No. 2001AA415310)

式表达宏观的软件开发组织方式,而用细节层次的过程模式表达过程的细节.通过不同层次的过程模式的复用既能够保证过程开发的规范性,又能够适应过程的动态性和不确定性的特点.本文正是建立在此概念的基础上.

3 递阶过程模式的表达方法

3.1 递阶过程模式的定义

为了提高过程模式描述的规范性,我们采用本体来描述一个过程模式.

定义 1 本体可以表示成三元组 (L, A, S),其中 L 是一种语言,A 是非空集合,表示某一领域,其元素 $a \in A$ 是该领域的客观存在(实体或关系),而 S 是一结构,它使得 L 中的词汇 v 在领域 A 内有明确的意义,即 $S(v) \rightarrow a, a \in A$.

针对表达软件过程模式的需要,我们引入三类本体,即过程本体、软件对象本体和软件开发技术本体.关于这三类本体的具体内容,我们在此文中不再详细阐述.

定义 2 过程模式可以表达为 $P_p = \langle D, W, C_{on} \rangle$,其中 D 为过程模式的一般信息,包括模式的名字,创建者等,W 为该模式对应的过程模型,C_{on} 为过程模式适合的场景(Context).

过程模式的场景表达了该过程模式适用的场合,它指导用户确定在何种情况下重用何种模式.场景的描述是基于本体方法的.

定义 3 过程模式的场景定义为 $C_{on} = \langle O_p, O_o, O_t \rangle$,其中 O_p 为采用过程本体描述的过程特性,O_o 为采用对象本体描述的对象特性,O_t 为采用技术本体表达的技术特性.

例如某一过程模式的场景可以定义为 $c = \langle [\text{Project}(LOC < 2000, Time < 3\ \text{month})], [\text{Type}(\text{“Database Application”}), \text{Domain}(\text{“Well Known”})], [\text{Tools}(\text{“Visual Basic”}), \text{Database}(\text{“SQL Server”})] \rangle$.

场景的定义方便了过程模式的选用.在项目启动前或者项目进行过程中,项目管理者可以对实际项目的场景和过程模式的场景进行比较,在此基础上选择合适的过程模式,也可以依据算法在计算机的帮助下来进行过程模式的选取.

过程本体中的领域元素之间存在一个递阶层次结构关系,我们将该关系表示为 Sub,它是一个二元关系,例如 Sub(“Project”,“Inception”)就表示项目过程下包含项目初始化阶段.该关系具有传递性.依据过程本体的递阶关系就可以形成过程模式的递阶关系.

3.2 事件驱动的软件过程模型

在过程模式的定义中,W 为过程模型.本节中我们提出一种基于事件的软件过程模型,以适应过程模式动态复用的目的.

定义 4 事件是一个函数,它将时间映射为布尔值,它可以表示为:

$$E: T \rightarrow \{\text{True}, \text{False}\}$$

$$E(t) = \begin{cases} \text{True}, & \text{if } E \text{ 类型的事件在 } t \text{ 发生} \\ \text{False}, & \text{其余情况} \end{cases}$$

t 称为事件 E 的发生时间.

事件可以分成原子事件和复合事件.原子事件是那些可以被系统直接探测到的事件.原子事件可以分为绝对时间事件、活动状态变化事件和操作事件.原子事件可以进行复合,我们定义两个事件复合算子:

(1) AND(并):$\text{AND}(e_1, e_2)$ 表示 e_1 和 e_2 均发生

(2) OR(或):$\text{OR}(e_1, e_2)$ 表示 e_1 和 e_2 至少有一个发生

我们将通过原子事件和复合算子形成的表达式代表的事件称为复合事件,表达式称为复合事件表达式.

定义 5 软件过程模型可以表达为 $W = (E, A, C, D, R, TC, RE, CF, DF)$,其中 E 是事件集合,A 是活动集合,C 是连接节点集合,D 为数据集合,R 为规则集合,$TC \in C \rightarrow \{\langle \text{AND}, \text{AND} \rangle, \langle \text{AND}, \text{XOR} \rangle, \langle \text{OR}, \text{AND} \rangle\}$ 为一个函数,它将每一个连接与其类型相映射,即 $\forall c, T(c) \in \{\langle \text{AND}, \text{AND} \rangle, \langle \text{AND}, \text{XOR} \rangle, \langle \text{OR}, \text{AND} \rangle\}, c \in C$,其中类型表达式 $\langle L_1, L_2 \rangle$ 中 L_1 表示连接节点的输入端逻辑,L_2 表示连接节点的输出端逻辑;RE 为一个函数,它将规则满足映射为事件,即 $RE(R) \in E$;$CF = \{E \times A \times C\} \vee \{E \times C \times A\} \vee \{E \times C \times C\}$ 为控制流,CF 表达式中的 E 称为控制流驱动事件,$DF = \{E \times A \times A \times D\}$ 为数据流,DF 表达式中的 E 称为数据流驱动事件.

定义 6 过程模型中存在两个特殊活动:开始活动 A_s 和结束活动 A_e.A_s 活动只有输出,没有输入,而 A_e 只有输入,没有输出.从 A_s 出发,存在从 A_s 开始的有向路径连接到过程模型中的任一节点,从任一节点出发,也必须存在从该节点开始的有向路径连接到 A_e.

图 1 为软件过程模型的图示表示,图中为模块编码过程.

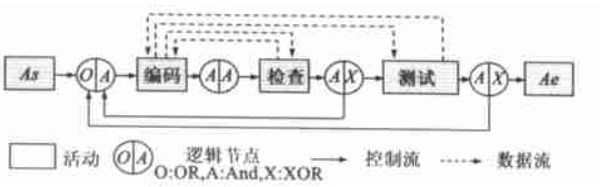

图 1 软件过程模型图形示例

上述软件过程模型中的活动可以分解成另一个软件过程模型.被分解的活动称为复合活动,不再被分解的活动称为原子活动.通过分解关系可以形成一个递阶的过程结构.假定活动为 a,我们记 $\uparrow(a)$ 为 a 的父活动,$\downarrow(a)$ 为 a 的子活动集合,$\rightarrow(a)$ 为与 a 具有同样父活动的活动集合,即 $\rightarrow(a) = \downarrow(\uparrow(a))$.$\uparrow\uparrow(a)$ 称为 a 的祖先活动,祖先关系具有传递性.

定义 7 设输入某连接节点 c 上的控制流为 $\{e_1 \times a_1 \times c\} \vee \{e_2 \times a_2 \times c\} \vee \cdots \vee \{e_m \times a_m \times c\}$,称 e_i 为节点 c 的一个输入事件,控制流上的事件由节点的输入端逻辑函数(设为 L_1)生成的事件称为该节点的驱动事件,即如果连接节点为 $\langle \text{AND}, \text{AND} \rangle$ 或 $\langle \text{AND}, \text{XOR} \rangle$,则该节点驱动事件为 $\text{AND}(e_1, e_2, \cdots, e_m)$,如果连接节点为 $\langle \text{OR}, \text{AND} \rangle$,则该节点驱动事件为 $\text{OR}(e_1, e_2, \cdots, e_m)$.如果该节点为 $\langle \text{AND}, \text{AND} \rangle$ 或 $\langle \text{OR}, \text{AND} \rangle$,从节点上引出的控制流的驱动事件直接为该节点的驱动事件,如果该节点为 $\langle \text{AND}, \text{XOR} \rangle$,则从节点引出的控制流的驱动事件为一规则满足事件.对于活动而言,其驱动事件定义为所有输入控制流的驱动事件取 OR 运算得到的复合事件.

4　基于递阶过程模式动态复用的过程演化技术

项目管理者在项目开始前,首先选定的是一个项目级的过程模式,通过对它的复用可以定义整个项目的过程组织方式,从而构成了项目执行的基准.

在项目开始后,根据需要,项目管理者对项目过程模型进行细化.细化的方法是选定某一过程模式,通过复用后得到子过程,并将该子过程插入到项目过程模型中去.在插入时,还需要定义实例化的子过程与已经存在的项目过程通过控制流、数据流相连接的方式,如图 2 所示.

过程模式复合中最为关键的是将实例化后的过程插入到软件项目过程模型中.为了插入到模型中,从纵向看,需要定义该子过程插入到哪一个父活动中,从横向看,需要定义其驱动条件.纵向关系的定义比较简单,只需要指定某父活动即可.由于活动的启动条件可以与多个活动逻辑相关,因而横向关系的定义比较复杂.

子过程在插入到软件项目过程模型中时,以一复合活动代表,假定该活动为 a. a 插入的父活动为 f, $\downarrow(f)$ 为父活动 f 的子活动集合,对于 $\forall b_i \in \downarrow(f)$,其状态变化事件为活动开始事件 Begin$Of(b_i)$ 和活动结束事件 End$Of(b_i)$. 取 $\downarrow(f)$ 中所有活动的状态变化事件,从而得到事件集合 E.

在 E 中选取不同活动的状态变化事件后,通过事件复合算子复合得到 a 的启动事件表达式,设为 p_s. 通过对 p_s 的解析可以将 a 通过连接节点和控制流连接到项目过程模型中.

在 p_s 中,每一个由函数构成的表达式的计算有其优先级,优先级可以通过表达式中的括号来计算,内部的括号比外部的括号具有更高的优先级.优先级的计算方法在此不再赘述.假定最高优先级为 $O_{\max}$,最低优先级为 1.下面的算法实现将过程模式生成的子过程通过逻辑节点和控制流连接到项目过程中.

算法 1:

(1) 在 f 的子过程模型中插入活动 a;

(2) $i = O_{\max}$;

(3) 取优先级为 i 的函数运算,设事件集合为 FS,假定逻辑函数为 OP(由括号括起的表达式对应一个连接节点):

如果 OP = AND,插入〈AND,AND〉型连接节点 N;

如果 OP = OR,插入〈OR,AND〉型连接节点 N;

(4) 从 FS 中所有活动事件对应的活动或复合事件对应节点连控制流到连接节点 N,控制流上的事件与事件表达式上的事件一致,N 将代表 P_S 中通过优先级 i 的运算形成的元素;

(5) $i = i - 1$,如果 $i > 0$,则转(3),否则转(6);

(6) 添加两个连接节点分别为 c_1:〈OR,AND〉和 c_2:〈AND,AND〉;

(7) 将直接引入到 Ae 的所有控制流改成连到 c_1:〈OR,AND〉上,并从该节点引控制流到 c_2:〈AND,AND〉;

(8) 从 a 引控制流到 C_2:〈AND,AND〉,驱动事件取活动结束事件;

(9) 从 c_2:〈AND,AND〉引控制流到活动 A_e;

(10) 结束.

假定驱动事件表达式为:p_s = AND(OR(EndOf(a_1), EndOf(a_2)), EndOf(a_3)). 应用上述算法分析后的结果如图 3 所示,带阴影的节点为插入的节点.

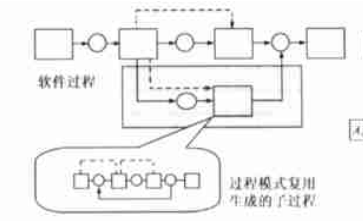

图 2　软件过程动态定义示意图

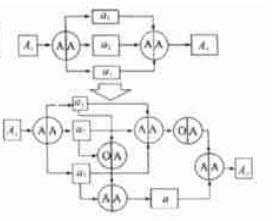

图 3　过程复合方法示意图

5　基于软件结构映射的递阶过程模式复用

软件开发的后续阶段,特别是编码和测试,与软件的结构密切相关.因此,在项目进行中,软件结构是软件开发过程规划的依据.

定义 8　软件对象可以定义为 $s_o = \langle K_{so}, P_{so}, M_{so} \rangle$,其中 K_{do} 为软件对象类型,P_{so} 为对象属性集合,M_{so} 为对象方法,用以访问和操纵对象.软件对象之间的构成关系可以用树型结构来表示.

定义 9　软件对象结构树定义为 $SO_t = \langle SO, s_{or}, SR_{sub} \rangle$,其中 SO 为软件对象集合,s_{or} 为软件对象结构树的根对象,SR_{sub} 为 SO 中各个对象之间的父子关系集合.

对于结构树中的节点,我们可以选定其适用的过程模式,而在过程模型中,通过对该过程模式的复用可以得到一个子过程,在项目过程模型中,该子过程可以用一个父活动来代表.因而,软件对象结构树可以映射成一个过程结构树.

定义 10　过程结构树定义为 $PS_t = \langle PS, p_r, PR_{sub}, PR_{ord} \rangle$,其中 PS 为对应过程集合,p_r 为根过程,PR_{sub} 为过程的父子关系集合.

与软件对象结构树不同,软件过程结构树中添加了一个定义 PR_{ord},该定义通过串行函数 SEQ() 和并行函数 PAL() 规定了某一父过程下的子过程的执行依赖顺序.

通过函数表达式将一组活动复合起来的活动表达式,我们称之为逻辑活动,作为函数表达式中的自变量、构成逻辑活动的活动称为该逻辑活动的组成活动.

下面提供的算法依据过程结构对象实现过程的动态演化:

算法 2:

设 GetSon$Of(d_o)$ 为取过程结构树中节点 d_o 的子节点函数,DO_s 为先入先出过程结构节点栈,DO_s 为空:

(1) 将根节点压入 DO_s;

(2) 如果 DO_s 为空,则结束,否则取出 DO_s 中过程,设为 d_o;

(3) 如果 d_o 为根过程,则取 d_o 适用的某一过程模式,生成相应的过程,否则从过程中选取对应于 d_o 的活动,选取某一过程模式,生成子过程,记为 W_{do};

(4) 将 GetSon$Of(d_o)$ 中包含的节点压入 DO_s;

(5) 如果 d_o 不再包括子节点,则转(2),否则转下一步;

(6) 设 P_S 为 d_o 下的子节点对应的过程表达式,对 W_{do} 中的某一活动进行分解,设分解的子过程为 w_s,生成 d_o 下对应的活动,并添加逻辑开始活动和逻辑结束活动;

(7) $i = O_{\max}$;

(8) 取优先级为 i 的函数运算,设集合为 FS,集合中元素个数为 j

(9) 取出第 j 个函数运算,其中包含的组成活动集合为 AS,集合中元素顺序与函数中自变量顺序相同,设逻辑函数为 OP;

(10) 如果 OP = SEQ,在 w_s 中插入连接节点〈AND,AND〉,从该节点引控制流到 AS 中第一个活动,插入〈AND,AND〉节点,从该活动引控制流到该节点,并选该活动的结束事件为控制流驱动事件,从该节点引控制流到第二个活动,……,重复上述过程,直至插入最后一个〈AND,AND〉节点,并从最后一个活动引控制流到该节点

(11) 如果 OP = PAL,插入连接节点〈AND,AND〉,从该节点到所有 AS 中活动引控制流,插入最后一个〈AND,AND〉节点,从 AS 中所有活动引控制流到该节点,各控制流的驱动事件为对应活动的结束事件.

(12) $j = j - 1$,如果 $j = 0$,转下一步,否则转(9);

(13) $i = i - 1$,如果 $i = 0$,从开始活动引控制流到整个活动,从活动引控制流到结束活动.否则返回(8);

(14) 返回(2).

应用上述算法对 PR_{ord} = SEQ(PAL(SEQ(a_{do1}, a_{do2}), a_{do3}) a_{do4})的例子进行规划后的结果如图 4 所示.

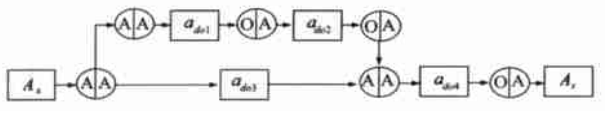

图 4 过程规划示例

6 系统实现与实例

图 5 为支持上述技术的系统结构.在前台,系统提供建模工具,管理工具和任务工具.建模工具使过程工程师能够设计过程模式;管理工具给项目管理者提供了项目初始化,项目过程模型复合,项目监控的功能;任务工具给软件项目中的工作者提供了接受任务、提交任务的接口.

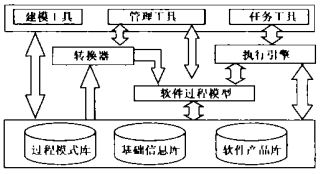

图 5 系统结构

管理工具通过后台的转换器将过程模式实例化为子过程,进而复合成项目过程模型,同时管理工具也可以监控项目的进度、各个人员和资源的负荷和成本消耗.执行引擎通过解析项目过程模型来分发任务、动态判断事件的发生情况,进而推进整个过程的执行.

在数据层,过程模式库中保存软件企业总结出来的典型的规范化的过程模型,基础信息库中保存了软件企业的人员、资源信息,软件产品库中保存软件开发过程中生成的所有数据对象.

根据本文提出的技术和系统结构,我们开发了软件产品开发管理系统 SPDM,并在几个软件企业正在进行应用.该系统前台具有 Web 风格,并与 Exchange Server 2000 集成,能够支持项目管理者动态确定过程的安排.

7 结论与展望

从某种意义上看,每一个软件开发项目都是独特的,而在软件开发过程中,每个过程又有一定的不确定性,因而无法事先统一定义一个软件过程模型.过程模式是对软件开发过程典型环节的总结,它的作用体现为一种过程复用.一个软件开发项目过程将由多个典型的流程环节构成.根据该思路,本文针对提供软件项目过程运行支持的目的,提出了一种递级过程模式的概念,在此基础上研究了递级过程模式动态复合技术,以实现项目过程模型的动态定义.

参考文献:

[1] Roger S Pressman. 软件工程——实践者的研究方法(第 5 版)[M]. 梅宏,等,译,北京:机械工业出版社,2002.8.

[2] Vincenzo Ambriola, et al. Assessing process-centered software engineering environments [J]. ACM Transactions on Software Engineering and Methodology, 1997, 6(3): 283 - 328.

[3] R Conradi, et al. EPOS: Object-oriented cooperative process modeling [A]. B Nuseibeh, et al. editors, Software Process Modeling and Technology [M]. John Wiley and Sons, 1994. 33 - 70.

[4] Sergio Bandinelli, Elisabetta Di Nitto, et al. Supporting cooperation in the SPADE-1 environment [J]. IEEE Transactions on Software Engineering, 1996, 22(12): 841 - 865.

[5] S W Ambler. Process Patterns: Building Large-Scale Systems Using Object Technology [M]. Cambridge University Press, 1998.

[6] E Gamma, R Helm, R Johnson, J Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software [M]. Addison-Wesley, October 1994.

[7] M Fowler. Analysis Patterns: Reusable Object Models [M]. Addison Wesley, October 1997.

[8] M Shaw, D Garlan. Software Architecture: Perspectives on an Emerging Discipline [M]. Prentice Hall, 1996.

[9] S W Ambler. More Process Patterns: Delivering Large-Scale Systems Using Object Technology [M]. Cambridge University Press, 1999.

[10] H Störrle. Describing process patterns with UML [A]. Lecture Notes in Computer Science [M]. 2077, Springer-Verlag, 2001. 0173 - 0181.

[11] M Gnatz, F Marschall, G Popp, A Rausch, W Schwerin. Towards a living software development process based on process patterns [A]. Lecture Notes in Computer Science [M]. 2077, Springer-Verlag, 2001. 182 - 202.

作者简介:

曹 健 男,1972 年 9 月生于江苏宜宾,博士,副教授,现为上海交通大学计算机系教师,主要研究方向过程工程,企业建模,服务计算和软件工程,参与和承担多项国家自然科学基金和国家 863 计划项目,已经发表论文 60 余篇.

张申生 男,1951 年 2 月生于上海,博士,教授,博士生导师,现为上海交通大学计算机系教师,目前担任上海交通大学电子信息与电气工程学院副院长,主要研究方向企业集成技术,知识管理,虚拟现实和软件工程,主持多项国家自然科学基金和国家 896 计划项目,发表论文 100 余篇.