

基于委托代理模型的审批流转方案的研究

孔 芳

(苏州大学 计算机科学与技术学院, 江苏 苏州 215006)

(kongfang@zhz.org)

摘 要:提出了一个基于委托代理模型的审批流转方案。与一般传统审批流转方案不同,该模型针对传统模型中流转审批路线整体难以确定、流转路线及审批规则在审批过程中难以临时修改的问题,提出了可复用的审批流段,并为各审批流段设定委托监听器,由监听器对审批流的临时变化进行调整,这样形成了审批控制单元,增强了部件的可复用性和对各类变化的适应度。

关键词:委托代理;审批流转;监听器

中图分类号: TP390 **文献标识码:** A

Research of approval flow scheme based on consignment Agent model

KONG Fang

(Computer Science and Technology School, Suzhou University, Suzhou Jiangsu 215006, China)

Abstract: In general traditional approval flow scheme, there were some problems need to be resolved. It was difficult to set up the complete approval flows and it was not appropriate for the highfrequency temporary varieties. To resolve these problems, we divided the complete approval flow into some components. and binded each segment with some consignment agent listeners. The latter project is responsible for processing the adjustment to the temporary variety. So segment with some listeners becomes the reusable approval control units and helps to strengthen the adaptability of approval flow.

Key words: consignment Agent; approval flow; listener

在实际工作中,企业都会产生大量的单据、文档,例如各类申请表、合同、报销单、采购单、出库单等,其中很多单据都需要审批。随着信息化进程的不断推进,企业的这些单据、文档已经逐渐从纸质向电子化过渡,而原来的纸质审批过程也渐渐转入电子审批。许多 MIS 系统中都有关于电子审批流转的内容。

本文首先分析了企业内部的几种原子审批流转的结构,给出了一般传统审批流转的解决方案。在分析传统审批流转方案不足的基础上,提出了审批流转的委托代理模型,并对这一模型的特点、优势进行了说明。

1 传统审批流转方案

审批是企业中非常重要的一项日常事务。随着企业规模的不断扩大,多级、多人的联合审批是审批流转中必不可少的一类。

1.1 原子审批流转结构

原子审批流转结构一般只有三种:线性、并行和协商。

1) 线性结构

如图 1 所示,整个流程是一个线性过程,即审批通过,单据就流转 to 下一个节点进行审批;如果审批不通过,单据将回到提交者手中。

2) 并行结构

并行审批流转相当于日常生活中的举手表决,超过半数(或者某个要求)的审批过程可以继续流转到下一节点。而图 2 中的多个节点就是参与举手表决的审批人。某种意义上来说,线性结构的审批流转方式,实质上是一票否决的并行审批流转。

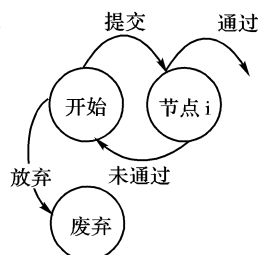


图 1 线性审批流转的流程

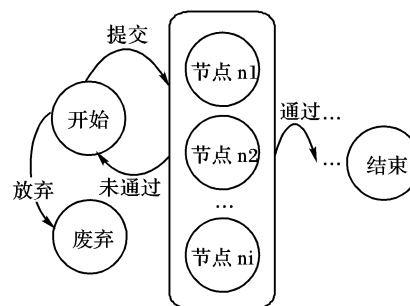


图 2 并行审批流转的流程示意图

3) 协商结构

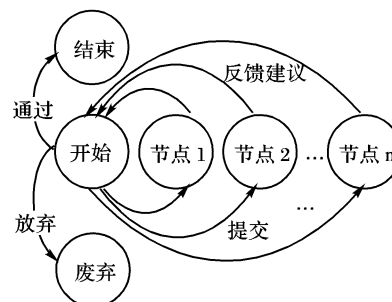


图 3 协商审批流转的流程示意图

单据审批的协商结构就是一个征求意见,但不给予决定权的过程。如图 3 所示,单据提交后,将它同时传送给多个审批者,接受他们的反馈意见和建议。这些意见和建议不一定

影响决定,帮助提交者作出更合理的决策。

1.2 一般传统审批流转方案

各种单据的审批流转实质上是上述三种原子结构的组合。由于企业内各种单据的审批流转路线各不相同,所以常规的做法是:

1) 首先将各类单据按其审批流转路线归类,将路线相同的归为一类;

2) 接着为每一类单据设定审批流转模板,在模板设定时指明所经过的节点信息,即哪些人可以参与这类单据的审批,他们的次序如何;

3) 模板设定完毕后,在某类单据提交时,就会按照已经规定的流转路线进行审批。

但这一审批流转方式在电子化后实际上存在一些很难解决的问题,具体包括:

1) 有些单据的审批流转路线很难确定。例如,某些单据需要首先采用协商方式去征求意见、建议,然后根据意见和建议再决定是采用一票否决制度,还是有超过半数以上的人同意即可。这样的后续审批流转方式由前面的审批结果决定的情况,显然是很难预先设定审批流转的路线的。

2) 单据的审批流转方式在模板设定好,按模板路线进行流转的过程中如果想要修改,是非常困难的。而在现实中却经常会出现这一情况。在传统的审批流转方式中,这是无法解决的。

3) 企业中所谓的规则,并不严格的对应我们所谓的模板。根据需要,临时增加一个审批环节,或删除一个审批环节的情况是很常见的。而传统审批流转方式对于这类情况的支持,只是一味的增加模板,这样造成了很多使用率极低的模板的存在。而且临时使用某一审批流转路径,还需先设定模板,再进行流转,这是非常不方便的。

2 委托代理模型

从上面的分析可以看到,造成传统审批流转方案实施困难的最主要原因就是,它将审批从开始到结束当成一个整体设定,然后按设定的路线完成。这在理论上讲是可行的,但现实中由于各类因素的影响,很难按这种理想状态发展。因此,我们提出了委托代理模型。主导思想就是能将审批流转的整体路线按用户需要分割成若干,当然最小情况就是原子结构。

2.1 委托代理模型的基本结构

委托代理模型如图4所示。我们首先给出委托代理模型中各构成元素的定义。

1) AC (Approval Component) 审批原子:共有三种审批原子,也就是前面给出的三种审批流转的基本结构,而原子中包含节点和 FC 流转控制。实质上,AC 是最简单的审批控制单元 ACU。

2) ACU (Approval Control Unit) 审批控制单元:它由多个审批原子 AC、子审批控制单元 ACU 和流转控制 FC 构成。在实际使用中,用户根据需要定制自己可确定的审批控制单元,再将 ACU 与相应的事件监听器 Event Listener 绑定,由某一监听器进行 ACU 间的一些特殊情况的处理。

3) FC (Flow Control) 流转控制:FC 是指正常流转中出现的几种信息流,主要包括:通过、不通过、提交、放弃四种。

4) Event List 事件列表:包含了流转控制中可能出现的特

殊情况,随着实际使用的展开,事件列表会不断充实。主要包含这样一些具体事件:

直接提交给 $\times \times$ 审批控制单元:不论所在 ACU 中原定的流程怎样,由当前审批者要求直接转到某一 ACU。

出现异常(例如:超时),临时将下一 ACU 更换成 $\times \times$ 审批控制单元。

退回给 $\times \times$ 审批控制单元再次审批,例如:部门经理对部门主管的审批结果有异议,要求他重新审批等。

希望下一审批环节能够多种审批单元同时进行。例如:某个议案(合同)在部门主管级进行并行方式的 ACU 中处理,同时也想在若干普通员工中以协商方式听取意见、建议等。

5) Event Listener 事件监听器:在某一 ACU 级上注册事件监听器对象,由它承担代理的角色,负责各类事件的监听。随着事件列表中事件数量的不断增加,监听器也会相应分类,不同类型的监听器负责监听某类事件。在各类监听器中,给出了事件发生后的处理方案,即事务处理器。

6) Action Handler 事务处理器:负责根据用户传入的相关参数信息,进行事件的处理工作。

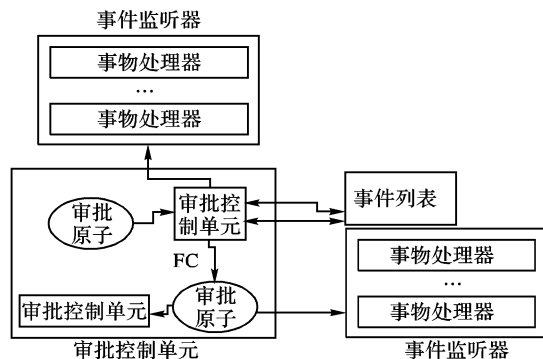


图4 委托代理模型

从图4中可以看到,ACU 是事件监听器的注册单元,不同级别的 ACU 中可以注册不同的事件监听器,再由这些监听器,即代理,完成各项任务。

2.2 委托代理模型的特点

委托代理模型的主导思想是:

1) 将难以确定的完整审批流程分割成可以确定的审批流程段,而段与段之间如何衔接,交给注册的监听器处理,而在实际运作中由当前审批者完成审批后指定下一审批流程段,确定后形成相应的事件消息通知监听器,由它进行流转控制;

2) 采用监控式流程处理模式,借助第三者来监控和触发事件,进行相应的流转控制。如果在段与段之间没有注册监听器,或者 ACU 足够大,那么它就形成了传统方式下的审批流转方式。

与传统审批流转方案相比,基于委托代理模型的审批流转方案有如下一些优势:

1) 要预先确定完整的审批流转路径(即静态的)是非常困难的,而某个审批环节相对比较容易。因此,基于委托代理模型的审批流转方案中的 ACU 的确定更容易,可行性更高;

2) 在委托代理模型中将完整的审批流转路径分割成了一个审批流转段,便于复用;

3) 即使设定好了流转路线,只要对可能出现特殊情况的

ACU 注册监听器,在有特殊情况发生时,可以指挥注册的监听器更改流程,设定审批流转具有很好的适应性;

4) 基于委托代理模型的审批流转方案对于临时要求,例如:重复审批、同时进行多个 ACU 等具有良好的支持。

3 委托代理模型的实现

从上述委托代理模型的介绍可以看出,它的实现主要由五个部分构成:

1) 节点定制:所谓审批就是某一单据或文档在人与人之间传递,给出最终的决定。因此很容易想到节点就是某个具体的人。而在实际工作中,人员的变动频度是比较高的,如果在定制委托代理模型时将节点定制成人员,在以后系统运作过程中,一旦人员变动,模型就要调整,这是不合理的。所以通常将节点定制成人员组、职务、角色等相对稳定的信息,而在指定流转的具体单据或文档时再选定某一组、职务或角色中的具体的人员。

2) 事件状态对象库:ACU 内部在没有发生特殊情况时是按照 ACU 定制时指定的流转流程进行处理,不涉及特殊处理。而系统中的每一特殊情况都对应一个自定义事件状态对象类。在 Java 这样的面向对象的程序设计语言中都提供了对自定义消息、事件的支持。在 Java 语言中可以通过继承 java.util.EventObject 生成自己的事件状态对象类。

3) 监听器接口库:与定义的事件状态对象相对应,创建相应的监听器接口。在 Java 中,事件操纵方法都被定义在继承了 java.util.EventListener 类的 EventListener 接口中,根据事件状态对象库中定义的各种事件信息定义生成监听器接口库。

4) 监听者库:监听者是相应监听器接口的具体实现,是一个个的 Java 类,每个监听者都实现了具体的某个监听器接口,也就是给出了事件的响应。

5) ACU 定制:AC 是最简单的 ACU,ACU 实质上是由节点集、控制方式(线性、并行、协商)构成。而 ACU 定制除生成 ACU 单元外,还要建立 ACU 与事件监听者间的事件流,也就是在 ACU 单元中必须为事件监听者提供注册和注销的方法。在 Java 中,指定节点、控制方式,形成 ACU 单元,然后在 ACU 单元的类中定义一个储存事件监听者的 Vector 数组,接着在 ACU 单元类中添加 add $\times \times$ Listener 和 remove $\times \times$ Listener 方法即可。

要建立基于委托代理模型的审批流转系统,除上述提到的主要构成部分外,还涉及具体系统中要进行审批流转的对象,此处不再进一步展开。

4 结语

本文在分析目前一般审批流转方案不足的基础上提出了一个委托代理模型,该模型具有一定的通用性,可用于电子单据、文档等对象的审批流转,对审批流转类系统的设计与实现有一定的参考价值。

参考文献:

- [1] 陶晓东,潘浙丹. 基于多代理技术的申请审批系统[J]. 计算机工程, 2003(20): 191 - 193.
- [2] 高志伟,牛江川,张利,等. Internet 环境下政府并联审批系统研究[J]. 计算机系统应用, 2004, (3): 12 - 15.
- [3] HORSTMANN CS, CORNELL G. Java 2 核心技术(卷 I: 基础知识)[M]. 程峰,黄若波,章恒翀,译. 北京: 机械工业出版社, 2003.
- [4] <http://info.shangdu.com/new/2003-5-8/200358110522.htm>, 2005.
- [5] <http://www.matrix.org.cn/resource/article/0/264.html>, 2005.

(上接第 490 页)

在仿真过程中,只需给出新的坐标和松弛长度。表 1 中第三列给出了执行投影步的时间,第四列是计算拓扑结构的时间,最后一列是总的时间。这些值都是在一秒的仿真时间内在 PIV/2.0GHz 处理器上的平均时间。

表 1 每秒仿真时间内计算虚拟粒子所用的平均时间

实例	粒子数目	投影时间(ms)	拓扑计算时间(ms)	共计(ms)
圆桌	100	38.5	38	76.5
球	400	150.5	83.5	234

因为只使用了位于边界上的虚拟粒子,所以要为物理仿真计算新的拓扑结构。整个执行情况依赖于基本的物理模型和使用的数值求解器。进一步的工作主要集中在投影算法的改进上。更加优化的加速方法和算法改进将进一步减少投影的时间。

参考文献:

- [1] TERZOPOULOS D, FLEISCHER K. Deformable models[J]. The Visual Computer, 1988, 4: 306 - 331.
- [2] BREEN DE, HOUSE DH, WOZNY MJ. Predicting the drape of woven cloth using interacting particles. Proceedings of SIGGRAPH'94 (Orlando, Florida, July 24 - 29, 1994)[A]. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH[C]. ACM Press. ISBN 0-89791-667-0, 1994. 365 - 372.
- [3] EBERHARDT B, WEBER A, STRASSER W. A fast, flexible,

- particle-system model for cloth draping[J]. IEEE Computer Graphics and Applications, 1996, 16(5): 52 - 60.
- [4] NG HN, GRIMSDALE RL. Computer graphics techniques for modelling cloth[A]. IEEE Computer Graphics and Applications[J]. 1996, 16(5): 52 - 60.
- [5] HUTCHINSON THD, PRESTON M. Adaptive refinement for mass/spring simulations[A]. In 7th Eurographics Workshop on Animation and Simulation[C]. 1996. 31 - 45.
- [6] BARAFF D, WITKIN A. Large steps in cloth simulation[A]. SIGGRAPH 98 Conference Proceedings, Annual Conference Series[C]. ACM SIGGRAPH, Addison Wesley, ISBN 0 - 89791 - 999 - 8, July 1998. 43 - 54.
- [7] VOLINO P, MAGNENAT - THALMANN N. Efficient selfcollision detection on smoothly discretized surface animations using geometrical shape regularity[J]. In EuroGraphics, 1994, 13: 155 - 166.
- [8] PROVOT X. Collision and self-collision handling in cloth model dedicated to design garments. In Graphics Interface'97[A]. Canadian Information Processing Society, Canadian Human-Computer Communications Society[C]. 1997. 177 - 189.
- [9] SHEWCHUK JR. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. Applied Computational Geometry: Towards Geometric Engineering, volume 1148 of Lecture Notes in Computer Science[A]. From the First ACM Workshop on Applied Computational Geometry[C]. 1996. 203 - 222.