用着色 Petri 网建模工作流模式*)

闻立杰　王建民　孙家广
(清华大学软件学院　北京 100084)

摘　要　工作流模式指在工作流过程模型中反复出现的过程基本构造，是衡量工作流建模语言在控制流方面的表达能力和适用性的重要标准，目前市场上的工作流引擎对其支持得并不好。本文重点阐述了基于着色 Petri 网的工作流建模语言对当前流行的 20 种工作流模式的支持情况。结果表明，该语言不仅能很好地支持全部模式，而且具有建模简洁、准确的特点。同其它建模语言相比，用着色 Petri 网建模工作流模式具有较好的灵活性和扩展性。它也为如何使基于着色 Petri 网的工作流引擎能够正确、有效地支持全部 20 种工作流模式提供了有意义的指导。

关键词　工作流，建模语言，模式，着色 Petri 网，控制流

Modeling Workflow Patterns Using Coloured Petri Nets

WEN Li-Jie　WANG Jian-Min　SUN Jia-Guang
(School of Software, Tsinghua University, Beijing 100084)

Abstract　Workflow patterns refer to the basic process constructs which occur repeatedly in workflow process models. It is an important criterion to evaluate the expressiveness and suitability of a workflow modeling language in control flow perspective. Currently, no workflow engine can fully support all the needed patterns. This paper gives a comprehensive analysis on the adaptability of the workflow modeling language based on Coloured Petri Nets in modeling the 20 popular workflow patterns. The analysis result shows that the language can support all the workflow patterns very well with the characteristics of concision and preciseness. In addition, compared with other modeling languages, Coloured Petri Nets-based modeling language is more flexible and extensible in modeling workflow patterns. It is also helpful for making Coloured Petri Nets based workflow engines support all the 20 workflow patterns correctly and efficiently.

Keywords　Workflow, Modeling language, Pattern, Coloured Petri Nets, Control flow

1　引言

近年来，工作流作为业务过程管理、业务过程协调及业务活动监控的核心技术，在企业信息化、政务电子化、教育数字化等领域得到越来越广泛的应用。目前，市场上已有 200 多家工作流产品，其中工作流管理系统以 Staffware、COSA、InConcert、FLOWer、MQSeries 等比较有名。为进行工作流过程的设计和定义，工业界和学术界提出了许多种过程建模语言。其中一些语言基于现存的建模技术，如 Petri 网、状态图、任务网、活动图等；其它语言都是系统特定的。

Petri 网是一种适合描述并发特性的系统模型。它既有严格的形式化定义，又有直观的图形表示；既提供丰富的系统描述手段和系统行为分析技术，又为计算机科学提供坚实的概念基础[1]。着色 Petri 网是在传统 Petri 网的基础上扩展颜色得到的高级 Petri 网，它不仅继承了传统 Petri 网的所有优点，更通过 Token 颜色值、库所颜色集、弧表达式、警卫函数等增强了控制逻辑表达能力[2]。笔者所在的工作流小组就采用了着色 Petri 网作为工作流过程建模语言。

工作流模式指在过程模型中反复出现的过程基本构造，与特定的工作流语言无关。在工作流模式方面的研究，以荷兰青年学者 Wil van der Aalst 的工作最为突出。在工作流模式[3]一文中，为比较不同工作流建模语言在控制流方面的表达能力和适用性，通过抽象多个组织中的实际业务需求，总结提出了 20 种常用的工作流模式，并切实调查比较了 15 种商业工作流产品的建模语言对这些模式的支持情况。比较结果表明，目前还没有任何产品支持全部 20 种模式。除 5 种基本控制流模式外，多数产品甚至只支持高级工作流模式(即模式 6～20)的一个小的子集。在 YAWL[4]一文中，作者指出 Petri 网适合建模基于状态的工作流模式，但对于一些高级模式(如多重实例模式、高级同步模式、取消模式等)，传统 Petri 网乃至高级 Petri 网，都显得力不从心；即使能够采用建模技巧做到支持这些高级模式，也会给过程设计人员带来很大的工作负担。为此，作者提出基于 Petri 网的新工作流建模语言 YAWL，通过设计一组具有全新语义的建模元素，并组合使用这些建模元素来支持全部模式。Daniel Moldt 等人提出使用 Reference Net 进行基于模式的工作流设计[5]，Reference Net 是面向对象的高级 Petri 网，利用 Reference Net 的独有特征(带有同步通道和柔性弧的网实例)可以很容易地实现对 20 种模式中全部高级模式的支持，并重点针对多重实例模式、高级同步模式和取消模式给出自己的解决方案。余鹏等人利用 Petri 网来描述和分析工作流的各种模式，比较模式之间的异同，并对模式进行合并和重划分[6]。龚晓庆等人在信牌驱动模型中的工作流模式[7]一文中，评估信牌驱动模型在工作流过程控制流建模方面的表达能力和适用性。信牌驱动模型是一种基于 Petri 网的工作流计算模型，它支持绝大多数工作流模式。陈大锋等人在对多重实例的类型和实现技术进行分析的基础上，提出了一种基于“活动数组”的方案来简化多重实例模式的建模[8]。

本文重点阐述着色 Petri 网对工作流模式的支持情况，其强大的控制逻辑表达能力配合工作流引擎内部应用调用，可以比较简洁、准确地实现全部 20 种工作流模式。同以前的工作相比，笔者首次给出采用现有成熟建模技术——着色 Petri

*)国家自然科学基金(60373011)；973 项目(2002CB312006)。**闻立杰**　博士生，研究方向为工作流建模、过程挖掘和适应性工作流；**王建民**　教授，博士生导师，主要研究方向为组织信息系统、数据库与工作流技术；**孙家广**　教授，博士生导师，中国工程院院士，主要研究方向为计算机图形学、计算机辅助设计及管理技术与系统以及软件工程与系统。

网建模所有模式的一套完整方案，并在实际的工作流管理系统中做了实现和验证，实践证明了该方案的可行性。本文首先有针对性地介绍基于着色 Petri 网的工作流建模语言的建模元素，接着详细介绍如何用着色 Petri 网建模全部 20 种工作流模式，最后给出总结并对未来的工作进行展望。

2 建模元素说明

工作流相关术语和概念完全符合工作流模式[3]一文中的规范。给出的模式示例也基本参照该文中给出的业务需求及相应示例。

着色 Petri 网相关术语和表示法完全符合 Kurt Jensen 等人提出的规范[2]：

圆圈表示库所，用于存放控制数据。圆圈内的文字表示该库所对应的颜色集，这里忽略了库所名称。

方框表示变迁，方框内的文字表示该变迁对应的活动名称，以后直接称变迁为活动。变迁上可附加警卫函数，表示活动的执行条件。忽略了名称的活动被称作哑活动，一般由工作流引擎内部自动执行。

箭头表示弧，用于连接库所和变迁，弧上的标注文字表示控制数据。从库所到变迁的弧上的表达式对应活动开始时要消耗的控制数据。同理，从变迁到库所的弧上的表达式对应活动完成后要产生的控制数据。例如，1`0 表示控制数据的个数是 1、值是整数 0。

更详细的有关着色 Petri 网变迁就绪和实施规则，参见 Kurt Jensen 先生的经典著作[2]。

3 模式实现

下面按照工作流模式[3]一文提出的模式分类，逐一给出基于着色 Petri 网的模式实现。

3.1 基本控制流模式

下面 5 个基本控制流模式几乎被全部工作流过程建模语言支持，但不能体现出基于着色 Petri 网的工作流过程建模语言的优势。

模式 1(顺序)

在同一工作流过程中，一个活动在另一个活动完成后就绪，则这两个活动之间存在顺序依赖关系，需要用到顺序模式。图 1 给出了两个活动 A 和 B 使用顺序模式的示例。

图 1 顺序模式

图 1 表明活动 A 完成后，会给自己的输出库所(也是活动 B 的唯一输入库所)生成一个值为 1 的整型控制数据；之后活动 B 才可能绑定这个控制数据，进入就绪状态，等待其对应的执行者来完成。

模式 2(并行分支)

在工作流过程中，一个单独的控制线程分裂为多个可以并行执行的控制线程，使得某些活动可以被同时或以任意次序执行，这时需要引入并行分支模式。图 2 给出了活动 A 执行后，B、C 两个活动可以被并行执行的示例。

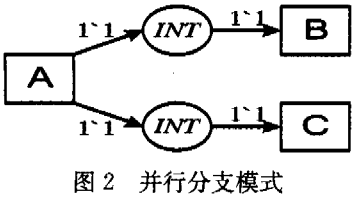

图 2 并行分支模式

模式 3(同步)

在工作流过程中，将多个并发活动汇聚成一个控制线程，即同步多个控制线程时，就需要引入同步模式。图 3 给出了 A、B 两个活动同步于活动 C 的示例。

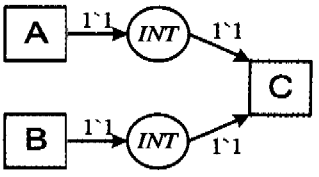

图 3 同步模式

模式 4(互斥选择)

在工作流过程中，根据控制数据选择多个分支活动中的一个分支时，会用到互斥选择模式。图 4 给出了活动 A 执行后，根据整型控制数据 x，活动 B、C 之一被自动选择执行的示例。其中，活动 B 的执行条件是 $x<5$，而活动 C 的执行条件是 $x>=5$，两个执行条件是互斥的。若不根据控制数据而是 B、C 的执行者主动做出决策，则去掉 B、C 上的警卫函数即可，此时该模式会等同于后面的延迟选择模式。

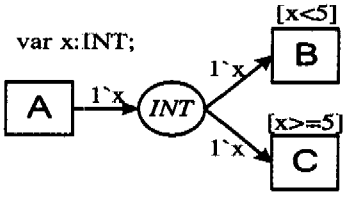

图 4 互斥选择模式

模式 5(简单汇聚)

在工作流过程中，两个或多个可选分支非同步地汇聚在一起时，需要用到简单汇聚模式。该模式对过程前面部分的假设是这些可选分支从来不会被并行执行。图 5 给出了两个可选分支(活动 A 和 B)进行简单汇聚的示例。活动 A、B 中的任何一个活动被执行后，活动 C 均可进入就绪状态。

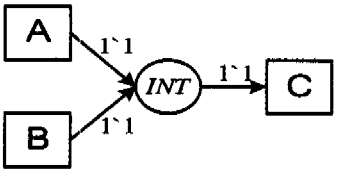

图 5 简单汇聚模式

3.2 高级分支与同步模式

多数工作流引擎都不能直接支持全部这类模式，然而它们在现实的业务场景中相当普遍。

模式 6(多重选择)

在工作流过程中，根据控制数据选择一组分支活动中的多个时，会用到多重选择模式。图 6 中，在活动 A 执行前或执行后，可得到整型控制数据 x 和 y。根据这些控制数据，引擎将自动决定选择执行活动 B 和 C 或活动 B 或活动 C 或二者均不执行。活动 B 的执行条件是 $x>5$，活动 C 的执行条件是 $y>7$。不带标签的两个哑活动分别当 B 或 C 的执行条件不满足时，自动移除其输入库所当中的控制数据 x 或 y。若不根据活动 A 产生或传递的控制数据决定哪些活动可以被执行，而是由 B、C 的执行者主动进行选择，则可以去掉这些

活动上的警卫函数。但此时无论执行者是否愿意执行 B，都必须明确指定，即要么选择执行活动 B，要么选择执行活动 B 对应的哑活动；活动 C 亦是如此。若用户只在执行活动 B 时才通知引擎，而不想执行 B 时不采取任何动作，则需要采用图 7 中前半部分所示的多重选择模式的第二种实现。

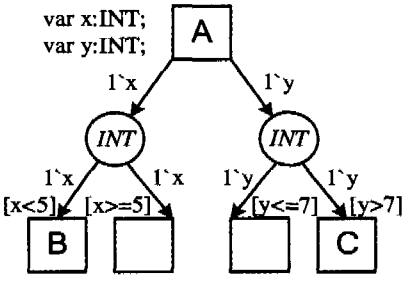

图 6　多重选择模式

模式 7(同步汇聚)

在工作流过程中，多条路径汇聚成一个单独的线程。若多于一条路径被选中，就需要进行多个激活线程的同步，否则无需同步。在同步过程中，一个已被激活的分支不能被再次激活。图 7 给出了 B、C 两个活动进行同步汇聚的示例。可以看到，A 被执行后，B、C 可以被并行执行，也可能只有 B 或 C 被执行，之后 D 被执行；也可能 A 被执行后，到达某一截止期限，B、C 均未执行，则执行 D 对应的时间触发的哑活动，表示取消活动 D。当然也可以设计成手工执行的业务活动。通过布尔型控制数据来限制活动 B、C，可以被执行 0 或 1 次，这是多重选择的第二种实现；而活动 D 只有在 B、C 之一或全部被执行后才能被执行，否则不予执行，这一目的是通过 D 的警卫函数“x or y”达到的。运用该原理，也可以对图 6 中所示的多重选择模式中的分支活动进行汇聚，只是控制数据将包含两个属性：第一个属性为整型，第二个属性为布尔型。若活动 B 执行，则对应布尔型属性值设置为 true；若 B 对应的哑活动执行，则设置该值为 false。活动 C 情况类似。之后的同步过程与图 7 相同，只是活动 D 的哑活动不再需要时间触发条件，因为此时活动 D 及其对应的哑活动不再与活动 B、C 共享输入库所，并不会在活动 A 执行后立即就绪。

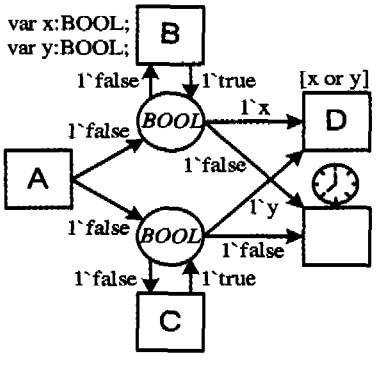

图 7　同步汇聚模式

模式 8(多重汇聚)

在工作流过程中，两个或者更多分支进行非同步汇聚，后继活动为每个到来的分支执行一次，这就需要多重汇聚模式。图 8 给出了对 B、C 两个可被并行执行的活动进行汇聚的示例，活动 D 会为 B、C 对应的每个分支各执行一次。本质上，该模式同简单汇聚模式相同。

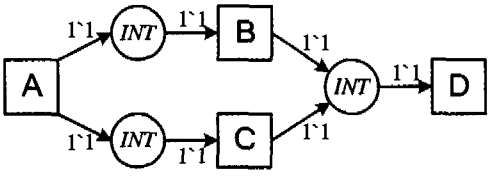

图 8　多重汇聚模式

模式 9(鉴别器)

在工作流过程中，当某个汇聚可能有多于一个输入被激活，等到第一个输入被激活后，后继活动立即就绪，并忽略后面被激活的所有输入，这时会用到鉴别器模式。该模式更通用的情况是：当 m 个分支中的前 n 个输入被激活后($m>=n$)，后继活动才就绪，并忽略后面已被激活或尚未激活的 $m-n$ 个输入。图 9 给出了对 m 个输入分支应用鉴别器的示例。活动 A 被执行后，进入鉴别器模式，活动 $B_1, B_2, \cdots, B_m$ 中 n 个活动被执行后，后继活动 C 可被执行。之后，来自其它输入的控制数据被哑活动逐一消耗掉，完成对剩余输入的忽略任务。若设计人员希望在活动 C 开始执行的同时，取消所有输入分支中正在或尚未执行的活动，则需要结合取消模式。这样，可以更及时地忽略那些对活动 C 的执行没有任何贡献的输入。

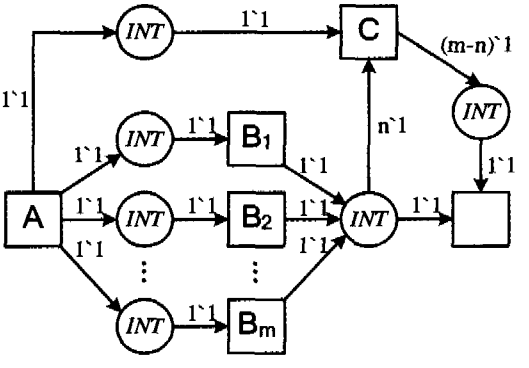

图 9　鉴别器模式

3.3　结构化模式

不同的工作流管理系统对它们的工作流模型强加了不同的限制(如不允许任意循环、只有一个开始节点或结束节点等)，这些限制从建模的角度看并不自然，而且有可能约束业务分析员的定义自由。

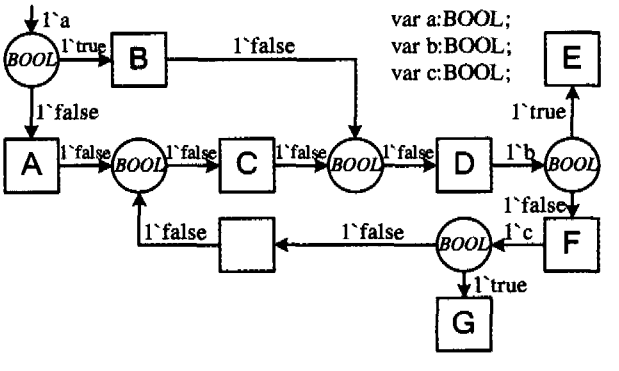

图 10　任意循环模式

模式 10(任意循环)

在工作流过程中，一个或多个活动可被重复执行时，需要使用任意循环模式。图 10 给出了该模式的一个具体示例。

布尔控制数据 a 来自前面的部分，D、F 活动负责获取用户决策或应用计算的结果 b 和 c，三者共同控制循环中选择分支的走向。当 b 的值为 true 时，活动 E 的执行标志着循环的结束；同理，当 c 的值为 true 时，循环结束于活动 G 的执行。

模式 11(隐式终止)

当一个案例无事可做时，该案例应该被终止。换句话说，针对该案例没有被激活的活动，也没有其它活动可能被激活，同时该案例所处的工作流过程没有处于死锁状态。从对隐式终止模式的描述可以看出，隐式终止是针对整个案例而言的，应该由工作流引擎内部自动实现。案例＝状态＋活动，当二者均不存在时，案例理应自动结束。对于基于着色 Petri 网实现的工作流引擎，这个结论是显而易见的。

3.4 含有多重实例的模式

在特定情况下，需要用到下面将要提到的几类多重实例模式。由于设计约束和缺少对这一需求的预测，多数工作流引擎不允许同一活动在同一时刻拥有多个激活的实例。因此，目前绝大多数工作流引擎无法全部实现这类模式。

模式 12(没有同步的多重实例)

对于同一案例，能够创建同一活动的多个实例。即有一种工具来产生新的控制线程，这些控制线程相互独立；此外，没有同步这些线程的需求。图 11 给出了该种模式的一个示例，由活动 A 接收用户在运行时输入的活动 B 可被实例化的次数，活动 B 的所有这些实例可被并行执行，活动 B 前面的哑活动负责自动为其产生足够数目的整型控制数据。

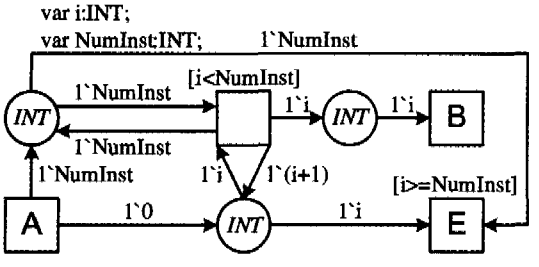

图 11 没有同步的多重实例模式

模式 13(具有先验设计时知识的多重实例)

对于同一案例，同一活动可被执行多次，而且给定活动的实例个数可在设计时知晓。该活动的所有实例都被完成后，其后继活动才能开始执行。图 12 给出了该种模式的一个示例，活动 B 对应的实例个数被限定为 8。活动 A 执行后，活动 B 会被实例化 8 次，之后这些完成的实例同步于活动 C。

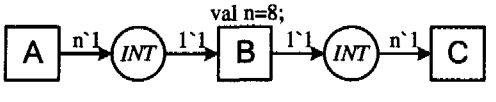

图 12 具有先验设计时知识的多重实例模式

模式 14(具有先验运行时知识的多重实例)

对于同一案例，同一活动可被执行多次，而且给定活动的实例个数是可变的。该个数可能依赖于案例的特征或资源的可用度，但一定会在运行期间该活动对应的实例被创建前知晓。该活动的所有实例都被完成后，其后继活动才能开始执行。图 13 给出了该种模式的一个示例，由活动 A 接收用户在运行时输入的活动 B 可被实例化的次数。活动 A 完成后，活动 B 实例化指定的次数，全部实例完成后同步于活动 C。活动 C 通过绑定控制数据 n，获知活动 B 需要被同步的实例数目。

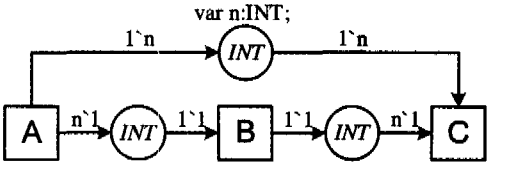

图 13 具有先验运行时知识的多重实例模式

模式 15(不具有先验运行时知识的多重实例)

对于同一案例，同一活动可被执行多次，且给定活动的实例个数在设计时以及运行期间该活动对应的实例被创建前的任何阶段均无从知晓。该活动的所有实例都被完成后，其后继活动才能开始执行。该模式与模式 14 的不同之处在于，即使同一活动对应的多个实例中的一些正在被执行或已经完成，新的实例依然可以被创建。图 14 给出了该模式的一个示例。

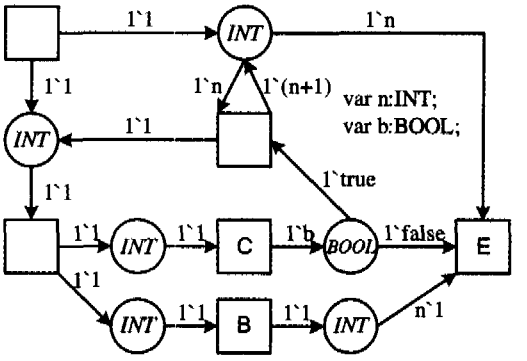

图 14 不具有先验运行时知识的多重实例模式

活动 C 用于让用户指定是否仍然需要继续实例化 B 活动。当 B 的所有实例都完成后，后继活动 E 开始执行。通过一个计数库所，可以很巧妙地获知活动 B 对应实例的最终个数，从而活动 E 可以对所有这些实例进行同步。对此示例稍加修改，就可以让用户自己指定第一次实例化的个数 i 以及每次仍然需要实例化的个数 m，这样活动 B 对应的实例个数的最终计算公式为 $i+m_1+m_2+\cdots$。

3.5 基于状态的模式

在现实工作流中，多数工作流实例处于等待处理的状态而不是正在被处理，因此存在需要显示表达状态的业务场景。Petri 网或着色 Petri 网为显示建模状态提供了完美的解决方案。

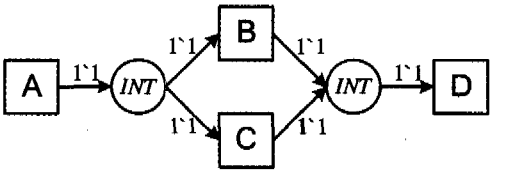

图 15 延迟选择模式

模式 16(延迟选择)

在工作流过程中，由环境决定如何选择多个分支活动中的一个分支时，需要用到延迟选择模式。这些分支活动在某一时刻均就绪，然而一旦其中某个活动被选中执行，其它可选分支活动将不再就绪。图 15 给出了该模式的一个示例。活动 A 被完成后，B、C 两个活动等待其执行者进行选择。B、C

之一被选中执行后，活动D才可以被执行。

模式17(交叉并行路由)

一组活动能够以任意次序被执行：组内每个活动都会被执行，次序在运行时决定，且没有任何两个活动会同时被执行。图16给出了该模式的一个具体示例，其中B、C两个活动的执行模式满足上述要求，通过为这两个活动增加一个共享的输入库所作为互斥信号量来达到这一目的。由活动A负责为信号量放置初始控制数据，完成控制作用后，由负责同步的活动D移除该信号量中的控制数据。

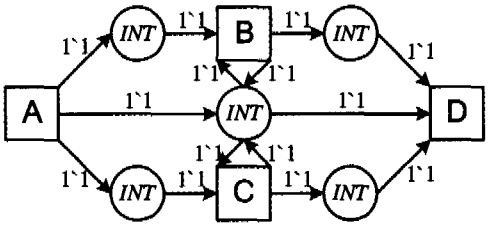

图16 交叉并行路由模式

模式18(里程碑)

一个活动的就绪依赖于案例处于某特定的状态，即该活动只有在一个特定的里程碑已经达到而且还没有过期的情况下才能就绪。如图17给出了该模式的一个示例，活动A只有在活动B已经被执行，且活动C还没有被执行时才能就绪，即活动A在活动B被执行前以及活动C被执行后都不会就绪。这里，活动B、C之间的那个库所就是活动A的里程碑。只有当该库所中存在相应控制数据时，活动A才可以就绪。

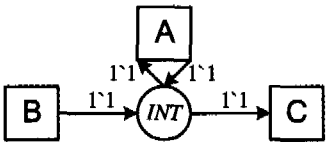

图17 里程碑模式

3.6 取消模式

事实上，下面两个分支可以直接合并为更通用的模式，即取消活动区域模式。活动区域由设计人员从过程中手工选定。

模式19(取消活动)

一个就绪的活动被废除，即一个活动的等待执行线程被中断。如图18给出了该模式的一个示例，活动A执行后，活动B和其哑活动均就绪，哑活动的执行表示活动B的取消。若两个活动可被并行执行，且活动B已经开始执行，则需要采用内部应用调用方式，为该哑活动设置用于取消活动B的应用。这样，当哑活动执行时，引擎会自动执行该应用取消活动B。若是否取消活动B的决策由用户决定，则应将该哑活动改为实际的业务活动。

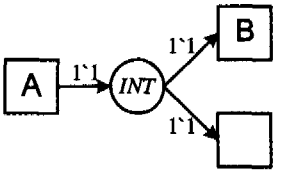

图18 取消活动模式

模式20(取消案例)

一个案例被完全移除，包括被实例化多次的活动以及嵌套子流程中的活动。取消案例可作为工作流引擎服务和内部调用应用来实现，可将该应用分配给任意活动，用来实现取消整个案例。如图19所示，活动B是过程的第一个活动，活动E是过程的最后一个活动，活动X被分配了取消案例的内部应用。当活动X被执行时，引擎会自动执行取消案例应用来取消该活动所属案例。

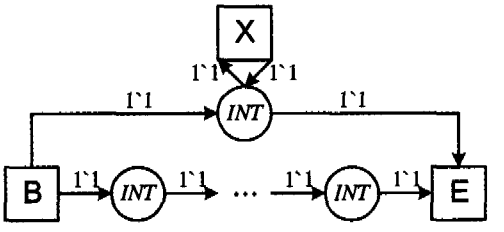

图19 取消案例模式

总结与展望 通过对工作流模式的支持情况可以看出，基于着色Petri网的工作流建模语言在建模工作流模式时，具有简洁、准确的特点。同某些专门为工作流模式设计的工作流建模语言相比，采用着色Petri网建模过程控制流虽然会给过程设计人员带来一定的负担，但却换来了较高的建模灵活性和模型可扩展性。本文提出的建模方案已在实际工作流管理系统当中做了具体的实现和反复验证，证明了该工作流建模语言具有很强的表达能力和很好的适用性，同时证实了建模方案的可行性。同时，本文为如何使基于着色Petri网的工作流引擎实现对全部20种工作流模式的支持提供了指导。不基于着色Petri网的工作流引擎，进行等价转化后，同样可以使用该方案。

未来工作将主要集中在以下两个方面。首先，需要针对特定的工作流模式实现进行抽象，提炼出可复用的通用模式基本构造，设计人员只需要输入一些简单的参数，即可以得到用着色Petri网表达的具体模式实现。其次，进一步推广工作流管理系统在组织当中的应用，深入挖掘组织业务，力争总结或发现更多的符合组织实际业务需要的工作流模式。

参 考 文 献

1 袁崇义. Petri网原理与应用. 北京：电子工业出版社，2005

2 Jensen K. Coloured Petri Nets. Berlin：Springer-Verlag，1997

3 van der Aalst W M P，ter Hofstede A H M，et al. Workflow Patterns. Distributed and Parallel Databases，2003，14(1)：5～51

4 van der Aalst W M P，ter Hofstede A H M. YAWL：yet another workflow language. Information Systems，2005，30(4)：245～275

5 Moldt D，Rölke H. Pattern Based Workflow Design Using Reference Nets. van der Aalst W M P，et al. eds. BPM 2003，LNCS 2678. 246～260

6 余鹏，周国富，屈婉玲，等. 基于Petri网的工作流模式—工作流模式分析研究. 系统仿真学报(增刊)，2003，15(S1)：119～122

7 龚晓庆，葛玮，郝克刚. 信牌驱动模型中的工作流模式. 西北大学学报(自然科学版)，2004，34(1)：13～18

8 陈大锋，吴泉源，刘江宁，等. 一种多实例工作流模式的解决方案. 计算机工程与科学，2005，27(1)：89～92