第四代语言 并行进化式 软件开发模型

软件开发

计算机科学 1996 Vol. 23 №. 5

79-81

支持第四代语言的并行进化式软件开发模型 CESD*)

TP311.52

李 彤 王黎霞

(云南大学计算机科学系 昆明 650091) (云南大学经济学院)

摘 要 Developing software concurrently is an effective way to shorten the software development cycle. In this paper, we lead the evolutionary prototyping approach into the concurrent software development process and propose a concurrent evolutionary software development model (CESD) and mechanisms to manage and control the concurrent development process. CESD model can support the software development using 4GLs which have become the main tools presently in the domain of application software development.

关键词 Fourth Generation Language, Concurrent Software Development, Evolutionary Prototyping, Development Process, Development Monitor.

目前,第四代语言(4GL)已成为应用领域的主流软件开发工具。随着实践的深入,人们逐渐意识到:目前广泛流行的瀑布式软件开发模型不适应基于 4GL 的应用软件开发活动。究其原因在于二者刻画问题空间思维方式的不匹配性。软件开发活动是一个不断反复、逐步求解的过程,4GL 快速高效的开发生产率有效地支持了这一过程;而瀑布式模型刻画软件开发活动过于呆板苛刻,未能很好地反映反复这一本质。此外,在应用软件开发过程中,很多开发活动实际上是可以并行的,这一点在开发实践中已得到验证,瀑布式模型未能提出在较高抽象级上并行的机制。因此,软件开发模型如何从概念上到本质上支持基于 4GL 的软件开发活动,成为了亟等解决的问题之一。

一、并行进化式软件开发模型 CESD

如上所述,在传统的软件开发过程中很多开发活动是并行的。这给我们一个启示,能否在较高抽象级上使软件开发活动并行起来?

事实上,参与软件开发活动的往往是一个课题组的很多人。大型软件系统的开发甚至还要划分为若干个课题组,每一个子课题组解决一个或若干个子问题,实际上是可以并行工作的。因此,我们可以把一个软件系统的开发过程划分为若干个可以并行进行的成分,这个成分我们称为开发进程(Development Process),每一个开发进程完成一个子系统或一个模块的开发任务。当各个开发进程都完成之后进行系统集成和测试,最终完成整个系统的开发。开发进程是并行软件开发过程中的基本并行成分。为了反映软件开发过程不断反复这一特性,我们把进化式原型方法引入到每个开发进程中,提出了如图 1 所示的并行进化式软件开发模型 CESD(Concurrent Evolutionary Software Development Model)。CESD 模型把进化式原型方法和并行开发方法集成在一起,它具有以下特点:

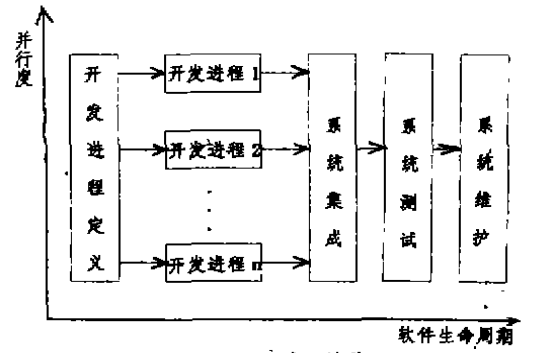
(1)并行存在于整个软件开发过程中,传统的串行模型可能在某些局部是并行的,但这往往在较低抽象级上进行。而 CESD 模型在整个开发过程中是并行的,并行在较高抽象级上进行。

(2)处理实体的多样性:不同于传统模型,CESD 模型的并行开发进程不但要面对自身处理的实体,而且还要同其它开发进程发生相互作用并共同处理某些公共实体;不但要对自身上一步骤所获结果进行处理,还要处理其它并行的开发进程所产生的结果。

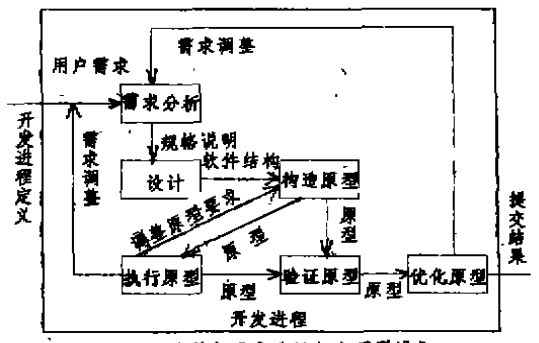
(3)并行控制复杂:由于有多个开发进程在并行,而这些开发进程又服务于一个整体(它们在开发同一个系统),它们必然发生各种各样的联系。因此,必须引入相应的管理与控制技术,才能保证开发结果的正确性。

*)本文得到云南大学青年教师科研基金项目"第四代语言[软件开发方法学研究"资助。李 彤 讲师,主要研究领域为软件工程和语言处理。王黎霞 讲师,主要研究领域为软件 程和会计电算化。

在CESD模型中，管理与控制技术成为了关键问题，因为我们可能会遇到在串行模型中所从未遇到的问题，是并行模型所独有的，它来源于模型本身。其中最主要的问题是各并行开发进程及其所获结果之间相互依赖所带来的复杂性。导致这个问题的根本原因在于系统内部各成分之间的联系性和各并行开发进程之间的相互作用。由于所获结果可能并行地被产生和修改，从而如何保证结果的一致性与相容性成为了关键问题。并行开发过程虽然以缩短软件开发周期为目标，但却使系统开发的复杂度大大增加，人员间接口也更加复杂。若不对各并行开发进程加以有效的管理和控制，就无法保证开发出的系统内部各成分间的一致性、相容性，也无法解决冗余问题(数据存储冗余和开发内容冗余)，使系统的正确性和可靠性得不到保证，并行开发所追求的缩短软件开发周期的目标也不能实现。



(a)CESD 宏观模型



(b)一个并行开发进程中的原型进化

图1 CESD模型

在CESD模型中引入进化式原型方法的目的在于该方法对改变的快速支持能力，反映了软件开发过程不断反复这一特征。进化式原型不仅是作为模型或实验需要而生成的，它实际上是一个实际系统的前身或核心。该方法在串行开发模型中已得到广泛的应用，证实为一种行之有效的软件开发方法。因此，我们把该方法引入到CESD模型的并行成分——开发进程之中，在适当的工具(如4GL类工具)支持下，进化式原型对于降低CESD模型中并行开发进程控制的复杂度是十分有效的。

由于开发的对象是一个系统，那么系统内各成分之间必然存在各种联系。因此，为开发一个统一的系统而划分出的各并行开发进程必须存在这样和那样的依赖关系，或者说这些并行开发进程存在交互性和相互作用。它们互相配合，同步推进，完成统一的目标。因此，管理可以从以下三个方面进行：

(1)项目管理。项目管理在各种软件开发活动中都是存在的，但在并行开发模型中却显得更加重要。项目管理处于管理活动的高层，它的任务是有效地分配和控制各种开发资源，协调从事各开发活动的小组间的关系，对系统进行集成和测试，以确保开发的质量和速度。系统开发的资源包括数据资源、人员资源、阶段性成果、文档等等。

(2)产品管理。每个并行开发进程在一个阶段完成后都提交其产品或中间产品，包括各种文档、原型、代码、数据和可提交的产品等。产品管理的任务是对各种产品和中间产品加以有效的分类、归档，对可以运行的使用测试技术加以验证，对不可运行的采用复审方法加以评价。

(3)并行开发进程的管理。对并行开发进程的管理是指从技术上提供一系列有效的控制机制，对各并行开发进程进行管理和控制，保证系统的正确性和系统目标的实现。其任务主要有二个方面：一是并行开发进程的定义；二是并行开发进程的控制。

二、并行开发进程的管理

CESD模型对并行开发进程的管理必须解决以下两个方面的问题：

1. 并行开发进程的定义

一个软件系统被划分为若干成分(模块或子系统)，每一个成分由一个开发进程来完成。并行开发进程定义的任务是确定可以并行的若干开发进程以及这些开发进程所具有的结构(目标、需求、开发进程间依赖关系、系统集成接口等)。成分的划分应有利于开发进程的并行。并行开发进程定义的步骤如下：

(1)根据开发目标和并行开发的要求初步确定若干并行开发进程；

(2)寻找开发进程间的相互作用关系并把具有相互作用关系的部分分离出来；

(3)分析分离出来的部分是否能成为一个新的开发进程。若能，将其定义为一个新的开发进程；若不能，则将其定义为一个开发管程(Development Monitor)；

(4)对各开发进程和开发管程加以优化,该归并的加以归并,该分解的加以分解;

(5)若开发进程集合和开发管程集合还可进行优化,则返回第(2)步;

(6)结束。

并行开发进程的定义结束后,我们获得了二个集合:开发进程集合和开发管程集合。开发进程是并行开发的基本单位,开发进程间的相互作用关系由开发管程来处理。开发进程和开发管程之间的关系如图 2 所示。

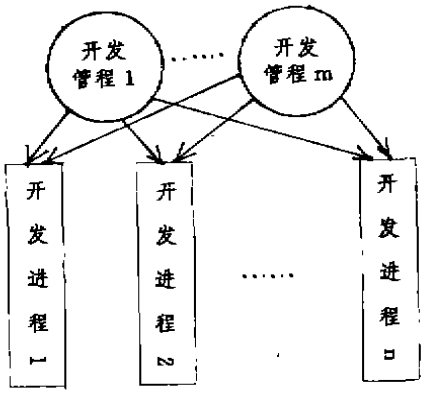


图 2 开发进程和开发管程的关系

2 处理开发进程间相互作用关系的机制—开发管程

开发管程是将并行开发进程中有相互作用关系的成分分离出来而得到的,因此它至少涉及到二个或二个以上的并行开发进程。在从并行开发进程中分离出开发管程后,并行开发进程之间的相互作用即通过开发管程来处理,从而一个并行开发进程的其余成分变得同其它并行开发进程没有关系,使得并行开发可以独立地进行,避免了接口关系的过于复杂化和冗余问题的产生。当并行开发进程间发生联系时,可认为已被开发管程所解决。开发管程是从若干个开发进程中分离出来的共同的成分,这些开发进程必然有共同的作用对象,不同的开发进程作用于该对象上的内容可以不一样。我们可以把开发管程看成是由对象和施加于该对象上的操作所构成。并行开发进程通过开发管程实现相互作用如图 3 所示。

在系统开发过程中,可以设置专门人员处理开发管程,完成开发管程的工作。开发管程使得分散在各个并行开发进程中的对共享对象的操作进行集中处理,大大减少了系统开发的复杂度,使得各并行开发进程能够按自己的速度向前推进,避免或减少了冗余,保证了各并行开发进程所开发出的成分间的一致性与相容性,使各并行开发进程之间的相互作用更为清晰。

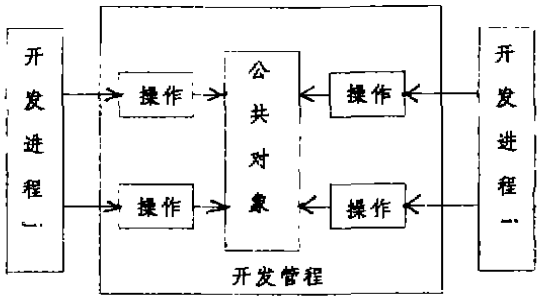


图 3 并行开发进程通过开发管程实现相互作用

三、CESD 模型对第四代语言的支持

CESD 模型以缩短软件开发周期为目标,引入了进化式原型方法并使开发过程并行起来。这就要求开发工具具有快速灵活的反应能力和原型构造能力,还要求便于组装和系统集成。因此,没有强有力的支撑工具是很难想象的。具有面向问题和非过程化特征的 4GL 则是一类比较理想的 CESD 模型支撑工具。由于 4GL 具有友好的用户接口、面向问题和非过程化等特点,因此可以大幅度提高应用软件开发生产率,非常适于作原型构造工具。反之,4GL 也需要有适于 4GL 特点的软件开发模型的支持。CESD 的二个主要特征是并行和原型进化。首先,并行使软件开发活动在较高抽象级上同时进行,因此并行控制的复杂度就大幅度上升。若系统实现采用 4GL 作工具,那么在需求分析和设计阶段就要结合 4GL 面向问题的非过程化特点来进行,需求分析和设计的方法和工具也较传统模型有所不同,一般而言更简洁,文档的种类和数量也较少,这样就使并行控制的复杂度大大下降,使 CESD 模型更便于实施。其次,原型的进化需要有效的工具支持才有实用意义。进化式原型方法强调经粗略的需求分析和设计之后就快速构造原型,并反复修正和调整原型,同时也修正和调整需求说明和设计说明,使原型不断进化成为最终系统。因此,原型进化的主要特征就是反复性和多变性,而这一点正是 4GL 相对于传统工具的优势所在。用 4GL 作为原型构造工具,经反复修正和调整原型,使原型最终进化为成熟的产品。因此,CESD 模型支持 4GL 进行应用软件的开发,它有利于充分发挥 4GL 的优势,实现缩短软件开发周期的目标。

结束语 本文提出了支持第四代语言的并行进化式软件开发模型 CESD 及相应的管理与控制机制,但是还存在着以下尚待解决的问题:有效地对并行开发进程进行控制的形式化模型和支撑工具、并行开发进程划分的算法和同步机制、CESD 模型与面向对象技术的关系等。这些问题随着我们研究工作的深入将逐步加以解决。(参考文献共 11 篇略)