软件过程的 JMOSP 模型*

张 然 张育超 叶云文

(复旦大学计算机科学系,上海 200433)

摘要 由于软件过程模型在提高软件生产力方面具有重要的意义,它已经成为软件工程领域内的一个研究热点.本文在简单地分析了当前几种不同的软件过程模型的缺陷后,提出了软件过程的 JMOSP 模型.

关键词 软件过程模型,软件过程,JSD.

近年来,软件工程研究中的一个热点就是关于软件过程的研究[1],尽管在这方面已取得一些进展,但软件过程在提高软件生产力方面所具有的重要意义仍旧没有得到应有的体现,一个重要的障碍在于缺乏合适的软件过程模型[2,3].

软件过程本身所具有的高度复杂性使得很难有一种软件过程模型能完全满足软件过程模型的种种功用性.但是,我们认为一个合适的软件过程模型应尽可能具有下述特征:

(1)软件过程模型应该能够体现软件开发过程的进化性.在软件开发中,有些活动可能被修改,软件过程模型应能指出这一修正所涉及的范围.

(2)软件过程模型应该能够体现软件过程的并行性.

(3)软件过程模型应该能够体现软件过程的动态变化性,即软件过程本身的状态随着开发活动的进展变化.

(4)软件过程模型应该具有易理解性.

(5)软件过程模型应该具有良好的形式化基础.

(6)软件过程模型应具有良好的可适用性,即可适用于不同的软件开发项目.

本文在简单地分析评价了目前已有的多种软件过程模型以后,提出了软件过程模型 JMOSP(JSD Model Of Software Process).软件过程模型 JMOSP 以 JSD 方法为基础,具有直观性、灵活性、可复用性、并行性等优点.

1 软件过程模型

如今已存在一些描述软件过程的模型,由于所考虑的重点不一样,因此它们从不同的角度去刻划软件过程的特征.当前的软件过程模型大致分为以下几类:

* 本文 1992-04-03 收到,1992-07-16 定稿

作者张然,57 岁,教授,主要研究领域为软件工程.叶云文,27 岁,助教,主要研究领域为自动程序设计,形式化方法、软件过程.张育超,38 岁,讲师,主要研究领域为软件工程.

本文通讯联系人:张然,上海 200433,复旦大学计算机科学系

(1)任务型.这种类型的软件过程模型将软件过程分为若干有序步骤,描述每一步骤的输入是什么,做什么事情及所产生的输出是什么.关于这类模型的一个典型例子就是"瀑布模型".它们的一个致命弱点就是不能描述步骤之间的逆向关系和不同步骤之间的并行关系.

(2)实体关系型.这种类型的软件过程模型非常类似于 ER 模型,它们主要考虑的是软件过程中实体(产品,资源)之间的关系.在文献[4]中所介绍的结构化模型即属此类模型.它们简单,并具有较好的稳定性,即一个实体关系模型能适用于相当一类软件过程.但作为软件过程模型,仅描述实体及它们之间的关系是不够的,更重要的是要描述它们在软件过程中的动态变化过程.

(3)Petri 网型.这类软件过程模型强调软件过程的动态性,其典型代表是 DesignNet 模型[5].尽管 Petri 网型的模型较好地考虑了任务的激活条件、任务的执行顺序及由于任务的执行而产生的实体之间的相互转换等问题,但它们忽略了任务对实体内部状态所产生的影响;同时,它们是用一个完整的具有较强相关性的网来描述软件过程,对软件过程的任何改动都可能对这类模型产生全局性的影响,因此 Petri 网型的模型通常不利于复用.

为了弥补上述各类软件模型的缺陷,本文将提出一种新的软件过程模型——JMOSP,它的基础是 Jackson 系统开发模型[6].JMOSP 模型通过对软件开发活动中各个实体的生命周期以及实体之间关系的生命周期的描述,刻划出整个软件开发过程.

2　JMOSP 模型的背景及基本概念

JSD 方法的主要目的是为了帮助软件人员建立现实世界的模型,并以模型为基础系统化地开发软件.JSD 模型的优点在于:首先,JSD 模型是以实体为中心的,相对于以功能为中心的模型而言,JSD 模型具有更好的稳定性,因而也具有更好的适用性;其次,JSD 模型采用分散方式进行模拟,这使得它具有较好的可扩充性;第三,JSD 模型以进程作为模拟单位,每一个 JSD 进程是顺序的,但不同的 JSD 进程是并行的,这一特征对描述软件过程的并行性具有特别重要的意义.

但是,如果要用 JSD 模型来描述软件过程,它存在一个严重的缺陷.实体之间的关系在软件过程中有极其重要的意义,但是 JSD 却不能对这种关系进行刻画.我们在这里提到的实体之间的关系不仅仅指这些实体的静态相关性,更主要的是指实体之间的动态相关性,即一个实体被修正以后将以怎样的方式和范围影响其他的实体.JMOSP 模型克服了 JSD 模型的这种缺点.在 JMOSP 模型中,关系被提高到与实体同等重要的地位,并以同样的方式描述和模拟.

JMOSP 和 JSD 的另一个重要区别是在 JMOSP 模型中,进程是可参数化的,而且进程中的事件往往与一个动作例程相关联.

在 JMOSP 模型中,实体既可以表示一个在软件过程中产生的软件产品(如设计规格说明书、源代码等),也可以表示软件过程中所使用的资源(如机器、显示器等),还可以表示软件过程中的一个角色(程序员、管理员等).实体所拥有的特征是由定义在它上面的一组事件以及这些事件的发生顺序确定的.一个实体可以经历生成、修改、删除等活动,它是 JMOSP

模型中的一个动态成员.

在JMOSP模型中,事件表示软件过程中可能发生的一个活动.事件不能独立存在,它总是相关于一个或同时相关于几个实体或关系.事件的发生将导致实体或关系的变化,它总是在某一时刻发生,而不是在某一时期发生.而这些先后不断发生的事件维持了一个实体或关系的生命周期.每一事件都有一个事件描述(或动作例程),它说明了当某一事件发生时,什么动作将被执行.

JMOSP模型中的属性描述了软件过程中实体和关系的静态特征,例如,一个源程序的属性可能包含它的作者名、完成日期、源程序文本等等.

进程是JMOSP的基本构成成分.进程分实体进程和关系进程两种,它们分别相关于一个实体或关系,它们描述什么事件将发生于这一实体或关系,并且这些事件将以什么顺序发生.进程从事件的角度刻划了相应的实体或关系在软件开发过程中的动态生命周期,在一个进程内部,事件的发生是顺序的;但不同进程之间事件可以并行发生.不同的实体进程之间的同步制约可以通过显式的进程同步指明,也可由同一事件出现在不同的进程中而隐式地指明.关系进程用来表示不同实体进程之间的复杂关系.一个关系进程将表明当一个实体进程中的某个事件发生之后,什么样的影响被传播到其它实体进程中.进程由一种类树的结构图定义,其中根结点代表实体类,其它结点则代表事件.这种结构图的语义和语法与JSD方法一样.

3 软件过程模型JMOSP

这一节以具体实例解释JMOSP模型是怎样模拟软件过程的.这些实例是关于一个典型的软件过程中某些实体类的进程结构.由于它们仅用于解释性的目的,因而并不是完全的,但它们足以说明JMOSP模型完全具有第1节中所指出的软件过程模型的若干重要特征.

在JMOSP中,每个实体和一个或多个进程相联系,这些实体进程描述了将在这个实体上发生的事件和这些事件的发生顺序.图1是一个表示"模块实现"这一实体进程的结构图.图1表明,一个"模块实现"被创建以后,它就开始了它在软件过程中的生命期,它将经历一系列的事件如"修改"、"静态分析"、"退化测试"等等.这个进程结构的核心部分是"修改",它由"修改/测试/报告"的重复组成,"修改/测试/报告"又由五个部分的顺序结构组成,其中的核心是"修改/编译部分"."修改/编译部分"由"修改/编译"的重复组成.

事件"连接到模块#"中的记号'#'表示它可匹配于一个模块名.该事件把本"模块实现"连接到由'#'表明的上层模块中.

进程中的每一个事件都有一个事件描述,它指出当该事件对某个实体或关系发生时,对实体或关系的属性将采取什么样的动作.对图1中的各事件的描述如下:

修改:修改模块的源代码;

静态分析:检查源代码的语法正确性;

编译:从模块的源代码产生目标代码;

测试性集成:为了测试,把模块的调用模块或桩模块集成在模块中并产生可执行的代码;

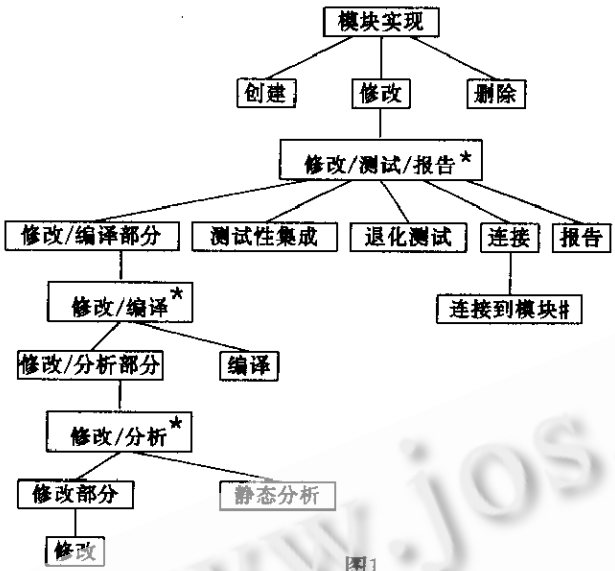

图1

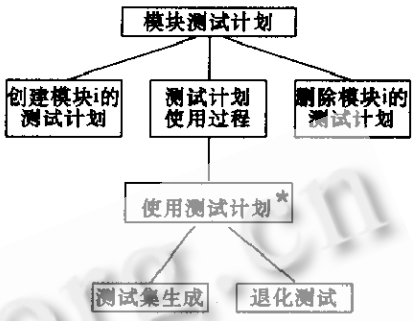

图2

退化测试：测试修改后的模块；

连接到模块#：把模块的目标代码连接到'#'指明的模块；

报告：产生本次修改的报告.

图 1 的进程结构描述了一个已实现模块的生命周期. 在这一进程中,事件的发生顺序受到了若干限制,例如,"静态分析"总是发生在"编译"之前,而两次"静态分析"之间至少有一次"修改"发生等等. 但是,图 1 对发生于"模块实现"的事件顺序所加的限制是不完全的,例如,在实际过程中,"退化测试"发生的前提条件之一是存在相应的测试计划,这些限制在图 1中并没有得到反应. JMOSP 模型用进程同步的方式加上这些限制. 一个事件可能与多个实体或关系进程相关,那么,这样一个事件的发生将引起不同进程的同步,即该事件必须等它所相关的进程的状态都允许它发生时才发生,例如,"退化测试"这一事件同时出现在图 1所示的"模块实现"进程和图 2 所示的"模块测试计划"进程中,那么"退化测试"的发生要求"模块实现"和"模块测试计划"的进程同时存在并且测试集已被生成.

实体之间的关系在软件过程中有重要的作用,它不仅表示了实体之间因某些属性相同而发生的静态关系,还影响各实体中事件的发生顺序以及表示一个实体的状态变化可能影响的范围. 在 JMOSP 模型中,关系和实体一样也可被一个或多个进程以同样的方式进行模拟. 一个模拟关系的进程指出当一个实体的状态发生变化时,什么影响可能被传播到相关的其它实体.

图 3 描述了一个关系进程结构,它模拟软件配置对模块的包含关系. 它表明如果配置 c 包含模块 i,则运行 c 之前必须将模块 i 连接到 c 上. 如果考虑到配置 c 包含模块 i,且模块 i 依赖于模块 j,则问题变得更为复杂. 在这种情况下,将 i 连接到 c 之前,必须保证 i 的目标代码是按照 j 的目标代码的最近版本生成的(这类似于 UNIX 的 MAKE 中的 time－stamp 手段)这可由图 4 关于模块依赖关系的关系进程结构模拟. 图 4 表明,如果模块 i 依赖于模块 j,那么,只要模块 j 产生了一个新的版本的目标码,在将 i 连接到任何构造之前,必须生成 i

的新版本的目标码.

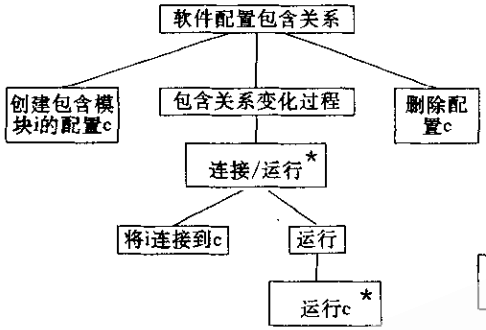

图3

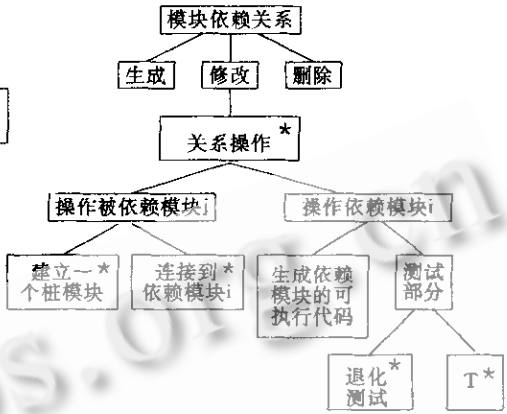

图4

JMOSP 模型并不为每一个实体和关系提供一个统一的进程结构图,而是提供了多个进程结构图,这是因为:

(1)在同一个图中说明进程中事件的顺序关系会变得很复杂,难于理解.

(2)多个进程结构图提供了对软件过程的多种视图.

(3)用多个进程结构图更好地体现了软件过程的并行性.

为了更灵活有效地描述软件过程,JMOSP 模型提供了参数化进程的概念.在进程结构图中可以引进一些参数,这些参数在进程被创建时被赋予一些特定的值.例如,一些软件产品可能要定期地进行后援存贮,因此,需要定义一个时钟实体进程.但是由于不同的软件产品对后援间隔时间的要求各不相同,那就可能需要很多不同时钟实体进程.此时,如果建立一个参数化的时钟实体进程就可带来很多的方便.

4 结束语

一个有效的软件过程模型在软件开发环境中扮演三个重要角色:决定软件开发工具的内部连接方式;决定由更新软件对象所引起的传播影响;提供辅助软件开发管理的信息.任何软件开发环境总是以软件过程模型作为基础的.传统的软件开发环境通常不明确说明所依赖的软件过程模型,而是将这种依赖性蕴含在由各个工具集成的软件开发环境的实现机制中.为了提高一个软件开发环境的适用范围,它不仅应该是软件过程敏感的,而且应该是可调整的,以适应于不同的软件过程.这就要求软件开发环境中有分离的机制来维护软件过程知识,这些知识是可更新的,并被用来驱动整个环境的动作,这类环境被称为面向软件过程的软件开发环境,而若要建造面向软件过程的软件开发环境,必须有有效的手段来描述软件过程,本文提出软件过程的 JMOSP 模型正是在这方面所作的新的尝试.我们已经应用 JMOSP 模型模拟了软件过程中若干子过程,诸如版本管理、组合测试等等.尽管我们的研究结果尚不成熟,但已在很多方面体现了用 JMOSP 模型模拟软件过程的潜在价值.

参考文献

1 L Osterweil. Software process are software too. Proceeding of the 9th International Conference on Software Engineering, 1987: 2—13.

2 W W Agresti. What are the new paradigms, new paradigms for software development: tutorial. IEEE Computer Society, 1986: 6—10.

3 B Curtis, H Krasher, V Chen *et al*. On building software process models under the lamport. Proceeding of the 9th International Conference on Software Engineering, 1987.

4 R C Water. Automated software management based on structured models. Software Practice and Experience, 1989, **19**(**10**): 931—955.

5 L C Liu, E Horowitz. A formal model for software project management. IEEE Trans. on Software Engineering, 1989, **15**(**10**): 1280—1293.

6 M Jackson. System development. Prentice—Hall International Inc., 1983.

SOFTWARE PROCESS MODEL JMOSP

Zhang Ran, Zhan Yuchao and Ye Yunwen

(*Department of Computer Science, Fudan University, Shanghai* 200433)

Abstract　Software process model has become a hot topic recently in software engineering research circle because it is regarded as a good way to improve the ability to produce high quality software. This paper proposes a software process model JMOSP after briefly evaluating several typical existed software process models.

Key words　Software process model, software process, JSD.