**BPM领域的流行的21种模式**

目录

[BPM领域的流行的21种模式与AWCP的支持对比 2](#_Toc424830607)

[5种基本控制流模式 2](#_Toc424830608)

[Pattern1 顺序流（Sequence） 2](#_Toc424830609)

[Pattern2 并行分叉（Parallel Split） 5](#_Toc424830610)

[Pattern3 同步（Synchronization） 6](#_Toc424830611)

[Pattern4 独占式选择（Exclusive Choice） 10](#_Toc424830612)

[Pattern5 简单聚合（Simple Merge） 12](#_Toc424830613)

[4种高级分支同步模式 12](#_Toc424830614)

[Pattern6 多重选择（Multiple Choice） 12](#_Toc424830615)

[Pattern7 同步聚合（Synchronizing Merge） 16](#_Toc424830616)

[Pattern8 多重聚合（Multiple Merge） 17](#_Toc424830617)

[Pattern9 鉴别器（Discriminator） 17](#_Toc424830618)

[2种结构化模式 18](#_Toc424830619)

[Pattern10 任意循环（Arbitrary Cycles） 18](#_Toc424830620)

[Pattern11 隐式终止（Implicit Termination） 19](#_Toc424830621)

[2种结构化模式在JBPM4中的综合实现 20](#_Toc424830622)

[4种包含多实例的模式 21](#_Toc424830623)

[Pattern12 无同步的多实例（MI without） 22](#_Toc424830624)

[Pattern13 设计时确定的多实例（MI with a Priori Design Time Knoledge） 23](#_Toc424830625)

[Pattern14 运行时确定的的多实例（MI with a Priori RunTime Knoledge） 24](#_Toc424830626)

[Pattern15 运行时无法确定的的多实例（MI without a Priori RunTime Knoledge） 24](#_Toc424830627)

[3种基于状态的模式 25](#_Toc424830628)

[Pattern16 延迟选择（Deferred Choice） 25](#_Toc424830629)

[Pattern17 交叉存取并行路由（Interleaved Parallel Routing） 26](#_Toc424830630)

[Pattern18 里程碑（Milestone） 29](#_Toc424830631)

[2种取消模式 29](#_Toc424830632)

[Pattern19 取消模式（Cancel Activity） 29](#_Toc424830633)

[Pattern20 取消实例（Cancel Case） 30](#_Toc424830634)

[具有中国特色的工作流模式 31](#_Toc424830635)

[Pattern1 回退（Reject） 31](#_Toc424830636)

[定义与描述 31](#_Toc424830637)

[AWCP的模式 31](#_Toc424830638)

[Pattern2 取回 32](#_Toc424830639)

[定义与描述： 32](#_Toc424830640)

[AWCP的支持： 32](#_Toc424830641)

[Pattern3 委托代办 32](#_Toc424830642)

[定义描述： 32](#_Toc424830643)

[AWCP的授权 33](#_Toc424830644)

[Pattern4 催办 Escalation 35](#_Toc424830645)

[定义描述 35](#_Toc424830646)

[AWCP的支持 35](#_Toc424830647)

[Pattern5 自由流 35](#_Toc424830648)

[定义与描述 35](#_Toc424830649)

[AWCP的支持： 36](#_Toc424830650)

[Pattern6 会签 37](#_Toc424830651)

[定义与描述 37](#_Toc424830652)

[AWCP的定义： 37](#_Toc424830653)

[总结说明 40](#_Toc424830654)

[概述 40](#_Toc424830655)

[AWCP控制流程运行模式的方式： 40](#_Toc424830656)

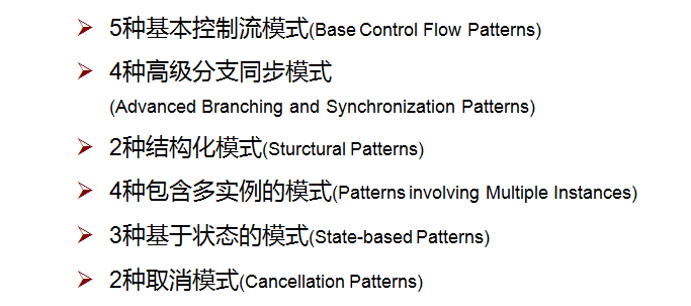
[AWCP 的4大流程运转模式的解释 41](#_Toc424830657)

# BPM领域的流行的21种模式

本文件以国外流行的20中流程模式的设计与jbpm4的模式案例与当今中国开源的AWCP流程引擎对照，以便让各位评估人员能够了解到中国国情的工作流引擎与估计流行的设计规则的差别、不同、与优缺点。以便各位能够做出正确的AWCP与jbpm评估。

为了方便预览，请使用文档结构图的模式进行查看。

在BPM领域，比较流程通用的就是满足如下21中流程模式的支持，如下图：



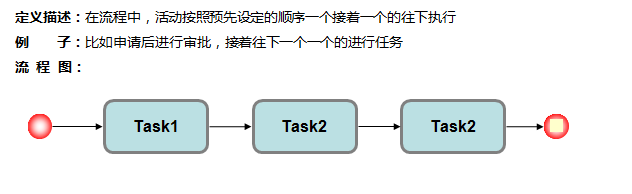
下面的章节里，我们对这写模式进行一个个的对比，以找出AWCP与jbpm的差异。

## 5种基本控制流模式

### Pattern1 顺序流（Sequence）

#### 基本概念

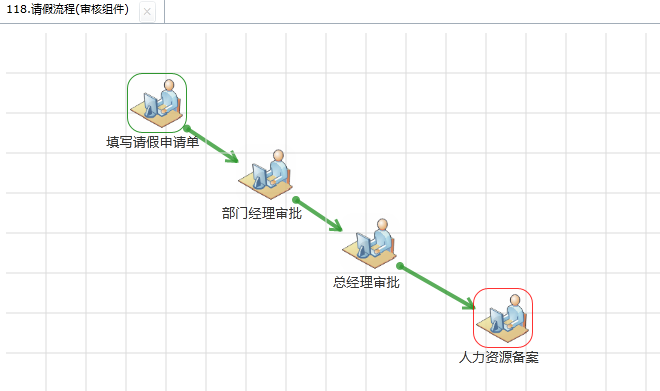
就是按照流程设计的步骤，一步步的向下运行，这样的模式下每个节点有先后顺序，就是每个节点只有一个节点是活动的，如下图：



#### AWCP的支持

这个实际上在AWCP上来看他是不同的线性流程，也叫顺序流程，没有分支，流程一般在最后一个节点停止，并标识该流程完成，AWCP允许设置节点的完成条件，与流程的完成条件，当符合该条件时候，流程自动结束。

在AWCP的demo流程中，如下流程符合这种模式：



这样的流程模式属于最基础的流程模式，AWCP的特点是：允许用户自己定义流程完成条件，在任何一个节点运行过程中，AWCP都要去检查条件设置，如果满足这个条件流程就自动结束。AWCP的流程完成条件设置相对简单实用、直观。

**条件类型：**



**按照表单条件：**



对于条件的设置规则，请参考AWCP的操作手册，条件规则的设置，在以后很多的场景都用到。

#### AWCP特点

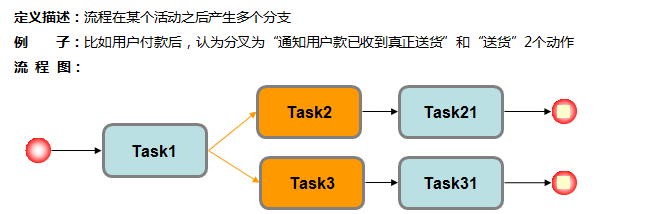
1, 支持条件的图形化配置，让其在指定的节点自动结束。

2，条件的配置支持多种方式，因为AWCP与szcloud是集成的，所以AWCP在流程运转中可以操纵szcloud的数据，让配置更容易，生动，直观。

3， AWCP还支持流转自定义功能，就是说在指定的节点上，可以自己定义以后节点运行的顺序与执行人。 请参考：http://AWCP.mydoc.io/?v=5404&t=17978

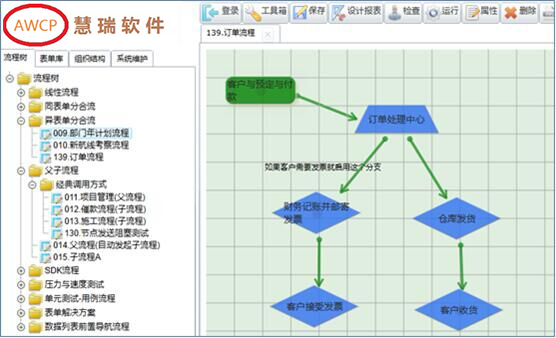
### Pattern2 并行分叉（Parallel Split）

#### 基本概念



#### AWCP的支持

这个属于AWCP 异表单分合流，一个动作结束后，并行启动多个分支，每个分支都要向下运动, 每个分支运行完后自动该分支工作完成，由最后一个分支的完成来决定整体的流程完成。



**方向条件的设置：**如果客户需要发票，就可以启动这个分支。



#### AWCP的特点

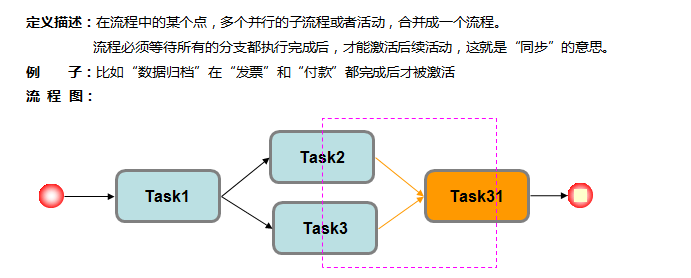
1. AWCP的多种方式的规则设置、支持条件的优先级计算，满足个流程运转的配置需要。

请参考说明书：[http://AWCP.mydoc.io/?v=5404&t=20865](http://jflow.mydoc.io/?v=5404&t=20865)

2，支持多个条件的组合计算。

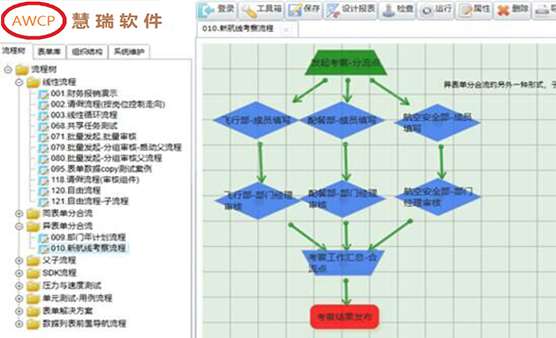
### Pattern3 同步（Synchronization）

#### 基本概念



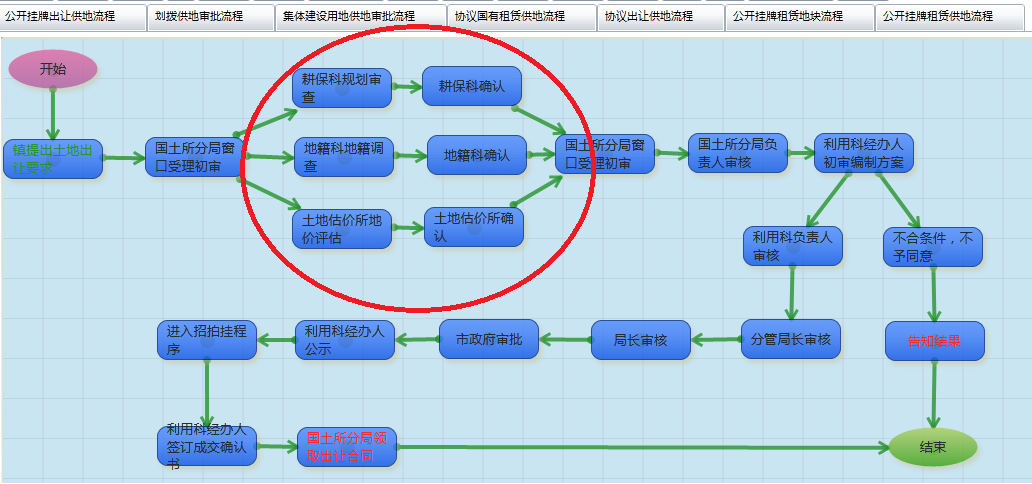
#### AWCP的支持

这种模式对应的是AWCP的分合流：为了让用户更容易配置，与理解AWCP根据同步节点处理的内容是否相同，分为同表单分合流，与异表单分合流。以上的模式是异表单的分合流，因为task2,task3所处理的工作是不一样的，他类似于AWCP流程案例库中的飞行考察流程。

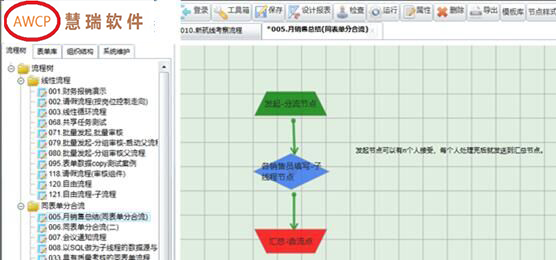


在最后的合流节点里汇总的是多个表单.

也类似于国土行业的流程。



但是在同步流程中，我们扩展了同表单的分合流，因为这样的应用模式很多，并且需要数据汇总，比如：销售经理下达任务要求每个人汇报上个周的销售总结，销售经理汇总销售清单，如下流程图：

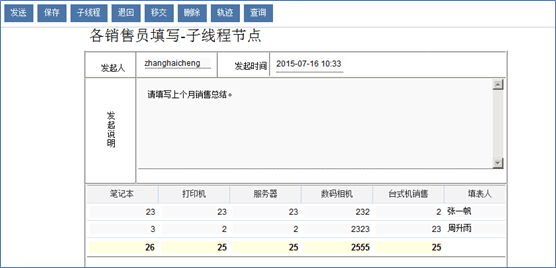


**流程运行效果如下：**

第1个点表单：



第2个点表单：每个子线程同时填写.



第3个点表单：得到汇总数据。

#### AWCP特点

1, 根据子线程节点，填写的内容是否相同，支持同表单分河流，异表单分合流。

2，AWCP可以操纵szcloud数据进行汇总，完成复杂的业务数据操作。

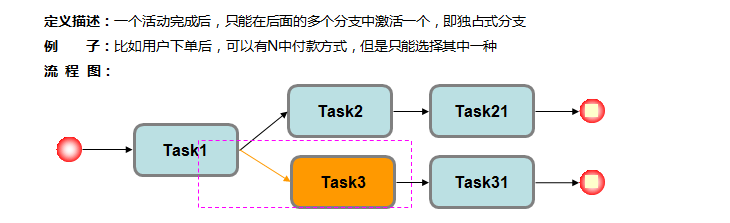
3，AWCP有合流节点完成率的概念，可以指定一定的完成率，才能到达合流节点。

4,对于未完成的子线程，AWCP有子线程的删除规则，满足不同情况下的对子线程的删除操作。



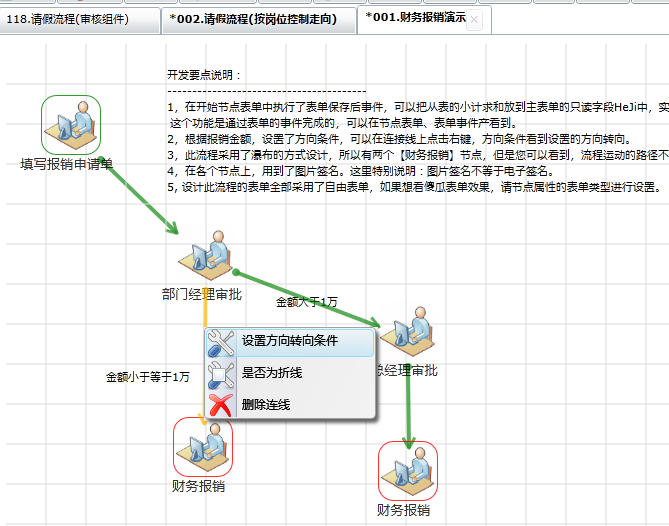
### Pattern4 独占式选择（Exclusive Choice）

#### 基本概念

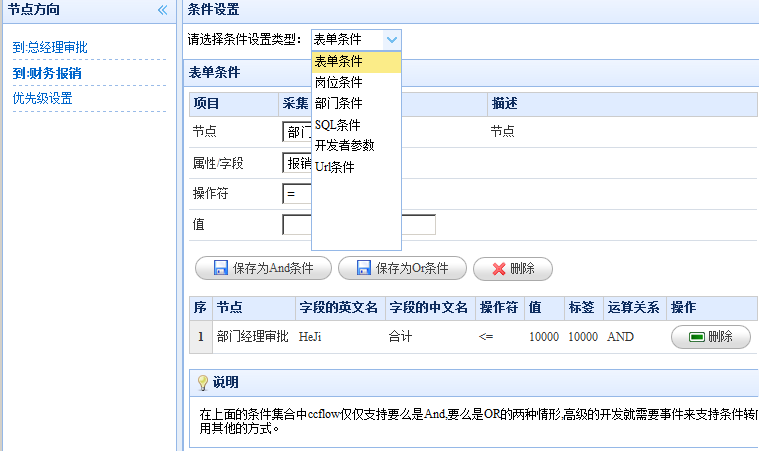


#### AWCP的支持

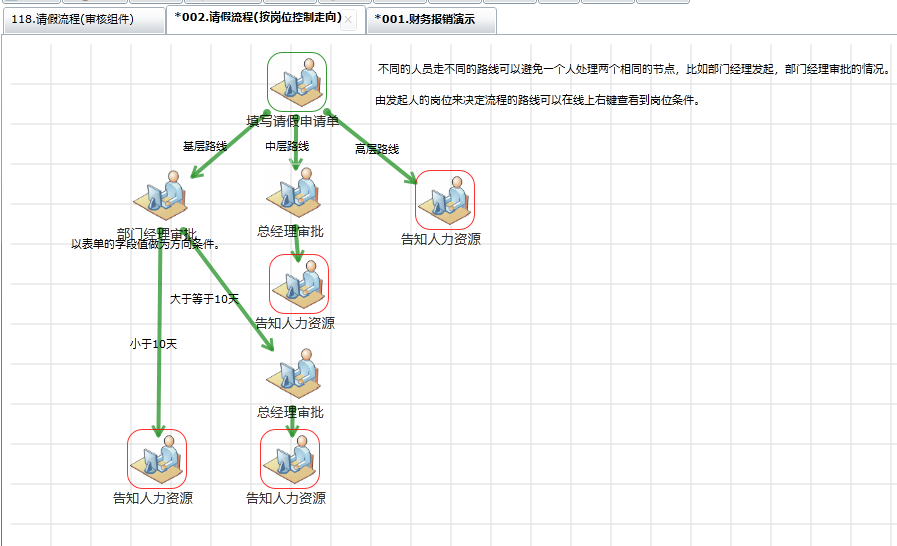
**解释：**流程在运行的过程中，根据不同的条件进行不同的走向，在AWCP看来这样的也属于线性流程，就是在一个节点是出现了分支，比如：



当金额小于 1万的时候，直接报销。大于1万的时候，需要走总经理审批，这是一种使用表单的字段作为一个判断条件，AWCP的特点是支持多种方向条件的判断，比如：按照表单字段值，按照当前人的岗位、当前人的部门，按照url，按照表达式等等，满足多种场合下的配置需要，并且支持多个条件的组合。



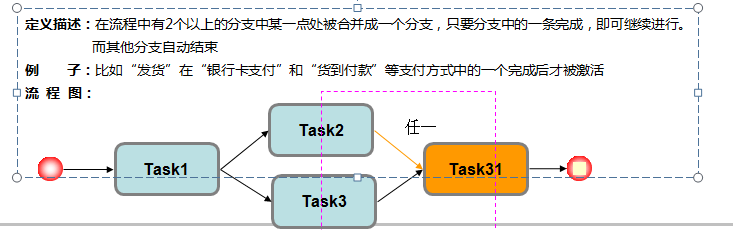
**支持优先级计算：**比如如果分支的条件有多个的话，同时满足，如何处理优先级判断的问题。



#### AWCP的特点

### Pattern5 简单聚合（Simple Merge）

#### 基本概念



#### AWCP的支持：

这种行为，属于AWCP的异表单的分合流概念，在AWCP中，有一个完成率，来解决此问题。

#### AWCP的特点

这仅是一个简单的分支，是否可以起用，是通过条件规则，子线程的删除条件规则来确定的。

AWCP的完成通过率概念：

[http://AWCP.mydoc.io/?v=5404&t=17945](http://jflow.mydoc.io/?v=5404&t=17945)

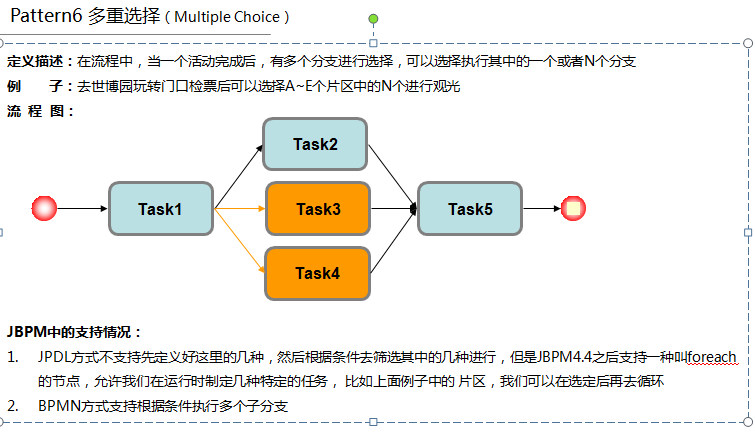
对于子线程（子任务）的删除操作规则，请参考：

[http://AWCP.mydoc.io/?v=5404&t=17992](http://jflow.mydoc.io/?v=5404&t=17992)

## 4种高级分支同步模式

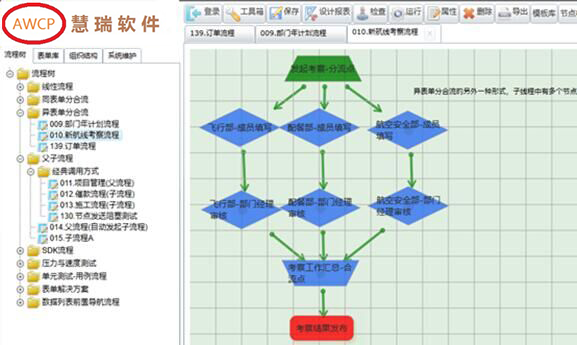
### Pattern6 多重选择（Multiple Choice）

#### 基本概念



#### AWCP的支持

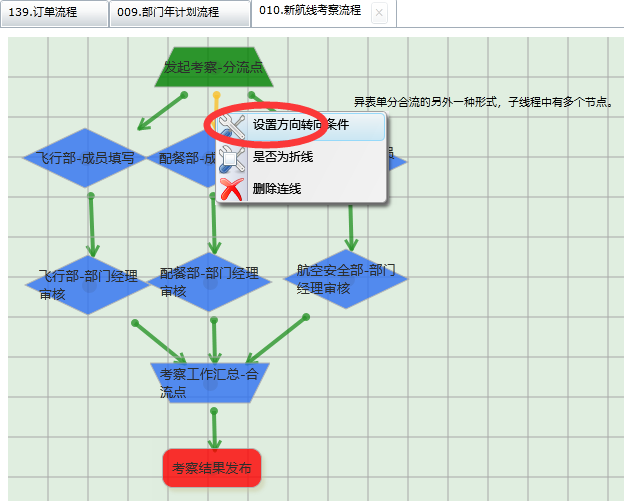
这种类似于异表单的分合流的模式, 我们看看如下应用场景。



发起流程表单：



条件设置



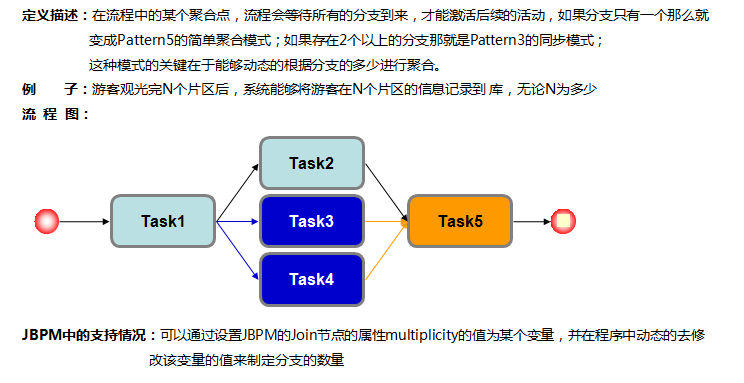


#### AWCP的特点

1. 概念少。
2. 配置简单，可以任意的启动指定的节点。

### Pattern7 同步聚合（Synchronizing Merge）

#### 基本概念



#### AWCP的支持

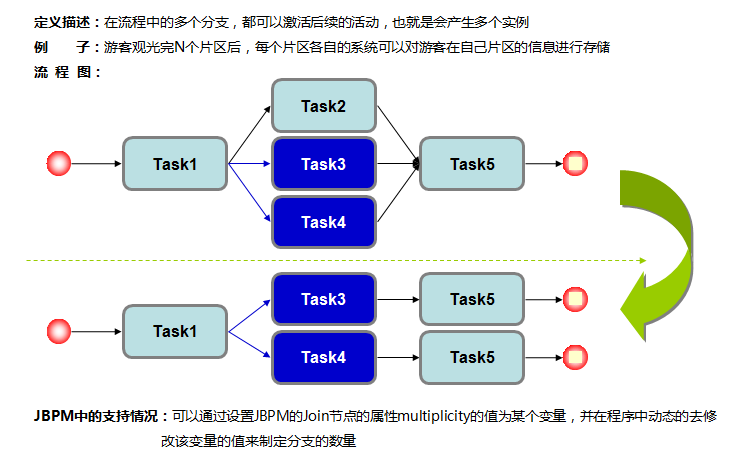
与AWCP的异表单的分合流相同， 解释说明请参考pattern 5 解释。

#### AWCP的特点

与AWCP的异表单的分合流相同， 解释说明请参考pattern 5 解释。

### Pattern8 多重聚合（Multiple Merge）

#### 基本概念



#### AWCP的支持

与AWCP的异表单的分合流相同， 解释说明请参考pattern 5 解释。

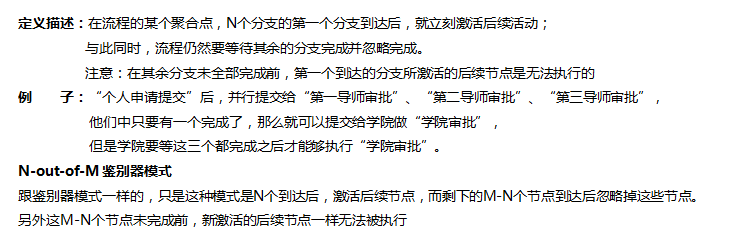
AWCP有workid与fid两个概念，一个是工作ID,一个是流程ID,他们来交替控制整个流程实例与每个子线程实例的状态。

#### AWCP的特点

与AWCP的异表单的分合流相同， 解释说明请参考pattern 5 解释。

### Pattern9 鉴别器（Discriminator）

#### 基本概念



#### AWCP的支持

在AWCP中，有两个属性的控制，可以实现该功能。

第1个：子线程完成率。 该规则可以决定是否可见

第2个：子线程删除规则。该规则决定那些子线程可以被删除以及他们的删除方式。

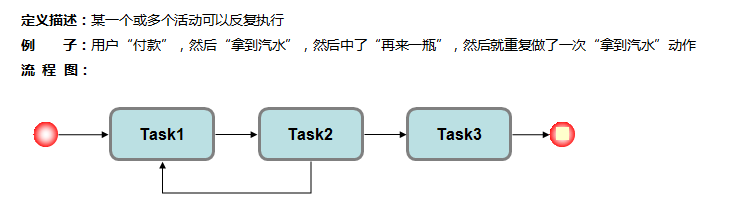
#### AWCP的特点

简单、可靠。

## 2种结构化模式

### Pattern10 任意循环（Arbitrary Cycles）

#### 基本概念



#### AWCP的实现

类似于AWCP的demo中的稿件发布流程。



如果稿件符合要求，就跳出循环否则一直在循环里修改该文件。

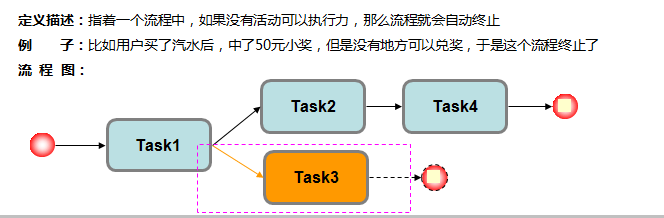


#### AWCP的特点

完全是条件判断，在表单中增加一个审核组件，就可以把每次校验的信息，写入里面，完整的显示出来整个轨迹。

### Pattern11 隐式终止（Implicit Termination）

#### 基本概念



#### AWCP的实现

这种类型属于AWCP的线性流程的一种，该流程配上流程完成条件，就可以实现该功能。

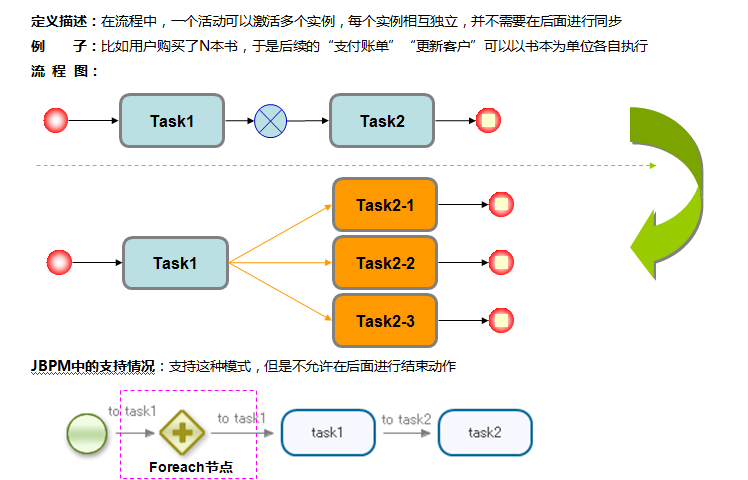
流程完成条件，就是流程在前进中检查的条件，如果满足该条件，流程就停止运行，该流程实例结束。

#### AWCP的特点

流程完成条件：配置方式有多种表达式，根据不同的场景可以配置不同的条件。从这一点可以看出21中模式在AWCP中的一些属性的配置就可以轻松实现。

### Pattern12 无同步的多实例（MI without）

#### 定义与概念



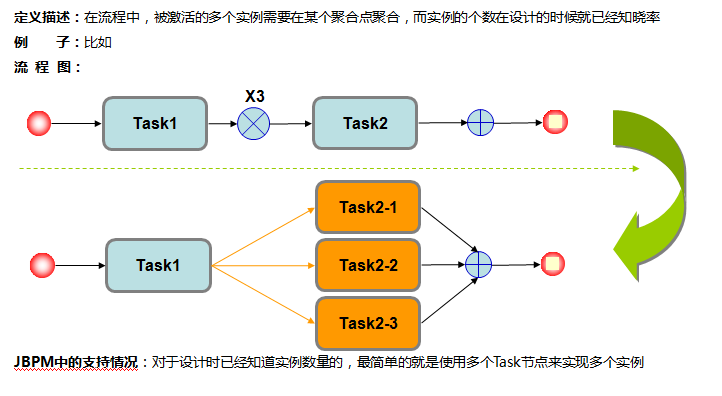
#### AWCP的模式实现

AWCP 有两种方式可以实现，第一种是父子流程，第2中是分合流。这两种模式都可以实现该种效果，但是不同的模式选择不同的场景。

对于买书这个事例使用分合流更贴切一些实现, 分合流中一次购买是一个主线程，多次付款就是子线程。

### Pattern13 设计时确定的多实例（MI with a Priori Design Time Knoledge）

#### 定义与概念



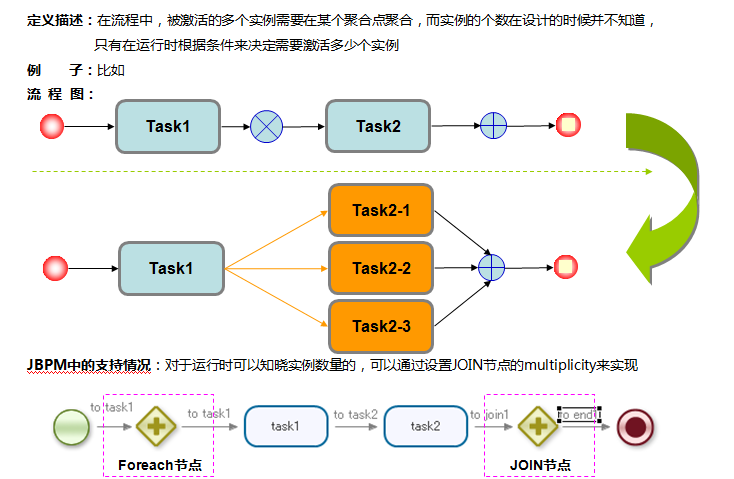
#### AWCP的模式

这属于AWCP的典型的合流节点处理各个子线程的任务比率。

**概念：**完成率 = 子线程上已经完成的数据/所有子线程数量\*100%

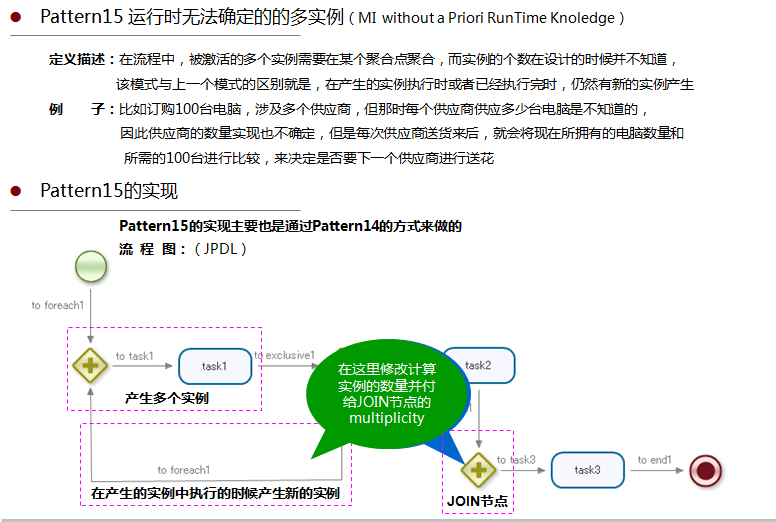
**说明：**该节点对于合流节点与分合流节点有效，当子线程的完成率达到该值的时候，该节点的待办才能显示出来，否则该节点的人员不能处理待办。如果合流节点的处理人能够看到待办，他就可以对该流程进行操作，比如：发送、删除、退回、删除子线程等等。

### Pattern14 运行时确定的的多实例（MI with a Priori RunTime Knoledge）



### Pattern15 运行时无法确定的的多实例（MI without a Priori RunTime Knoledge）

#### 定义与概念



#### AWCP的实现

这种方式属于AWCP的父子流程来实现，开始节点启动一个任务，需要采购100台电脑，需要发起n此的选择供应商采购的子流程，每个子流程完成后，就访问父流程节点信息，进行相关的业务处理(就是是否启动下一个子流程，如果满足100，就不启动子流程了，直接完成父流程的任务，结束主流程.)。

## 3种基于状态的模式

### Pattern16 延迟选择（Deferred Choice）

#### 定义描述

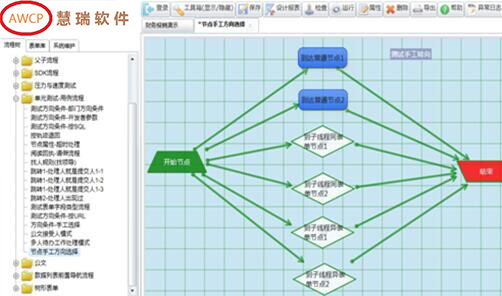
**定义描述：**流程中某个点可以有多个分支进行选择。

**与Pattern4相比较：**不是基于简单的数据或者决定就可以很明显地作出选择，而是会向系统或者执行环境提供多种可选择的分支；但是又不同于AND-Split模式，延迟选择只能选择一个分支执行，一旦选择了其中第一个分支，那么其他分支就会被撤销。这种延迟一直会持续到第一个选择分支开始实际运行。

**例 子：**收到的一批商品运送到各个部门，到底选择什么样的运行方式，要看资源的可用性

#### AWCP的模式：

AWCP模式支持方向条件的手工选择，有当前节点的操作人员来决定流程要发送到什么地方去。 详细文档：[http://AWCP.mydoc.io/?v=5404&t=17992](http://ccbpm.mydoc.io/?v=5404&t=17992)

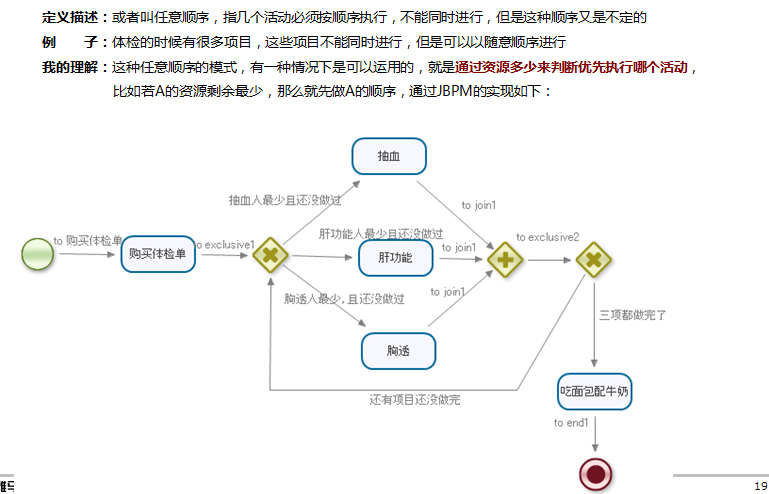


**运行效果：**



### Pattern17 交叉存取并行路由（Interleaved Parallel Routing）

#### 概念与定义



#### AWCP处理的模式

这是典型的一种多任务分配流程，使用多维度的分合流可以实现，这个流程的特点是，一个操作人员可以处理三个不同的任务，这三个任务属于三个子线程，与普通的分合流不同的是这三个子线程是同一个人处理。在这里就决定了，有一个批次号(项目维度)，在这里就是检查项目。

在接收人规则里：AWCP对合流的节点处理人有一个这样的规则约定，下面是该文档的描述：

**关于合流点的接受人按sql获取接受的表达式的问题**

注意子线程向合流点发送时，接受人规则的表达式的变量是临近合流点的子线程节点变量。

比如: 流程编号为ABC三个节点.

A 是分流点， C是合流点 C是子线程。

如果C的接受人员规则是按sql计算:

配置的表达式如下表达式是错误的:

select UserNo as No, xx as Name from ND2701 WHERE OID=@OID

如下表达式才是正确的:

select UserNo as No from ND2701 WHERE OID=@FID

这是因为子线程在发送时获取的变量OID 是子线程的ID而非，干流上的WorkID.

**关于子线程接受人的特殊约定：**

如果遇到分组的维度,就约定返回4个列来解决问题，流程demo:\流程树\同表单分合流\一人多子线程模式(批次维度任务模式)流程.

在第2个子线程节点配置了如下SQL。



该数据源返回了三个列，分别是：No,Name,BatchNo。 No=操作员编号，Name=操作员名称，BatchNo批次编号。

**注：这里的批次号，就可以标识了检查项目。**

### Pattern18 里程碑（Milestone）

#### 定义概念

**定义描述：**一个活动的激活需要一种具体的状态，比如活动A，B，C，只有在AB都执行完了的情况下才能执行C

**我的理解：**这种模式有点类似于顺序模式？或者是同步模式？

#### AWCP的实现

这个可能类似与AWCP的延续流程，当一个流程长度很大的时候，需要跨度很多年实施的时候，把该流程截成一段段的，分开设计，一条流程，是上一条流程的延续。

延续流程是父子流程的一种，但是延续流程只有一个段接一段，就是一个父流程有多个子流程，但是延续流程就只能有一个子流程。

## 2种取消模式

### Pattern19 取消模式（Cancel Activity）

#### 定义与描述

**定义描述：**简单的理解，就是将某个活动取消

**例 子：**比如下好订单后，“支付货款”活动正在运行中，此时用户取消订单

个人理解：取消某个活动，会引起几种后续情况：

1. 整个流程结束；比如订单取消后，订单的这个流程结束。

2. 退回上一个节点；比如付款后，用户取消付款，重新下订单。

#### AWCP的模式：

此模式与AWCP的流程删除相同，流程删除是有删除规则的，不同的规则应用于不同的删除场景。在线文档：http://AWCP.mydoc.io/?v=5404&t=17992

**应用背景**： 流程启动运行后，在一些节点当前处理人有权利执行流程终止，把当前的流程实例进行删除，根据不同的应用场景删除的处理方式有很多种，每种删除方式都对应AWCP相关的api操作，下面我们把这几种删除方式说明一下，以方便让设计人员根据实际的规则来选择不同的方式。

**不能删除**: 该流程不允许用户删除.

**逻辑删除**：仅仅做上删除标记, 数据仍然在节点表单与流程报表里面.

**记录日志方式删除** : 删除节点表数据，流程报表数据，并记录到 wf\_workflowDeleteLog 表里作备案.

**彻底删除** : 彻底清除该流程的所有数据，包括该工作实例的节点表单数据、流程报表数据、轨迹数据、退回、移交操作信息。

**让用户决定删除方式** : 显示出对话框，让用户选择上面的删除方式.

**其他**：删除规则的API请查看代码: BP.WF.Dev2Interface.Flow\_DoDeleteFlowBy \*

1， 开始节点如果设置了删除按钮，如果不是退回来的节点，或者没有发送过的节点，就抛出异常。

### Pattern20 取消实例（Cancel Case）

**定义描述：**如果一个活动产生了多个实例，那么仅仅撤销这个活动是不够的

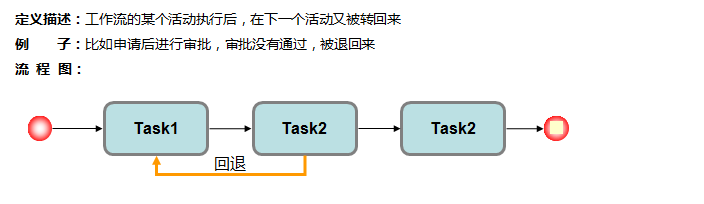
要将他所引起的所有实例都移除才行

**AWCP的规则：**此规则是 Pattern19的流程删除实例的一种情况。

# 具有中国特色的工作流模式

## Pattern1 回退（Reject）

### 定义与描述



### AWCP的模式

AWCP的退回，是由他的退回规则来确定的，他的退回规则如下：

在线帮助文档：[http://AWCP.mydoc.io/?v=5404&t=17968](http://jflow.mydoc.io/?v=5404&t=17968)



## Pattern2 取回

### 定义与描述：

**定义描述：**流程由A运转到B，在B还没有完成任务前，A的执行者可以收回任务，重新执行

**例 子：**比如科员提交了一份文件给处长审批，处长还没有签字前，科员可以自己拿回来重新修改

### AWCP的支持：

这里的取回，相当于AWCP的撤销，详情请参考：

[http://AWCP.mydoc.io/?v=5404&t=17919](http://ccbpm.mydoc.io/?v=5404&t=17919)



## Pattern3 委托代办

### 定义描述：

工作流的某个活动指派人由于某种原因无法完成任务，但是可以手动指定代办人

### AWCP的授权

授权与授权方式登录，是在AWCP中可以设置的。

授权就是，我把权利授权给指定的人，授权方式登录，就是我需要把权利授权给指定的人。

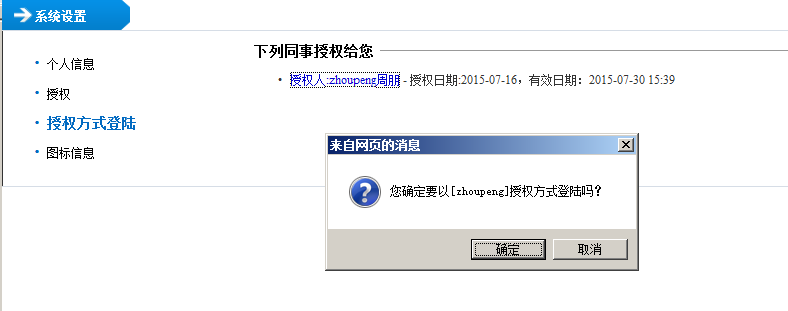


授权，有授权范围，如下图标识了三种授权范围：

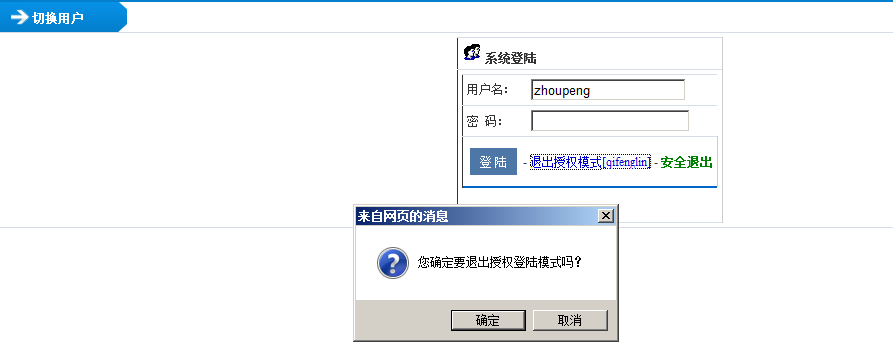




以qifenglin进来，进入授权方式登录：



退出授权登录模式



## Pattern4 自由流

定义描述: 能够自由人为的控制流程的流向

### AWCP的支持

**定义：**在特定的步骤，当前的工作处理人员需要跳转到指定的节点上去，这种应用场景叫跳转。跳转是在不规范的情况下，或者特定的应用场景下的需求，当前节点与被跳转到的节点之间可以无连接线，可以跳转的节点选择范围是有当前节点的跳转规则来决定的。点击如下图的跳转按钮，就转到跳转界面上来。

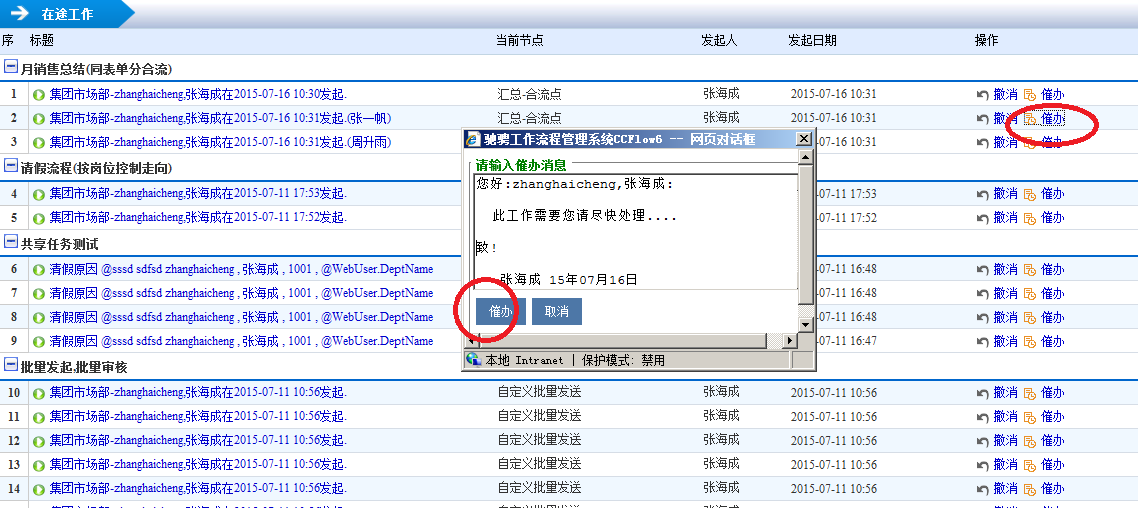


## Pattern5 催办 Escalation

### 定义描述

能够设置要求，如果某活动中某个时间内还未被完成，将进行通知等活动，甚至是修改活动的执行者

### AWCP的支持



催办之后AWCP是写入了消息表 sys\_sms ，这个表的数据用于发送邮件，微信、或者短信提醒。

## Pattern6 运行期自定义流程

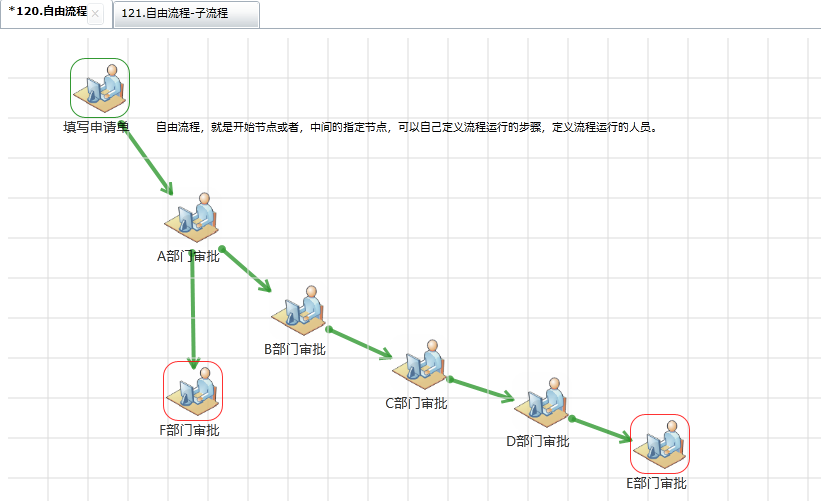
### 定义与描述

**定义描述：**能够自由人为的控制流程的流向。

**例 子：**比如设计时定义好的流程A-B-C，但是完成了A以后，认为的将流程变为A-C，直接将B给忽略掉.

### AWCP的支持：

指定的节点可以定义未来流程的走向，与指定的节点的处理人，让流程运转的控制路径，在当前节点可以自己定义，也可以插入一条指定的子流程，这就是AWCP的自由流程的概念。在线帮助文档：http://AWCP.mydoc.io/?v=5404&t=17978



流转自定义界面：



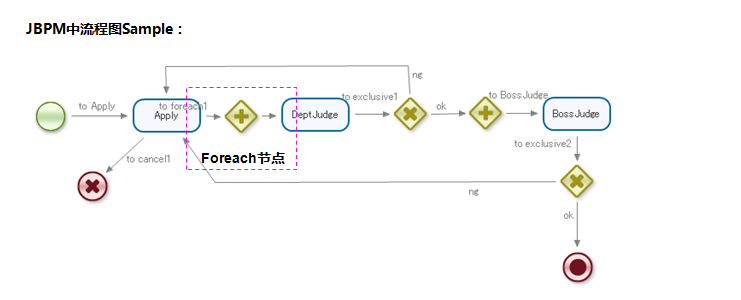
## Pattern7 会签

### 定义与描述

**定义描述：**会签是撰拟公文的过程中，主办单位主动与有关单位协商并核签的一种办文程序，一般当公文的内容涉及本单位的多个部门或与其它单位有关时，需要进行会签。

**例 子：**比如大学生毕业离校，需要在不同的部门机构进行签字，全部审核通过后才能离开学校。但是签字的先后顺序可能并没有指定。

**问 题 点：**问题在于如果会签过程中有一个部门审核不通过，其他已经签过的部门是否需要重新审核？



### AWCP的定义：

在AWCP里是多人处理规则，就是一个节点多个人都可以处理，根据业务场景不同，AWCP的多人处理规则如下，在线帮助文档描述的更清楚：

[http://AWCP.mydoc.io/?v=5404&t=17947](http://ccbpm.mydoc.io/?v=5404&t=17947)



根据不用的场景，AWCP把多人在普通节点下的处理模式分为如下几种。

**抢办模式**:

A发送到B ，B节点上有n个人可以处理。这n个人都可以看到待办，当其中一个人处理后，其他人的待办就消失了。这样的工作模式属于抢办，这n个人可以同时打开，当一 个人发送后，其他人都不能在发送了。通俗的说，也就是谁抢到了这件工作，就是谁处理的。抢办模式是一个默认的处理模式。

**协作模式**:

A发送到B ，B节点上有n个人可以处理。这个n个人都可以去处理，都可以去发送，有最后一个处理人发送到下一个节点上去，这n个人的处理不分顺序。这样的模式我们叫做协作模式。

协作模式通常用在具有审核组件的表单中，每个人都可以填写自己的意见，发表自己的看法，在审核组件里审核每个人在一个节点上只能写入一条数据，也就是说只能表达一次意见。

也经常用在传阅的情况下，必须这些人都读取了，知道了才能向下运动。它与抄送不同的地方是，抄送可以不看，就可以向下运行，但是他必须处理才能向下运行。

通常的协作模式，就是多人都要点击一下发送功能按钮，来确认自己处理了，确认自己同意了或者认可了。

**队列模式**:

A发送到B ，B节点上有n个人可以处理。这个n个人按照设置的先后顺序去处理，在同一个时间点必须有一个人待办，其他人看不到。

应用场景: 一个公文发送给一个部门，该部门下n个人可以接受（注意这是一个节点），按照国内的行政制度，这n个人会从小到达排一个队列，有最小的级别的工作人员审 批，然后依次类推，走到最后由最后一个人发送到下一个节点去。这n个人都是该节点的处理人，这n个人的审核步骤是由他们的顺序确定的，所以在设置该节点的 接受人时，要按照顺序执行。

通常这样的模式也与审核组件一起使用，每个人表达意见后，就发送到下一个节点，当然这样的模式也

**共享模式**:

A发送到B ，B节点上有n个人可以处理。AWCP有一个任务池，这写人都需要从这个任务池里获取。

　　关于对AWCP任务池的描述：

　　１，在流程属性与全局配置设置了启动任务池的状态下，如果一个节点遇到多个人处理，这个任务就放在了任务池里，只有参与处理的人人员才能看得到。

　　２，工作人员在处理该工作之前需要申请，放到自己的代办里，一人申请之后，其他人就不能申请了，多个人同时申请第一个人申请的先得到。

　　３，申请之后，任务池里就不出现在条工作而是进入了被申请人的待办里。

　　４，如果申请人处理不了这件工作，他可以把该工作放入任务池中，放入任务池的工作可以被其他的Ｎ个人可以看到。

　　５，不管取回与放入，都需要写入流程日志。

　　6，任务池的设计：

　　１，在流程引擎控制表 WF\_GenerWorkFlow 中增加了如下字段: 　　　　 TaskSta,　枚举类型， 0=不启用，1=共享，2=被申请。 　　　　 TodoEmpsNum, Int类型，处理人员数量，当前节点工作有多少人待办，如果此值大于等于２，讨论共享任务才有意义。 TodoEmps varchar(500) 待办人员,格式为　处理人员编号，处理人员名称；　比如：guobaogeng,郭宝庚;fuhui,福惠;

　　２，在流程引擎控制表的人员列表中　WF\_GenerWorkerList 中对于IsEnable字段,进行扩充了，如果等于　-1=任务池禁用，０＝分配工作被屏蔽，１＝正常。 　 　　３，增加了一个功能界面　/WF/TaskPoolSmaller.aspx 次功能界面是有在任务池启用才能被使用。

　　４，在流程属性里增加一个字段　IsEnableTaskPool 是否启用任务池？　此字段用来控制该流程是否启用任务池。

**任务池的ＳＤＫ接口**：

　　１，获得共享任务：　DB\_TaskPool

　　２，获得共享任务中我的待办：DB\_TaskPoolOfMy

　　３，从共享任务中取回：Node\_TaskPoolTakebackOne

　　４，把取回的共享任务放入任务池：　Node\_TaskPoolPutOne

# 总结说明

## 总结概述

在BPM领域支持流程运转的理论模型有多种，有的21种、28种，32种。每种模式都代表了这种模式的理论设计者研究者的人员主张、思想。这些模式尽可能的，完全去覆盖到现实生产、工作、应用上的流程流转规则。 但是任何领域都有自己未探索到的流程运转可能，如果说自己的理论能够涵盖所有的应用需求，那就说的有点太早。

就类似于世界上的教派或者哲学家一样，不同的教主与哲学家有不同的世界观，不同的世界认识，都对自己的“追随者”来宣扬自己的主张、观点，但是他们也能“和谐”相处，对于同一个事物都有自己的观点，也有相同的认识，同时也有自己的“理论”解释不了的事物。

在jbpm中有21种流程运行模式，在AWCP中都可以有相应的设计对应,如果AWCP与他自身带的ccfom结合一起工作，更能灵活的适应国内各种复杂的环境要求。

一下文章介绍了AWCP的流程的4大运行模式，以及是如何支持各式各样的流程变化。

## AWCP控制流程运行模式的方式

AWCP使用节点模式、流程属性、节点属性来控制流程运转规则，以不不变应万变的方式来应对各种生产、管理模式下的流程运转需要。

AWCP的概念比较简单容易理解，它有4大运行模式，分别是线性流程、同表单分合流、异表单分合流、父子流程。

**节点类型：**普通节点、分流节点、合流节点、分合流节点、子线程节点。



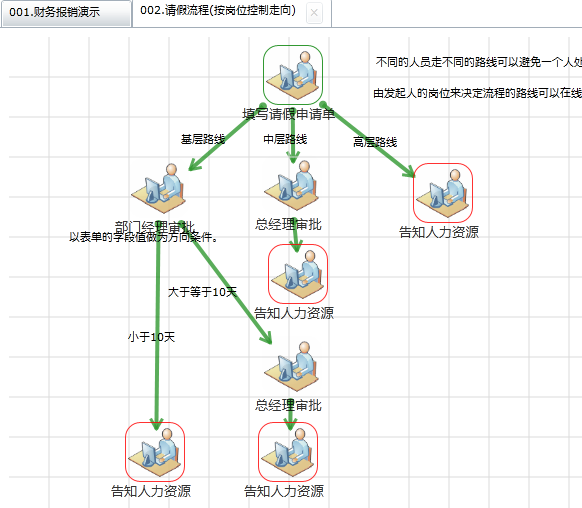
**节点多人处理模式：**抢办模式、协作模式、队列模式、共享模式。

与父子流程相关的一些属性控制等等，如下图。

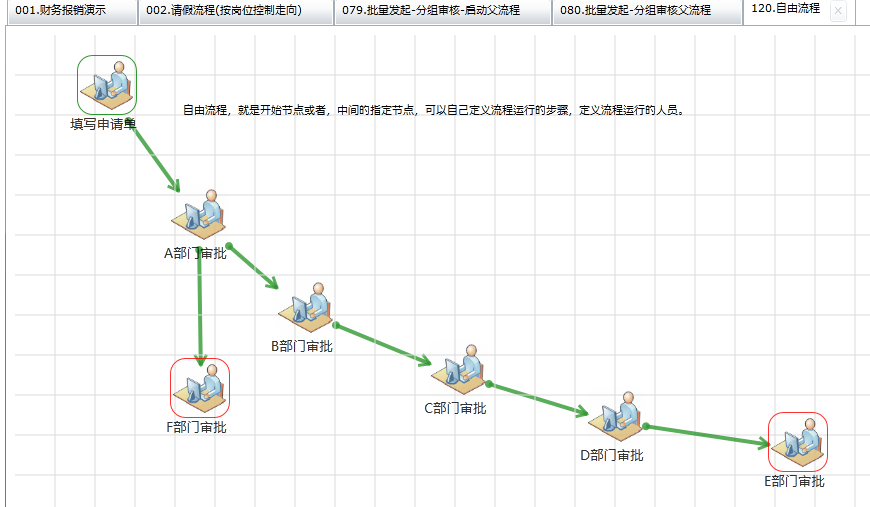


## AWCP 的4大流程运转模式的解释

**线性流程：**在统一个时间点，只有一个活动的节点，每个节点都按照一定的步骤，一定的顺序就执行，在特定的节点允许跳转、允许自定义以后的流程运行步骤，我们把这样的流程称为线性流程，也叫顺序流程。在一个系统中大约85%以上的流程，都是该类流程。

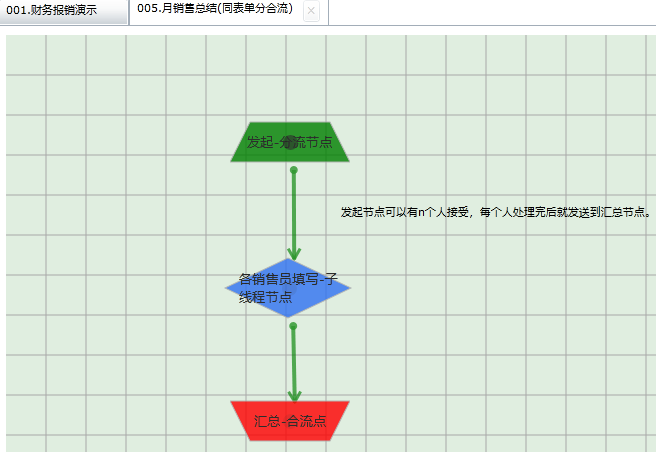


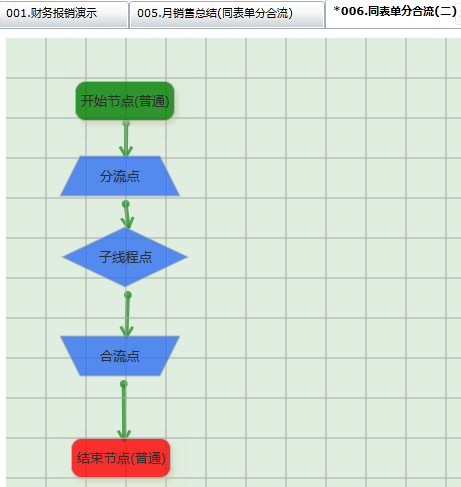
自由流程：

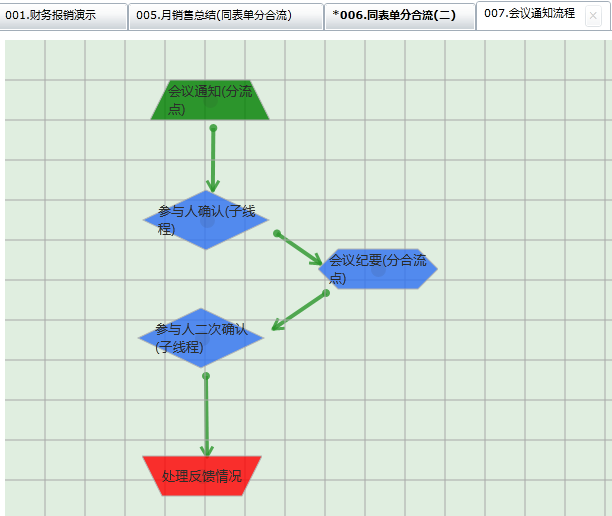


**同表单分合流：**AWCP的强大在于AWCP集成了szcloud。流程与表单是相辅相成的，就好像一个火车头与车厢的关系，同表单的分合流，就是一个节点上可以存在多个活动的线程，线程与处理人是1对1的关系，每个处理人的内容都是一样的，所以我们成为同表单的分合流。

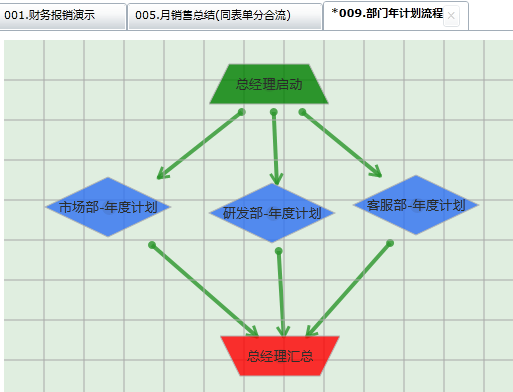
如下图：

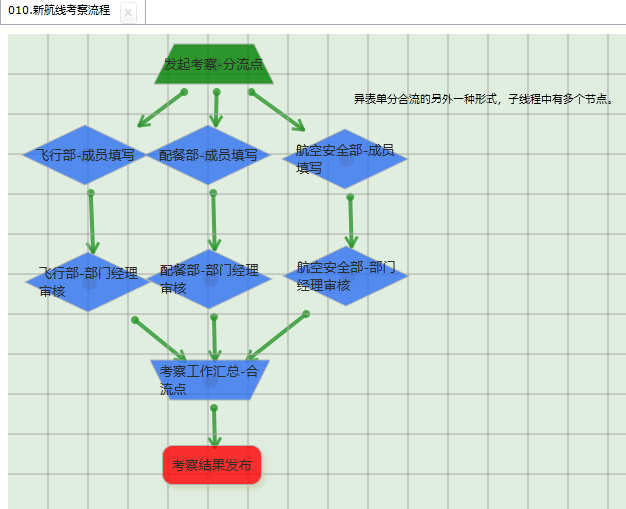






**异表单分合流：**如果您了解了同表单的分合流，就不难了解这个流程模式的概念了。如下图：





父子流程： 父子流程，就是在一个流程A的一个节点上，可以启动另外的一个流程B，当前的流程我们称为父流程，被启动的流程成为子流程，如果子流程B的一个节点上启动另外一个流程C，那末B流程与C流程就是父子流程， A 与C就是爷爷与孙子关系，但是AWCP不研究跨界关系，仅仅研究相近的两个流程之间的关系。

比如：一个工程类的流程，调用施工流程、与催款流程，如下图：



==========OVER==========