CPMS中例程处理机制的设计与实现*)

林向宇 胡 昊 鲍钦迪 杨 玫 吕 建

(南京大学计算机软件新技术国家重点实验室 计算机软件研究所 南京210093)

摘 要 过程支持系统(PSS,Process Support System)是一个支持定义、执行和监控业务过程的软件系统,其中通过过程模型描述过程的各种控制结构。在基于CMM的过程支持系统——CPMS(CMM-Based Process Management System)中,由于CMM中庇护性过程的存在,简单的过程模型的控制结构已显得不够灵活有效,需要在过程模型中加入一些新的机制以丰富其对CMM庇护性过程的支持。本文介绍了一种对CMM庇护性过程支持的机制——例程处理机制,其设计在借鉴了程序设计语言中异常处理机制的基础上,又结合了庇护性过程的实际需要对异常处理机制进行了部分修改。最后本文还给出了系统对例程处理机制的实现。

关键词 例程处理,CPMS,CMM,过程支持系统

Design and Implementation of the Routine Mechanism in CPMS

LIN Xiang-Yu HU Hao BAO Qin-Di YANG Mei LU Jian

(State Key Lab. for Novel Software Technology, Institute of Computer Software, Nanjing University, Nanjing 210093)

Abstract Process support system is a software system which can specify, execute and monitor business processes. It uses process model to describe the control structures of processes. In CPMS (CMM-based Process Management System), which is a CMM based process support system, because of the existence of umbrella processes in CMM, the control structures of the basic process model are not flexible and capable enough. We need to add new mechanisms to the process model to support umbrella processes. This article presents a mechanism supports the umbrella processes in CMM, which is named Routine Mechanism. The design of routine mechanism uses the idea of exception handling in programming language but makes some modifications according to the characteristics of umbrella processes. Finally, this article also describes the implementation of routine mechanism.

Keywords Routine mechanism, CPMS, CMM, Process support system

1 引言

实践证明,实施CMM能有效提高软件企业的生产效率及软件的质量[9]。实施CMM的关键在于对软件过程的改进,而软件过程的自动化能有效提高软件过程改进的效率[7],自动化的途径之一是使用"过程支持系统"(Process-Support System)。过程支持系统是一个支持定义、执行和监控业务过程的软件系统,其主要功能是定义业务过程,自动执行并对业务过程的执行情况进行监控[2,6]。CPMS正是一个基于CMM的过程支持系统,它能够支持符合CMM规范的软件过程。

过程支持系统通过过程建模语言(PML)描述过程模型,并通过过程引擎自动执行过程描述。过程模型中一般都包含控制过程活动的简单控制结构,如顺序、选择、并发、合并等等。目前,过程模型需要包含的控制结构并没有统一的标准,而是根据实际应用的需求而定。在CPMS中,过程模型除了要包含一般过程支持系统的控制结构外,还需要符合CMM的实际应用需求。庇护性过程,如SQA,SCM等,就是一类对CMM的实现而言必不可少的过程,对其抽象建模后发现,庇护性过程具有随机启动、流程固定和可返工这样三个特性。

在对庇护性过程的特性进行分析后发现,一般过程模型中的控制结构很难灵活有效地支持庇护性过程,而程序设计语言中异常处理机制却提供了支持庇护性过程的良好手段,尤其是在支持庇护性过程的随机启动这一特性上。因此,CPMS对过程模型进行了扩充,结合异常处理的思想提出了例程处理机制,并在过程引擎上实现了运行时的支持,从而能够有效地支持CMM中庇护性过程的定义和执行。

本文将详细介绍CPMS中的例程处理机制的设计和实现。第2部分简单介绍了CPMS系统的组成以及CPMS采用的基本的过程定义和执行机制,这也是现有许多过程支持系统使用的机制。在第3部分,本文分析了庇护性过程的特性,并与Java的异常处理机制进行了比较,给出了例程处理机制的设计。第4部分论述了例程处理机制大致的实现方案。最后是全文的总结。

2 CPMS简介

一般过程支持系统的核心是过程建模语言(PML, Process Modeling Language)和过程控制引擎(PCE,Process Control Engine)[2,6],CPMS也符合上述的事实标准。图1是

*)本文得到国家863项目支持,项目编号:2001AA113090.林向宇 硕士研究生,主要研究领域为软件工程、软件过程技术.胡 昊 博士研究生,讲师,主要研究领域为软件过程,工作流技术,移动Agent技术.鲍钦迪 硕士研究生,主要研究领域为软件工程、软件过程技术.杨 玫 硕士研究生,主要研究领域为软件工程、软件过程技术.吕 建 博士,教授,博士生导师,主要研究领域为对象技术、分布计算技术、移动Agent技术.

CPMS 的结构，包括了两个部分：定义部分和解释执行部分。

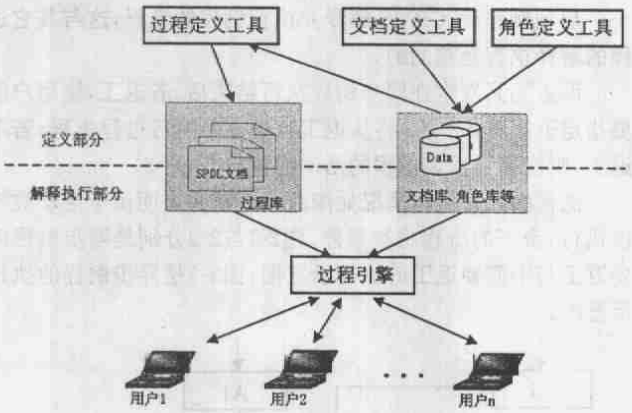


图1

CPMS 使用 SPDL(Software Process Description Language)作为过程建模语言，定义部分负责使用 SPDL 定义软件过程，它包括了过程定义工具、文档定义工具和角色定义工具。

解释执行部分的核心是过程控制引擎，CPMS 中称为过程引擎。此外，CPMS 中过程的执行需要用户参与，因此解释执行部分还包括了与用户的交互界面，用户通过 Web 界面与过程引擎进行交互。

CPMS 的工作过程如下：首先由定义部分定义软件过程及其软件过程使用的资源(文档、角色及成员等)，保存在 SPDL 文档中；过程引擎读取 SPDL 文档并执行软件过程，还负责监控软件过程的执行情况。

CPMS 采用了过程编程(Process Programming)的思想[4]，使用与应用程序编程类似的方式实现软件过程的定义和执行。SPDL 是用于描述过程的语言，而使用 SPDL 所建立的过程描述称为过程程序，解释器将严格按照 SPDL 的语法解释执行过程程序。

CPMS 的过程模型的基本元素是过程(Process)和活动(Activity)。过程是一段过程程序，由若干活动组成；而活动则是过程程序中的语句，在目前的 CPMS 中，活动的执行都需要用户的参与，没有系统自动执行的活动。

此外，CPMS 的过程模型具有三种基本的控制结构，分别是顺序、选择和并发。顺序和选择结构与结构化程序设计语言中的顺序和多分支结构相同；并发结构与选择结构类似，但它不做选择，而是并行执行多个分支。

这些基本的控制结构对于灵活有效的支持庇护性过程仍然存在结构上的不足，而借鉴了程序设计技术中异常处理技术的例程处理机制很好地解决了支持庇护性过程的问题。在下一节中，将通过对庇护性过程特性的分析引出例程处理机制的必要性，并分析了例程处理机制从 Java 的异常处理机制中借鉴的可行性。

3 CPMS 中的例程处理机制

3.1 庇护性过程

软件工程实践中存在着一类特殊的过程称为庇护性过程(也称为庇护性活动，Umbrella Activities)，这类过程独立于任何一个项目软件过程，且贯穿于整个过程，对项目软件过程起庇护作用。典型的庇护性过程有：软件过程管理、正式技术评审、软件质量保障、软件配置管理、度量、文档准备和生成、风险管理等[3]。

CMM 中许多 KPA 的实现都离不开庇护性过程的支持，其中包括：二级中的需求管理、软件项目跟踪和监督、软件质量保证以及软件配置管理；三级中的综合软件管理、软件产品工程以及同行评审；四级中的定量过程管理以及软件质量管理[5,8,9]。

我们以配置管理过程为例简要地说明庇护性过程的特性，配置管理过程的详细描述可参见文[5,8]。

首先也是最本质的，庇护性过程覆盖了整个项目软件过程，这意味着在项目软件过程的任何时候都可能执行庇护性过程，而庇护性过程的执行是由项目软件过程执行中产生的某种事件触发的。以配置管理过程为例，项目软件过程执行中的每次变更都需要执行配置管理过程加以控制，而变更可发生在项目软件过程执行的任何时刻。

其次，庇护性过程独立于任何一个项目软件过程，一个企业中所有项目软件过程使用的同一类庇护性过程通常都具有相同的流程。配置管理过程就是典型的例子，通常一个企业中所有项目软件过程使用的都是同样的配置管理过程。

再次，庇护性过程的执行可能产生对项目流程的返工。例如配置管理过程中评估了变更请求后，若该变更请求被拒绝，则提交变更请求的用户可能需要根据实际情况返工，再一次变更该配置项。

3.2 支持庇护性过程的过程模型

对庇护性过程的特性进行综合归纳后我们发现，能够灵活有效地描述并执行庇护性过程的过程模型必须具有以下三个特性。

1)随机启动。过程模型不能静态地刻画庇护性过程的进入点，而应该允许在运行的任何时刻随着事件的发生而随机地启动庇护性过程。

2)流程固定。庇护性过程在过程模型中没有必要重复定义，而应该将之描述为一种在多处被引用的预定义过程。

3)可返工。过程模型需要为庇护性过程提供返工的能力，重新执行部分流程。

在这三个特性中，随机启动这一特性是最本质的。但是，过程模型的三种基本结构只能提供静态的描述能力，使用它们所描述的过程模型是确定的，不具随机性。也就是说，仅使用过程模型中的三种基本结构无法体现随机启动这个特性，有必要对过程模型进行扩充。

对随机启动这一特性分析后我们发现，这一特性很类似于程序设计语言中的异常处理：在程序执行(项目软件过程执行)中的任何时候，一旦产生了异常(触发庇护性过程的事件)，将执行异常处理程序(庇护性过程)。

更进一步，庇护性过程尤其类似于出于监控的目的而进行异常处理的情况[3]。在这种情况下，异常的产生不是因为程序运行的错误，而是为了提供程序运行的情况供执行者进行监控和处理。

因此，为了实现随机启动这一特性，CPMS 结合了程序设计语言中异常处理的思想对过程模型进行了扩充，提出了例程处理机制。

目前，Java 中的异常处理[1]是较成熟的一种异常处理机制，并且 Java 的异常处理程序是随着异常的产生而执行的，同样具有随机启动的特点。因此 CPMS 中例程处理机制的设计具体参考了 Java 的异常处理机制。

但是，由于所面临的需求并不完全相同，因此 CPMS 的

例程处理机制对 Java 的异常处理机制做了一定的修改：

首先，Java 中对于相同的异常可以使用不同的异常处理程序，这是通过 catch 语句来实现的。而支持庇护性过程的过程模型必须具有流程固定的特性，CPMS 的例程处理机制采用了程序设计语言中过程的思想，对同一类例程使用相同的处理程序。

其次，由于处理的是程序运行的错误，因此在 Java 中，一旦产生异常，将终止 try 语句中所有语句的执行。而 CPMS 中的例程处理机制是出于监控过程的目的的，因此需要在处理完毕后恢复(resume)过程程序的执行。

最后，Java 的异常处理程序执行结束后，将顺次执行其后的程序，而不具有返工的操作。而支持庇护性过程的过程模型必须具有可返工这一特性，因此 CPMS 的例程处理机制需要提供返工的功能。

在下一节中将对 CPMS 中的例程处理机制做详细的阐述。

3.3 CPMS 的例程处理机制

CPMS 的例程处理机制包含了定义和执行两个部分的内容。定义部分的内容包括定义例程以及在过程程序中引用例程。

CPMS 在 SPDL 中增加了新的元素——例程(Routine)来描述庇护性过程，Routine 的 BNF 表示如下：

```
〈Routine〉::=Routine〈ID〉〈Name〉
            [〈Description〉]
            〈Property〉[〈Process〉]
```

例程包含了一段过程程序，这很类似于程序设计语言中的过程。一个庇护性过程在 CPMS 中被定义为一个例程。例程相当于 Java 中的异常处理程序，当触发该例程的事件产生时被调用执行。CPMS 中称触发例程的事件(例如配置项变更)为例程事件，它相当于 Java 中的异常。此外，针对不同庇护性过程的不同特性，CPMS 定义了两种类型的例程：同步例程和异步例程。这两类例程的不同主要体现在执行上，将在执行部分详细解释。

定义过程程序的活动时，需要显式定义活动执行过程中可能触发的例程，Activity 的 BNF 表示如下：

```
〈Activity〉::=Activity[〈Participants〉]
              [〈InputDocuments〉]
              [〈OutputDocuments〉]
              [〈Applications〉]
              [〈MeasureItems〉]
              [Routines〈RoutineList〉]
〈RoutineList〉::=〈ID〉{〈ID〉}
```

其中黑色加粗的部分用于定义所有可能触发的例程，CPMS 中称此为引用例程，这与 Java 中使用 catch 语句声明需要捕捉的异常是相类似的。

第二个部分的内容是执行，包括启动例程、执行例程以及返工。

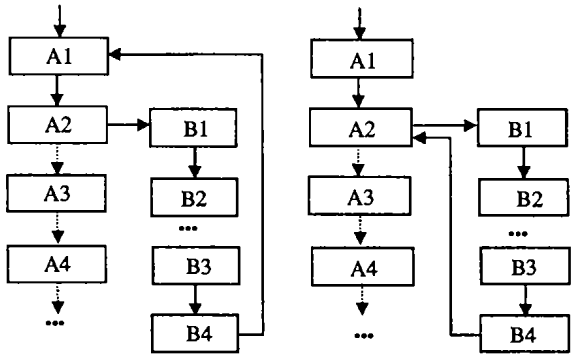
例程的启动是随机的。与 Java 中异常处理程序的启动类似，过程程序在执行中抛出例程事件，由 CPMS 截获例程事件并启动相应例程。在目前的 CPMS 中，例程事件是由用户抛出的，在活动执行过程中，CPMS 将活动引用的例程显示给用户，用户根据活动执行的情况抛出启动某个例程的例程事件。

此外，启动不同的庇护性过程将会对项目流程产生不同的影响。有的庇护性过程需要暂停项目软件过程的执行，例如正式技术评审；而有些庇护性过程允许项目软件过程与其并发执行，例如度量。因此，CPMS 定义了两种类型的例程：同步例程和异步例程。活动启动了同步例程后，将会被暂停执行直至同步例程执行结束；启动了异步例程后，活动的执行不被暂停，将与异步例程的执行并发进行。

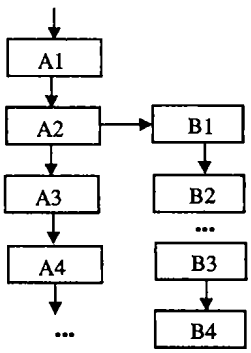
启动例程后，CPMS 解释并执行例程的流程，这与其它过程的解释执行是相同的。

而返工只发生在同步例程执行结束后。若返工，则用户需要指定返工点，CPMS 将从返工点处重新执行过程流程；若不返工，则恢复触发该例程的活动的执行。

例程的几种执行情况如图2，虚线箭头表明由于活动被暂停执行，余下的流程将被暂停。图2-1与2-2分别是同步例程需要返工与不需要返工的执行示意图；图2-3是异步例程的执行示意图。



2-1 同步：需要返工　　2-2 同步：无需返工



2-3 异步

图2　同步和异步例程

4 例程处理机制的实现

Java 的异常处理机制需要 JVM 做运行支撑环境，同样，CPMS 的例程处理机制离不开过程引擎运行时的支持，过程引擎的主要工作是启动例程、执行例程以及例程结束的处理，处理的流程如图3。

1. 启动例程

首先过程引擎要捕获产生的例程事件。过程引擎解释活动的语法，通过用户界面将活动引用的例程显示给用户，在用户选择了所要触发的例程后，用户界面产生相应的例程事件，并通过 Web 界面通知过程引擎。

捕获到例程事件后需要启动相应例程。同一个例程可能在多处被执行，必须保证它们之间不会相互影响。因此，在捕获到例程事件后，过程引擎首先为相应例程生成一个运行实例，该运行实例保存在数据库中。

随后，如果需要执行的是同步例程，还需要暂停触发该例程的活动的执行。过程引擎在数据库中保存了所有活动的运行状态，只需要将该活动状态修改为暂停的状态即可。

最后，过程引擎开始执行该例程的运行实例，这样就完成了例程的启动。

(下转第92页)

Reusability. In: Proc. Advances in Software Reuse, 1993. 80～88
3 http://home.sei.pku.edu.cn/95
4 Mili H, Mili F, Mili A. Reusing software: Issues and research directions. IEEE Transactions on Software Engineering, 1995, 21(6)
5 张世琨,张文娟,常欣,王立福,杨芙清. 基于软件体系结构的可复用构件制作和组装. 软件学报,2001,12(9):1351～1359
6 Plasil F. Behavior protocols for software components. IEEE Transaction on Software Engineering, 2002,28(11): 1056～1076
7 Opdahl A L, Sindre G, Vetland V. Performance consideration in Object-Oriented reuse. Selected Papers from the Second International Workshop on, 1993. 142～151
8 Stephen Sau, Taweponsomkiat C. An approach to object-oriented component customization for real-time software development. Computer Society, 2002
9 David L, Lary C, Augustin M. Specification and analysis of system architecture using Rapide [J]. IEEE Transaction on Software Engineering, 1995, 21(4): 336～355
10 Medvidovic N, Rosenblum D S. Domains of concern in software architecture and architecture description [A]. SHAW M. Proc. of' 97 USENIX [C], California, 1997. 199～212

(上接第88页)

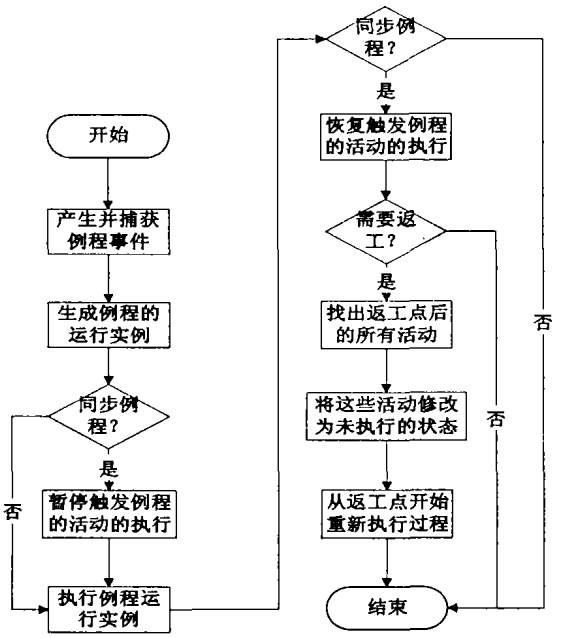


图3

2. 执行例程

例程运行实例的执行与其它过程的执行是完全相同的。

3. 例程结束的处理

例程运行实例执行结束后,过程引擎首先判断例程的类型。如果是异步例程,过程引擎不需要进行任何处理。

如果是同步例程执行结束,过程引擎首先将触发该例程的活动的状态修改为正执行的状态,以恢复该活动的执行。

此外,如果需要返工,过程引擎找出过程程序中在返工点后的所有活动,将这些活动的状态修改为未执行的状态,然后从返工点开始重新执行过程程序。

小结 CPMS 是一个基于 CMM 的过程支持系统,它需要能够支持符合 CMM 规范的过程。而 CMM 实际应用中除了一般的软件过程外,还存在着一类特殊的过程——庇护性过程,并且 CMM 的实现离不开庇护性过程的支持。

经过对庇护性过程的特性进行分析抽象后我们发现,能够支持庇护性过程的过程模型必须具有随机启动、流程固定、可返工三个特性。其中,随机启动是最本质的特性,并且它很类似于程序设计语言中的异常处理。而通过将过程模型与程序设计语言进行类比发现,过程模型的三种基本结构只具有确定性,不足以实现随机启动的特点。因此对过程模型扩充了类似异常处理的机制,在 CPMS 中,该机制称为例程处理机制。

CPMS 的例程处理机制的设计参考了 Java 的异常处理机制,同时根据应用需求进行适当的修改,主要表现在使用固定的例程处理程序、不中断过程程序而能够恢复其执行、可返工三个方面。

本文给出了 CPMS 的例程处理机制的详细描述,并给出实现方案。

致谢:本文受到了国家863项目"基于 CMM 的软件质量保障平台及应用"(2001AA113090)支持。

参 考 文 献

1 Strohmeier A, Chachkov S. A Side-by-Side Comparison of Exception Handling in Ada and Java, Version 1.1. ACM Press, 2001
2 Ellmer E. Three Perspectives on Process Supporting Environments: Integrating Workflow and Software Process Research. In: Proc. of the Intl. Symposium on Applied Corporate Computing, Oct. 1995. 25～27
3 Goodenough J B. Exception handling: Issues and a proposed notation. Communications of the ACM, 1975, 18(12)
4 Osterweil L. Software processes are software too. In: Proc. of the Ninth Intl. Conf. on Software Engineering. IEEE, 1987
5 Jalote P. CMM in Practice - Processes for Executing Software Projects at Infosys. Pearson Education, Inc., 2000
6 Bruynooghe R F, Parker J M, Rowles J S. PSS: A System for Process Enactment. In: Proc. of the First Intl. Conf. on the Software Process. IEEE Computer Society Press, 1991
7 Humphrey W S. Managing the Software Process. Addison-Wesley, 1989
8 Pressman R S 著,梅宏译. 软件工程——实践者的研究方法(第5版). 机械工业出版社,2002
9 卡耐基梅隆大学软件工程研究所编著,刘孟仁等译. 能力成熟度模型(CMM):软件过程改进指南. 电子工业出版社,2001